Alfred Kobsa Wolfgang Wahlster (Eds.)

User Models in Dialog Systems

With Contributions by

S. Carberry J.G. Carbonell D.N. Chin R. Cohen
J. Fain Lehman T.W. Finin A. Jameson M. Jones
R. Kass A. Kobsa K.F. McCoy K. Morik
C.L. Paris A.E. Quilici E. Rich K. Sparck Jones
W. Wahlster

With 113 Figures



Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong

EX-1006
U.S. Patent 8,073,681

Alfred Kobsa
Wolfgang Wahlster
Department of Computer Science
University of Saarbrücken
D-6600 Saarbrücken 11
Fed. Rep. of Germany

ISBN-13:978-3-642-83232-1 e-ISBN-13:978-3-642-83230-7
DOI: 10.1007/978-3-642-83230-7

Library of Congress Cataloging-in-Publication Data.
User models in dialog systems / Alfred Kobsa, Wolfgang Wahlster (eds.);
with contributions by S. Carberry ... (et al.).
p. cm. — (Symbolic computation. Artificial intelligence)
Based on UM86, the first international workshop on user modeling,
held in Maria Laach, Germany.
Bibliography: p. Includes index.
ISBN-13:978-3-642-83232-1 (U.S.: alk. paper)
1. Human-computer interaction. 2. Artificial intelligence. 3. Question-answering systems. I. Kobsa, Alfred. II. Wahlster, Wolfgang. III. Series. QA76.9.H85U74 1989 004'.01'9—dc19 88-28156 CIP

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. Duplication of this publication or parts thereof is only permitted under the provisions of the German Copyright Law of September 9, 1965, in its version of June 24, 1985, and a copyright fee must always be paid. Violations fall under the prosecution act of the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1989
Softcover reprint of the hardcover 1st edition 1989

The use of registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Media conversion: RL-SOFT, Robert Laudensack, Cham, Fed. Rep. of Germany
2145/3140-543210 Printed on acid-free paper

Preface

User models have recently attracted much research interest in the field of artificial intelligence dialog systems. It has become evident that flexible user-oriented dialog behavior of such systems can be achieved only if the system has access to a model of the user containing assumptions about his/her background knowledge as well as his/her goals and plans in consulting the system. Research in the field of user models investigates how such assumptions can be automatically created, represented and exploited by the system in the course of an "on-line" interaction with the user. The communication medium in this interaction need not necessarily be a natural language, such as English or German. Formal interaction languages are also permitted. The emphasis is placed on systems with natural language input and output, however.

A dozen major and several more minor user modeling systems have been designed and implemented in the last decade, mostly in the context of natural-language dialog systems. The goal of UM86, the first international workshop on user modeling, was to bring together the researchers working on these projects so that results could be discussed and analyzed, and hopefully general insights be found, that could prove useful for future research. The meeting took place in Maria Laach, a small village some 40 miles south of Bonn, West Germany. 25 prominent researchers were invited to participate. The pleasant setting of the conference site close to the medieval abbey of Maria Laach and the volcanic Lake Laach fostered a nice atmosphere for intensive discussions and the exchange of ideas until the early hours of the morning.

This book is one of the documented results of the workshop (the other is a special issue of the *Computational Linguistics* journal containing topics which are more closely related to natural language processing). The volume consists of four parts: Part 1 contains three survey chapters, the first introducing the field of user modeling as a whole, and the two others analyzing an important modeling technique and the contents of user models, respectively. Part 2 deals with the construction, and Part 3 with the exploitation, of user models by a dialog system in an ongoing dialog. Part 4 analyzes shortcomings of user models as developed to date and outlines promising directions for future research.

Although this volume is based on a workshop, it was designed right from the beginning to form not only a collection of unintegrated conference papers, but to provide a rather coherent survey of the field of user modeling. Each part is prefaced by a separate editorial introduction which classifies the individual contributions within the topic of the respective subfield, and summarizes their central

issues. Both in the prefaces and in the individual chapters, cross-references to other contributions have been included whenever appropriate. The references of all contributions have been compiled into a single bibliography and were supplemented by a number of papers which have not been cited by the contributors, but certainly form part of a representative bibliography of the field. All authors were able to contribute to this extended bibliography. A subject index facilitates the comparison of how individual issues are treated by different authors. A citation index integrated into the bibliography allows for easy back-reference from the titles to the citing contributors and the page numbers where cited.

An unusual amount of discussion and reviewing has been necessary to achieve this coherency: speakers of the workshop had to distribute tentative versions of their papers long before the workshop so that their talks could be discussed more deeply at the meeting. Tape recordings of these discussions were made and distributed to all speakers, and written reviews of each paper were prepared by all participants. Selected speakers were then invited to submit their revised and extended papers for publication in the book. The editors made additional comments which authors had to take into account. Final problems were discussed at an editorial meeting at the IJCAI-87 conference in Milan. Altogether, the written correspondence *about* this book would certainly fill another volume.

The publication of the book would not have been possible without the assistance and support of a number of individuals and institutions, to whom we would like to express our gratitude:

- The Collaborative Research Programme on Artificial Intelligence and Knowledge-Based Systems (SFB 314) of the German Science Foundation (DFG), which funded the workshop;
- Mark Line and Bernd Schäfer who prepared transcripts and rendered the editors' English intelligible even to native speakers;
- Bernd Nessen and Sokrates Evangelidis for their work as sound engineers;
- Susi Schild and Christa Weinen who carefully compiled the bibliography; and
- Doris Borchers, Gabriele Jacquinot, and Parinaz Mohammadzadeh for their administrative assistance.

The international user modeling community owes a lot to them.

Saarbrücken, April 1988 *Alfred Kobsa* *Wolfgang Wahlster*

Table of Contents

Part I. A Survey of User Modeling

Introduction 2

1 **User Models in Dialog Systems** 4
Wolfgang Wahlster and Alfred Kobsa

1. Introduction 4
2. Constructing User Models 10
3. Representing User Models 14
4. Exploiting User Models 18
5. Open Questions and Future Research in User Modeling 29
6. References 33

2 **Stereotypes and User Modeling** 35
Elaine Rich

1. Introduction 35
2. The Definition of a Stereotype 36
3. The Space of User Models 38
4. Stereotypes and User Modeling 39
5. Stereotypes and Contradiction Resolution 41
6. Adaptation of Stereotype Systems 48
7. Parallel Models of System Knowledge and of Users 50
8. Summary 51
9. References 51

3 **A Taxonomy of Beliefs and Goals for User Models in Dialog Systems** 52
Alfred Kobsa

1. Introduction 52
2. Beliefs, Goals and Plans 53
3. Basic Beliefs 56
4. Basic Goals 59
5. Beliefs with Respect to Other Agents' Beliefs and Goals 60
6. Goals with Respect to Other Agents' Beliefs and Goals 63
7. A Classification of Existing User Models 65

8. Discussion 67
9. References 68

Part II. Building User Models

Introduction 70

4 **KNOME: Modeling What the User Knows in UC** 74
David N. Chin

1. Introduction 74
2. Internal Representation of Users 78
3. Deducing the User's Level of Expertise 83
4. Modeling UC's Knowledge 97
5. Exploiting KNOME 103
6. Conclusion 104
7. References 107

5 **Detecting and Responding to Plan-Oriented Misconceptions** 108
Alexander Quilici

1. Introduction 108
2. An Explanation-Based Approach 110
3. Representing User and Advisor Beliefs 111
4. Explanation-Based Misconception Recognition and Response 114
5. A Taxonomy of Potential Explanations 116
6. A Detailed Process Model 120
7. Accessing Advisor Planning Knowledge 122
8. Related Work 125
9. Implementation Details 128
10. Limitations and Future Work 128
11. Conclusions 131
12. References 132

6 **Plan Recognition and Its Use in Understanding Dialog** 133
Sandra Carberry

1. Introduction 133
2. Plan Recognition in Dialog Systems 135
3. Inferring and Modeling the Task-Related Plan 136
4. Application of User Models 144
5. Improving Plan Recognition 150
6. Constructing and Exploiting Other Components of a User Model 160
7. Conclusions and Current Research 161
8. References 162

7 **Learning the User's Language: A Step Towards Automated Creation of User Models** 163
Jill Fain Lehman and Jaime G. Carbonell

1. Introduction: Adaptable Interfaces 163
2. Foundations: Least-Deviant-First Parsing and MULTIPAR 165
3. CHAMP: Design for an Adaptive Parser 170
4. Hidden Operator Experiments with Professional Secretaries 182
5. Concluding Remarks 192
6. References 194

Part III. Exploiting User Models

Introduction 196

8 **The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise** 200
Cécile L. Paris

1. Introduction 200
2. Identifying What Needs to Be in the User Model 203
3. Two Descriptions Strategies Found in Texts: Constituency Schema and Process Trace 205
4. Mixing the Strategies 212
5. TAILOR 221
6. Further Work and Related Issues 226
7. Conclusions 231
8. References 232

9 **Highlighting a User Model to Respond to Misconceptions** 233
Kathleen F. McCoy

1. Introduction 233
2. Knowledge Available 235
3. Related Work on Correcting Misconceptions 237
4. Misclassifications 238
5. Misattributions 240
6. A Rule for Choosing a Strategy 241
7. Highlighting and Object Similarity 242
8. Object Perspective 245
9. Using Perspective to Set *f* 247
10. Modeling a Domain with Perspectives 248
11. Choosing the Active Perspective 251
12. Perspective's Influence on Responses 252
13. Conclusions 253
14. References 254

10 **But What Will the Listener Think? Belief Ascription and Image Maintenance in Dialog** 255
Anthony Jameson

1. Introduction 255
2. Situation 1: Generating an Informative Monolog 260
3. Situation 2: Positive or Negative Bias 269
4. Situation 3: Anticipating the Pragmatic Interpretation of Comments 271
5. Situation 4: Discrepancy Between Actual and Projected Biases 274
6. Situation 5: Responding to Specific Questions 278
7. Situation 6: Establishing a Desired Projected Bias 286
8. Situation 7: Discrepancy Between Actual and Projected Ascriptions 293
9. Situation 8: Uncertainty in the Listener About the Speaker's Ascriptions 299
10. Situation 9: Uncertainty in the Speaker About Her Projected Ascriptions 303
11. Conclusions 307
12. References 311

11 **Incorporating User Models into Expert Systems for Educational Diagnosis** 313
Robin Cohen and Marlene Jones

1. Introduction 313
2. Application Domain 314
3. Designing the System 316
4. The User Model Structure 320
5. Related Work 329
6. A General Proposal for User Modeling in Expert Systems 331
7. Current Status and Future Work 331
8. Conclusions 332
9. References 333

Part IV. Shortcomings of Current Models, Prospects for the Future

Introduction 336

12 **Realism About User Modeling** 341
Karen Sparck Jones

1. Introduction 341
2. What Is Being Modeled 342
3. What Modeling Is For 346
4. What Modeling Is From 352

5. Rational Principles for Modeling 357
6. Conclusion 360
7. References 363

13 User Models and Conversational Settings: Modeling the User's Wants 364
Katharina Morik

1. Introduction 364
2. Terminology 365
3. Conversational Settings 367
4. User Modeling in HAM-ANS 376
5. Conclusion 383
6. References 385

14 Student Modeling in Intelligent Tutoring Systems – Implications for User Modeling 386
Robert Kass

1. Introduction 386
2. Intelligent Tutoring Systems 387
3. Student Modeling 389
4. Four Intelligent Tutoring Systems That Model Students 393
5. Further Work in Student Modeling 403
6. Summary: Comparing User Modeling and Student Modeling 406
7. Conclusion 410
8. References 410

15 GUMS – A General User Modeling Shell 411
Timothy W. Finin

1. Introduction – The Need for User Modeling 411
2. What Kind of User Model? 412
3. A General User Modeling System 417
4. The Current GUMS System 428
5. The GUMS Command Language 429
6. Conclusions 430
7. References 430

Appendices

Bibliography 432

Subject Index 459

List of Contributors 468

Part I

A Survey of User Modeling

Introduction

The aim of Part I is to give a survey of current research on user modeling in artificial intelligence dialog systems. The first chapter introduces the field as a whole, and the two other contributions analyze an important modeling technique and the contents of user models, respectively. The chapters describe the aims and the scope of the field, and discuss influential systems that have been implemented, as well as important general results that have been obtained. Part I thus provides a good basis for the more technical chapters in Parts II and III, though all chapters are self-contained and do not presume that other contributions have already been read before.

An analysis of the field of user modeling from different points of view can also be found in Part IV, which is devoted to the discussion of shortcomings of current systems, and proposals for future research. Readers who do not wish to proceed too deeply into descriptions of particular systems are therefore encouraged to have a look at the final part after reading the introductory chapters.

The survey chapter by **Wolfgang Wahlster** and **Alfred Kobsa** is divided into five subsections. First, reasons are identified as to why user modeling has now become so important for artificial intelligence dialog systems. Definitions of the central notions in this field are given, and the relationship to related areas of research within and outside of artificial intelligence is outlined. Section 2 describes by what means and from what information sources user modeling components can construct a user model in the course of an ongoing dialog. Section 3 surveys the representation schemes that have been proposed to date for the representation of the system's assumptions about the user. Possible applications of user models are then extensively reviewed, and some social problems associated with practical applications are discussed. Finally, a number of open questions in user modeling are pointed out, and prospects for application-oriented research are considered. Throughout the chapter, references to the other contributions are made when appropriate in order to facilitate their classification within the field as a whole.

The chapter by **Elaine Rich** analyzes the stereotype approach, an important technique for user modeling that has been widely employed (e.g. [CHIN*, FINI*, MORI*]). A stereotype is a collection of user properties that often co-occur. In the stereotype approach, these properties are attributed to the user by a modeling component as soon as a limited number of preconditions are fulfilled. The great advantage of this technique stems from the fact that predictions concerning the user

can be acquired from an amount of evidence that is small enough to allow for easy acquisition *before* these predictions are needed.

As a matter of fact, Rich is certainly a suitable person to review the current position of this technique in user modeling, since she introduced it into the field a decade ago. In her present paper, she first defines the components of stereotypes and characterizes the types of user models that can profit the most from stereotype techniques for the acquisition of assumptions. The central issues of her chapter (and a strong contribution to the theoretical underpinnings of the stereotype approach) is a general discussion of *contradiction resolution* and of *stereotype adaptation*: Since the use of stereotypes constitutes a form of default reasoning, conflicts may arise among stereotypical assumptions from different sources, or between stereotypes and direct observations. Moreover, stereotypical predictions which were originally correct may become outdated. Rich discusses both a number of heuristics for resolving such conflicts, and more elaborate strategies which are related to the general methods of default and probabilistic reasoning as developed in AI research.

With stereotype adaptation techniques, a dialog system can exploit the experiences gained from the application of stereotypes to a number of users, and modify the original stereotypes. In practice, the integration of such techniques into any user modeling component which uses stereotypes is highly desirable, since the reason for choosing particular values for stereotype properties is often just the intuition of the designer.

Alfred Kobsa's contribution is concerned with an analysis of the *contents of user models*. He investigates the types of beliefs and goals that have been modeled by user modeling components developed to date (and adds some more which have not yet been, but ought to be modeled), and puts them into a taxonomical order. Apart from its theoretical contribution, such a taxonomy can be practically exploited for supporting designers of application systems in investigating what types of user beliefs and goals are relevant – and should therefore be modeled – in their respective domains, and for judging the expressive power of representation schemes for beliefs and goals. Using this taxonomy, Kobsa analyzes several major existing user modeling components with respect to what is being modeled by them. A striking difference is found between the number of modeled belief and goal types in those systems that employ the user model for explicit speech act recognition and dialog planning, and those which do not. This observation might have implications for the development of user modeling shell systems (see [FINI*]), which can presumably be much less complex if not developed for the former purpose.

1 User Models in Dialog Systems [1]

Wolfgang Wahlster and Alfred Kobsa

Abstract

This chapter surveys the field of user modeling in artificial intelligence dialog systems. First, reasons why user modeling has become so important in the last few years are pointed out, and definitions are proposed for the terms 'user model' and 'user modeling component'. Research within and outside of artificial intelligence which is related to user modeling in dialog systems is discussed. In Section 2, techniques for constructing user models in the course of a dialog are presented and, in Section 3, recent proposals for representing a wide range of assumptions about a user's beliefs and goals in a system's knowledge base are surveyed. Examples for the application of user models in systems developed to date are then given, and some social implications discussed. Finally, unsolved problems like coping with collective beliefs or resource-limited processes are investigated, and prospects for application-oriented research are outlined. Although the survey is restricted to user models in natural-language dialog systems, most of the concepts and methods discussed can be extended to AI dialog systems in general.

1. Introduction

1.1. Why User Models?

During the 1970s, many special-purpose natural-language (NL) *interfaces* were developed for various domains of discourse, e.g. moon rock samples, airline fares, computer installations, payroll data, aircraft maintenance data, or university courses. These systems had no interests beyond providing the information-seeking user with relevant data by just *responding* in a mechanically cooperative way to the user's questions. The conversational setting for such dialog systems was somewhat unnatural compared to human dialogs. The user of these systems had to find an appropriate

[1] This research was supported by the Special Collaborative Research Programme on Artificial Intelligence and Knowledge-Based Systems (SFB 314) of the German Science Foundation. We would like to thank G. Ferrari, Ch. Habel, M. Hecking, A. Jameson and an anonymous referee for their comments on earlier versions of this chapter. This is an extended and revised version of [WAHL86b].

question strategy for getting the information s/he thought might be relevant for solving his/her problem, and could hardly expect any assistance from the system in this respect.

In the 1980s, a new class of task-oriented dialog systems has emerged, all of which are mainly characterized by the *conversational roles* which the system and the user are expected to take on. Here, more realistic conversational settings are modeled in which the system may play the part of, for example,

- a clerk in an information booth at a train station (cf. [ALLE79]),
- a hotel manager who tries to rent all his rooms (HAM-ANS, cf. [JAME80]),
- a librarian who recommends novels that the user might like to read (GRUNDY, cf. [RICH79a, RICH79b]),
- a nosey guest at a cocktail party who wants to find out as much as possible about the user (ULLY, cf. [HAYE76]),
- an expert who assists an apprentice in repairing an air compressor (KAMP, cf. [APPE82b, APPE85]),
- a UNIX and a SINIX consultant for beginners (UC, cf. [WILE84, WILE86, CHIN*]; SC, cf. [KEMK86, NESS87, HECK88]),
- a tax advisor who assists the user in filling in his/her income tax form (XTRA, cf. [KOBS86b, ALLG89]).

These systems are designed as *active* dialog partners who engage in a mixed-initiative dialog [WAHL84]. In contrast to the NL interfaces mentioned above, the user of these systems needs no prepared question strategy since the system tries to recognize the user's *intention* with respect to the domain of discourse in order to exhibit more cooperative dialog behavior (e.g. in order to provide better advice). Although even in NL interfaces such a characteristic was necessary to a limited extent, it is particularly in the above conversational settings that the construction and use of an *explicit model of the user's beliefs, goals and plans* becomes a central problem.

Thus, one reason for the recent emphasis on user modeling is the fact that such models are necessary prerequisites in order for a system to be capable of exhibiting a wide range of cooperative dialog behavior (cf. also [CARN83c, RICH83, SPAR84, BOGU85, KASS88c]). A cooperative system must certainly take into account the user's goals and plans, his/her prior knowledge about a domain, as well as false conceptions a user may possibly have concerning the domain (empirical evidence for this in the case of human expert consultation can be found in [POLL82a]). Thus it is no longer only the user's task to construct a mental model of the technical functioning of the system. Instead, it should also be up to the system to form assumptions about what the user believes, wants and plans, i.e. to develop a model of the user. Ideally, neither the user's nor the system's modeling task should differ from that which is required in person-to-person communication.

A simple example for such cooperative dialog behavior might look as follows:

(1) User: Where can I find the nearest gas station?
System: The nearest one which is still open is located ...

In order to respond in this cooperative way, a system must discover the presumable plans underlying the user's question, represent them in its knowledge base, examine them for hidden obstacles, and provide information to enable the user to overcome these obstacles. A user model is an indispensable prerequisite for these complex inference processes.

A second reason for the recent emphasis on user modeling is that it has become evident in the last few years that a model of the user is also an important basis for intelligent dialog behavior in general, regardless of whether the dialog is cooperative or not. Such models are required for identifying the objects which the dialog partner is talking about, for analyzing a non-literal meaning and/or indirect speech acts in his/her dialog contributions, and for determining what effects a planned dialog contribution will have on the dialog partner, etc. (cf. [ALLE86, ALLE87]). Thus a user model does not just sweeten-up a dialog system to render it more cooperative. Rather, user models constitute an indispensable prerequisite for any flexible dialog in a wider domain. They interact closely with all other components of the system and often cannot easily be separated from them.

1.2. Some Basic Definitions

Although in this chapter we will restrict the discussion to user models in NL dialog systems, most of the concepts and methods discussed can be extended to AI dialog systems in general. Furthermore, many of the user modeling components developed for NL dialog systems accept or generate expressions of some formal representation language, since they belong to the language-independent part of the NL system. In this context, we will use the following basic definitions:

> A *user model* is a knowledge source in a natural-language dialog system which contains explicit assumptions on all aspects of the user that may be relevant to the dialog behavior of the system. These assumptions must be separable *by the system* from the rest of the system's knowledge.
>
> A *user modeling component* is that part of a dialog system whose function is to incrementally construct a user model; to store, update and delete entries; to maintain the consistency of the model; and to supply other components of the system with assumptions about the user.

We stress that user models cannot be simply defined as information that the system has about its users. Consider an NL interface to a database, which contains the relation EMPLOYEE (EMP#, NAME, AGE, BONUS) with (26, Jones, 32, 40) as one of its tuples. When Mr. Jones happens to be the user of such a system and asks 'What is my bonus?', his query is translated e.g. into (Select BONUS from EMPLOYEE where NAME = Jones), and the system responds '40'. In this case, the system has information about the user, but we would not like to say that its response was based on a user model, since the system does not view the database tuple as an explicit assumption about its current user (cf. [WAHL88]).

In many conversational situations, the system has to construct and exploit models of more than one agent. Imagine an input to a hypothetical 'marriage guidance

system' such as 'This is Bob Jones. My son has problems with his wife. She always .... But I think that my son ...'. In this case, the system's advice should be based on models of Bob Jones, his son, his daughter-in-law and the relationships between these agents. With the definition given above we propose to restrict the term 'user model' to assumptions about aspects of the human directly interacting with the system. In addition, however, we introduce the concept of an *agent model* (or 'actor' model in [KOBS85a]) as a superconcept for 'user model'. Thus, in the example discussed, there are three agent models, only one of which is a user model. For user modeling the system can evaluate the observed behavior of its dialog partner, whereas in the general case of agent modeling this is not necessarily possible. It follows that not all of the techniques developed for user modeling can be generalized to agent modeling.

1.3. User Modeling in Related Areas

1.3.1. User Modeling Within Artificial Intelligence Research

Within the area of artificial intelligence (AI), research on user modeling in NL dialog systems is related to several other fields, namely to the work on *intelligent computer-aided instruction (ICAI), multiple-agent planning systems, text comprehension and generation, intelligent help systems, game playing*, and *expert systems* (see Figure 1).

ICAI systems [SLEE82c, DEDE86] typically incorporate a "student model" which represents the student's understanding of the material to be taught. A simple modeling technique in this field is the overlay approach (e.g. [CARR77, CLAN79]), in which the student's knowledge is represented as a subset of the system's *knowledge* about the domain (cf. Section 3 and [KASS*]). Compared to NL dialog systems, however, the possible user's input to ICAI systems is usually very restricted. Sometimes only expressions in a formal language are accepted, such as mathematical formulas or statements in a programming language.

Research in the field of multiple-agent planning (e.g. [POWE79, KONO80, ROSE82, GEOR83, ROSE85, WERN88]) investigates the problems that a group of agents have in collectively constructing and/or executing a plan for a given task. To coordinate their actions, each agent has to represent the beliefs and goals of the others. Much of this work is grounded on the belief representations developed in the possible-world approach (see Section 3). Problems of natural-language communication between agents are not studied in this field of research. Moreover, for representational purposes, a number of idealizations are typically made with respect to the kinds of beliefs and goals which agents can hold. Sometimes agents cannot possess beliefs which are contradictory to other agents' beliefs (e.g. to those of the system), or cannot hold various forms of mutual and infinite-reflexive beliefs (see Section 2.3). Contradictory beliefs on the part of different agents, however, are quite common in man-computer interaction,[2] and agents which hold various forms of mutual or infinite-reflexive beliefs are practically ubiquitous in this field.

---

[2] A lot of work in the field of user modeling is devoted to recognizing user misconceptions, i.e. conflicts between system and user beliefs (e.g. [SCHU83b, MCCO*, QUIL*]).

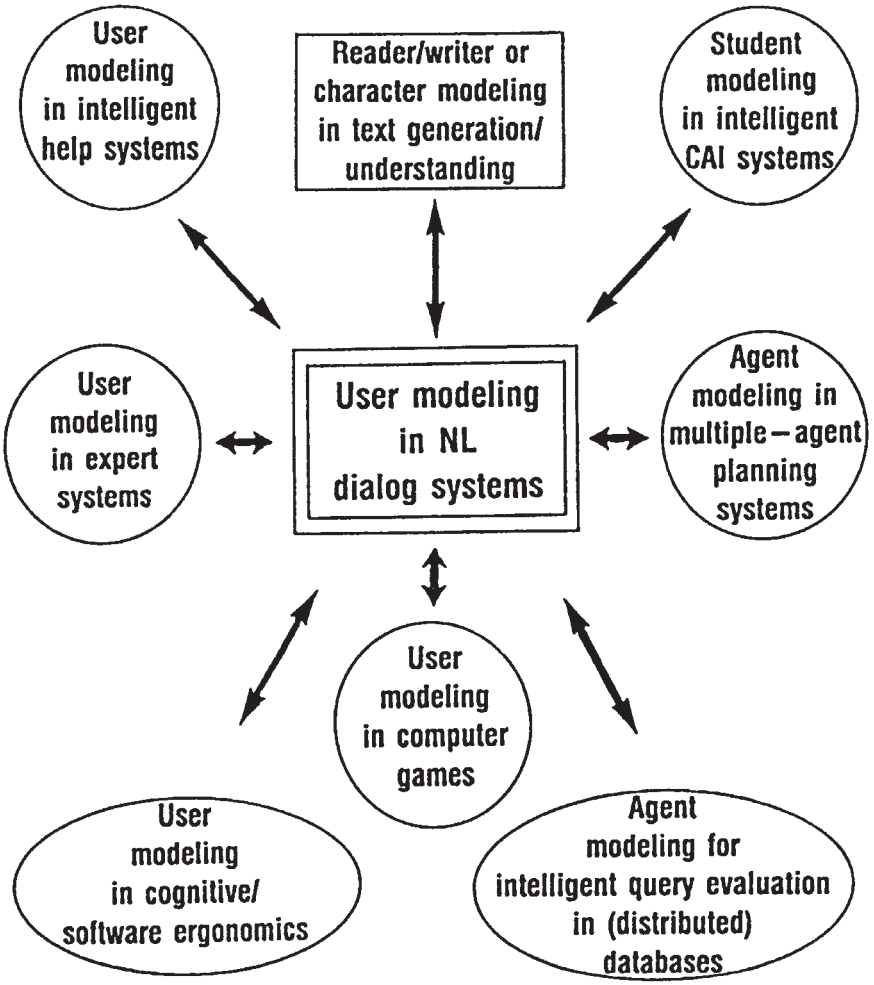


Figure 1. Relationships to other research areas

The adversary nature of two-person games like chess, checkers or backgammon makes game-playing another challenging domain for user modeling research, since in contrast to most other dialog situations, the best systems here behave in a most uncooperative way. It is clear that for brilliant play a model of the opponent's strengths and weaknesses as well as an understanding of his/her playing style is needed. Such a model of the opponent forms the basis for imagining the likely consequences of a planned move. Suppose that the system, being an end game expert in chess, believes that its opponent is a very weak end game player. In such a situation the system will not accept a proposed draw, even when at a disadvantage. Today, the best game-playing systems use an asymmetric evaluation function (see e.g. [HORA87]) and analyze the opponent's previous moves in order to construct a model of his playing strategy. Unfortunately, the user modeling techniques developed so far in this subfield of AI are extremely domain-dependent (they are mostly based on various numeric parameters and scoring functions), so that they cannot be transferred directly to other communicative situations.

1.3.2. User Modeling Outside of Artificial Intelligence Research

Outside of artificial intelligence, a growing interest in user models can be observed in the field of *Human-Computer-Interaction*[3] (for surveys see [CLOW85, MURR87a, MURR87b]). In HCI, however, the notion of 'user model' is employed in a much wider sense than in artificial intelligence research. On the most general level, user models are regarded as forming some kind of mental model [GENT83] which, according to current cognitive psychology, persons usually form of the things and people with whom they interact. Such mental models provide predictive and explanatory power for understanding the interaction. Entries in AI user models, by contrast, need not necessarily be psychologically relevant, i.e. correspond to "mental entities" of the user. They must only enable the system to reconstruct the user's beliefs and goals in sufficient detail.

The glossary in [NORM86b] discusses three different types of models that are called 'user models' in HCI research: "[...] the individual's own personal idiosyncratic model; the generalized 'typical user' model that the designer develops to help in the formulation of the design model; or the model that an intelligent program constructs of the person with which it is interacting." In artificial intelligence, the notion of 'user model' is only used in the last sense.

The aim of research in HCI is not necessarily to develop computer systems that construct a model of the user. Instead, knowledge about users' mental models should enable designers to build systems that can be more easily understood by users, or at least to predict learnability and performance. In order to make such models explicit, formal representation methods have recently been proposed for describing the user's and the system's knowledge about the task structure. These representations include state transition networks, formal grammars, and some sort of production systems. Perhaps closer convergence of research pertaining to user modeling in HCI and in AI dialog systems (which are currently nearly independent lines of research) can be achieved on a representational basis as soon as the HCI formalisms have been further developed.

It is worth noting that dialog-based user models can help to improve the performance of systems even for standard tasks like database retrieval. Botman, Kok and Siklossy [BOTM87] describe a so-called *active* database (ADB) system which extends the literal response to a user question by including extra information the user might be interested in, so that less interaction with the system will be necessary. In many retrieval situations a user is not able to give a precise and complete explication of his/her preferences at the beginning of the dialog. In the ADB system the user can indicate which items mentioned in the response to his/her original query s/he finds interesting. Using knowledge about similarities between database values or attributes, the system tries to extract the characteristics of the items upon which the user's choice was based. These assumptions about the user's interests are stored as user profiles together with a certainty factor in the user model. The profiles are used to compute individual interest values for all relevant tuples in the database, which are then used to select and rank information before presentation to the user.

---

[3] Related notions are 'Cognitive Ergonomics', 'Software Ergonomics', 'Cognitive Engineering', etc.

For example, when the user wants to see information about second-hand cars under $ 2,000.00 and then marks several of the displayed BMWs and Audis as interesting, the system may conclude that this choice was based on the 'make' and 'price' attributes, but not on 'color'. When updating the display, the system removes the 'color' attribute and shows more BMWs and Audis, but also includes Alfa-Romeos, because this make was often considered similar to BMWs in previous dialogs.

2. Constructing User Models

In this section, a number of knowledge sources will be identified from which information about the user can be obtained, and a number of techniques for acquiring a user model in the course of a dialog will be discussed.

2.1. Default Assumptions

In systems with a limited area of application, various beliefs and goals can be attributed to any user of the system, as long as there is no evidence to the contrary. Such default assumptions made by the system may concern, for example,

- *The user's general knowledge*: A management information system (MIS) with a user modeling component, for instance, can assume that all of its users possess basic knowledge in business administration.
- *The user's beliefs about individual facts*: An MIS installed in a firm can also assume that the user is familiar with the principal facts concerning the firm.
- *The user's goals*: All users of a hotel reservation system (cf. [JAME80, HOEP83b]) can be assumed to be searching for rooms which meet their requirements.

Other possible default assumptions concern the user's beliefs about the system's goals (e.g. that the system wants to rent its rooms), or assumptions about the user's beliefs about the system's beliefs, etc.

2.2. Initial Individual User Models

One criterion which helps to differentiate between the various dialog systems that include a user modeling component is whether

(A) The system can, at the beginning of a dialog, access an already existing model of a particular user which it had constructed during previous interactions with him/her.

(B) The system possesses no previously acquired information about individual users at the beginning of the dialog.

In the former case, the recorded user model will probably prove to be a valuable source of information for the current dialog as well. An example of a system with

such an initial user model is GRUNDY [RICH79a, RICH79b] which, at the end of a dialog session, records all information about the user inferred from his/her dialog behavior in a corresponding file. Whenever that user returns to the system and types in his/her name, this file will be retrieved and used as the initial user model.

Today, however, the latter type of system is developed and implemented much more often than the former. This is due not only to the legal, social and ethical problems of security and privacy in such systems, but also to the fact that for many prospective applications in the current era of distributed systems the probability that a user will consult the same system more than once is quite low.

2.3. Assumptions Based on the User's Input

Assumptions formed on the basis of the user's input into the system are the most direct and, hence, also the most reliable in problem-solving domains.

There are various techniques for transforming a user's input into assumptions about him which differ in complexity and in the reliability of the inferred assumptions. The simplest type occurs if the user directly expresses a belief or goal, as in

(2) User: I want to buy your new 500 XGL.
Assumption of the system: The user wants to buy one 500 XGL.

Unfortunately, such direct statements of the user about his/her beliefs or goals are usually very rare. Much more frequent are cases in which assumptions about the user have to be indirectly inferred from the user's input. One type of such inferences is purely *structure-based*. In this case, only the form of a user's input is used for drawing inferences about him/her, and not its meaning. The VIE-DPM system, for instance, forms the following assumptions based on a user's wh-question, and enters them into its knowledge base [KOBS85d, KOBS86a].

(3) User: To whom does Peter give the book?
Assumptions of the system:
(a) The user believes that Peter and the book exist and that Peter gives the book to somebody whose identity is unknown to the user.
(b) The user believes that the system knows to whom Peter gives the book.
(c) The user wants to be in a situation where
(*) both the user and the system have the same opinion about whom Peter gives the book to, and believe that (*)
(d) The user believes that the system believes that (a) – (d).

The infinite loops in the above definition describe specific belief constellations which are fundamental in dialog settings, namely *mutual beliefs* (or more precisely, *infinite-reflexive beliefs* – cf. [KOBS84]). All of the above assumptions can be formed simply because the user has entered a wh-question, without regard to the content of his/her question.

Other purely structural rules of inference (which were used, for example, in [ALLE80, CARB83, CARB*, SIDN83b, MCKE85c]) include

(4) If the user wants to know how to achieve an effect, then his/her goal may be to achieve that effect.

(5) If the user wants to know whether p, then his/her goal may be that p, or not p.

Other forms of inference depend on the *meaning* of a user's input to the system, and on a large amount of universal knowledge about the domain of discourse. From (3), for instance, the system could also infer the following:

(3)(e) Assumption of the system: The user believes that the book is now in the possession of the person whose identity is unknown to him/her.

This inference is based upon the content of the user's question, upon world knowledge about the meaning of 'to give' (namely that it denotes an action, the result of which being the possession of an object by the recipient), and the general inference rule

(6) If the user believes that an action occurred, then s/he also believes that the effects of the action occurred.

[KASS87a] proposes a variety of such general rules for acquiring additional assumptions about the user. Examples include:

Action rule: If the user model includes the belief that a user is familiar with an action, then the user modeling module can attribute to the user knowledge of the preconditions and effects of that action.

Concept generalization rule: If the user model indicates that the user knows several concepts that are specializations of a common, more general concept in the domain model, the user modeling module may conclude that the user knows the more general concept, and the subsumption relationship between the more general concept and the more specialized concept as well.

Inheritance rule: If the user believes a concept A has property P, and further believes that A subsumes concept B, then the user believes B has property P.

Assumptions from the user's input can also be used for inferring his/her possible plans. The systems of Allen & Perrault [ALLE80], Carberry [CARB83, CARB*] and McKeown et al. [MCKE85c] have a large number of domain-dependent precondition-effect sequences (i.e., operators) stored in their knowledge bases. They can then use inference rules such as

(7) If the user's goal is to achieve certain effects, then s/he will probably use a sequence of operators which yield this effect and whose preconditions are fulfilled

for inferring probable user plans. In all these models, however, the system and the user hold identical beliefs about these action operators (e.g. about their preconditions). This restriction is relaxed in the work of Pollack [POLL86a, POLL86b].

In contrast to the other systems, however, Pollack's model does not allow for the chaining of action operators.

Another type of domain-dependent inference is that by which, on the basis of "objective" facts about the user, assumptions are formed about his/her probable traits, in particular his/her preferences, requirements and criteria of assessment in a certain domain. Rich's [RICH79a, RICH79b] GRUNDY system, for instance, forms assumptions about the user's selection criteria for books to read from information about personality characteristics supplied by him/her. Morik & Rollinger [MORI85b] form assumptions about the user's assessment criteria for apartments from information about his/her personal situation, such as the number of children, the acceptable price range, etc. If a great number of inferences are drawn from an antecedent (as is in the purest form the case in GRUNDY), then the set of their consequents may be regarded as a *stereotype* (see Section 4.2 and [RICH*]).

Furthermore, the interpretation of *modal verbs* like 'must, should, need, may' in the user's input makes it possible to infer some of his/her wants and beliefs. Unfortunately, modals are ambiguous in most languages. For example, in German most modals can be used for expressing a want as well as a belief. Sprenger [SPRE88] has developed a computational model for disambiguating modals, so that they can be mapped onto modal operators in the semantic representation language of a user model which is integrated in the consultation system WISBER [SPRE87]. There are many technical problems relating to how a modal verb expresses a belief or goal. Even the correct interpretation of a modal like 'want' in (2) can be quite complex in other contexts. In (8),

(8) User: I want to buy a Porsche, because I would like to have a very fast car.

the user expresses the *belief* that buying a Porsche results in having a very fast car and the *goal* of having a very fast car. In (9), the clause

(9) User: I wanted to buy a BMW, but then I heard that a Porsche is even faster.

beginning with 'but' should cause the user modeling component to assume that the user now wants to buy a Porsche.

Additional assumptions about the user can be formed from *linguistic particles* in his/her input, as in

(10) User (to a text editor system): *Why* is this line *not* centered?
Assumption of the system: The user wants the line to be centered.

(11) User: I do *not* have *enough* disk space.
Assumption of the system: The user wants more disk space.

Up to now, the analysis of linguistic particles in the user's input has mostly been investigated on a theoretical basis [KOBS85a]. Such an analysis would offer a simple way of obtaining assumptions about the user. It seems, however, that interesting particles are not very frequently used in goal-oriented dialogs. Moreover, in contrast to the techniques discussed above, this method is also only applicable in natural-language dialog systems.

2.4. Assumptions Based on Dialog Contributions Made by the System

Dialog contributions of the system (i.e. answers, questions and commands) also lead to new entries in the user model. If the system had previously planned the dialog contribution by trying to anticipate its probable effect on the user, it can now hypothetically enter these effects into the user model (e.g. after answering a user's question the system can assume that the user now knows the content of the answer). These entries may give rise to the planning of additional unsolicited comments (cf. Section 4.3.3 and [JAME*]) or follow-up information and explanation.

2.5. Assumptions Based on Non-Linguistic Input of the User

In Section 2.3, a number of techniques were discussed for forming assumptions about the user based on his/her input into the system. In contrast to a face-to-face conversation between humans, present dialog systems are restricted to deriving such assumptions solely from the strings typed in by the user; they cannot access any visual, acoustic or other external clues.

By expanding Norman's futuristic idea of a personal user interface on a pocket-sized "smart" card, which, when plugged into any terminal, makes it emulate the one at the user's home or office (cf. [JOYC83]), one could imagine a slight deviation from that restriction, namely, that a system's user model is initialized by a kind of electronic identity card storing some of its owner's personality characteristics. The information on this card may be regarded as an initial individual user model in the sense of Section 2.2. The information thereon would be more general, however, and not restricted to the specific domain of a particular system.

At the beginning of an interaction with a dialog system, the use of such a device containing a user-provided individual profile seems no more unnatural than responding to a system's request like

(12) I'd like to know what sort of person you think you are. Please type in a few single words that you think characterize the most important aspects of yourself

as is required in Rich's [RICH79a, RICH79b] GRUNDY system. Of course, the problem remains as to whether the conveniences offered by such personality cards are not far outweighed by the dangers they give rise to in respect to loss of privacy and easier surveillance.

3. Representing User Models

In the field of user modeling, a wide range of formal means are employed for representing the assumptions of the dialog system regarding the user. What representation one opts for is usually determined by the application purpose of the system and by the demands imposed on the expressive power of the user model.

Rich's GRUNDY system, for instance, makes extensive use of simple *linear parameters.* For every personality trait in her system, one parameter indicates the assumed degree to which the personality trait characterizes the user, and another indicates the degree to which the system believes that its assumption about this specific personality trait is correct. An example of such personality traits is given in Figure 2.

Personality trait	Value	Certainty of assumption
Education	5	900
Seriousness	5	800
Piety	-3	423
Tolerate-sex	5	700
Tolerate-violence	-5	597
Tolerate-suffering	-5	597

Figure 2. Representation in the GRUNDY system

Sleeman's UMFE system [SLEE85] also uses very simple methods for representing its assumptions about the expertise of a user in a given domain. UMFE is a front end system for the explanation component of expert systems. It takes the inference chains yielded by such components and adapts them to the specific level of expertise of the current user. This is done by making sure that only those concepts are used in an explanation which the user has explicitly indicated familiarity with, or which the system has assumed him to know.

For representing the concepts that a user is probably familiar with, a sort of *overlay technique* is employed in Sleeman's model (also see [KASS*]). A simple parameter associated with the representation of each individual concept in the system's knowledge base indicates whether, according to the system's assumptions, the concept is KNOWN or NOT-KNOWN by the user, or whether the system has NO-INFORMATION in that respect. Another factor associated with each concept indicates the strength of belief with which the system holds its assumptions about the user's familiarity with these concepts. Currently, however, no use is yet made of these factors. An example of a simple overlay model (which is inspired by the UMFE system) is given in Figure 3.

As soon as the user model is to give a slightly more detailed account of what the user knows or does not know, of his/her misconceptions, or of his/her goals and plans, etc., the linear scale values must be replaced by representational structures with greater expressive power. In natural-language dialog systems, for representing knowledge about the *world*, representational schemes developed in the field of knowledge representation are employed. Such schemes are, for instance, formulas of first-order predicate calculus, network-like representational schemes, frames, etc. It thus seems most natural to extend these schemes so that the system's assumptions about the beliefs, goals and plans of the user can be represented.

Unfortunately, a simple extension is not possible. When using predicate calculus, one has to decide by what epistemological primitives notions like 'believe',

Concept hierarchy of the system	User model	
	user's knowledge state	certainty of assumption
INFECTIOUS-PROCESS	KNOWN	100
HEAD-INFECTION	KNOWN	100
SUBDURAL-INFECTION	NOT-KNOWN	100
OTITIS-MEDIA	NO-INFORMATION	100
SINUSITIS	NO-INFORMATION	100
MENINGITIS	KNOWN	100
BACTERIAL-MENINGITIS	KNOWN	90
MYCOBACTERIUM-TB	NOT-KNOWN	70
VIRUS	NOT-KNOWN	90
FUNGAL-MENINGITIS	NOT-KNOWN	100
MYCOTIC-INFECTION	NOT-KNOWN	100
ENCEPHALITIS	NOT-KNOWN	90
SKIN-INFECTION	KNOWN	100

Figure 3. An example of an overlay model

'want', etc. should be represented. Opting to regard them as sentential operators (as are, for example $\vee$, $\wedge$, $\sim$ or $\supset$) causes serious problems, since these operators would not be extensional (an operator is extensional [or truth-functional] if the truth value of a complex formula constructed by it depends solely on the truth value, and not on the "meaning", of its constituent formulas). The lack of this property causes serious problems because fundamental principles of predicate logic (as, for example, substitution of terms with same extension, existential generalization) are no longer applicable.

Another possibility is to regard these notions as meta-language predicates which are applied to meta-language constants and variables that denote object-language formulas. These formulas, in return, express the content of the beliefs and goals. The system's assumption that the user believes that Peter loves Mary thus would be represented in this approach by something like 'BEL(USER,P)', where 'BEL' is a meta-language predicate, 'USER' a meta-language constant that denotes the user, and 'P' a meta-language constant that denotes an object-language formula like 'loves(peter,mary)'. Konolige [KONO81] developed a logic for belief representation that is based on this idea. For more complex belief constellations, however, such representations become computationally untenable. Moreover, as has been shown in [KOBS85a], a number of beliefs that are fundamental for dialog planning based on a user model cannot be represented in Konolige's scheme.

Another approach is to regard notions like 'believe' and 'want' as modal operators, and to use one of the many modal calculi for their axiomatization. This idea was first developed in [HINT62]. Moore [MOOR80] and Appelt [APPE85] investigated a belief and goal representation which formalizes the standard semantics of modal logic, namely *possible-world semantics.* In their representation, modal formulas are immediately translated into possible-world sentences, to which the

standard deduction techniques can be applied. The modal operators 'believe' and 'want' are thereby translated into accessibility relations between possible worlds. An agent (e.g. the user) knows some fact p if p is true in all worlds which are compatible with what s/he knows.

The system's assumption that Peter loves Mary thus would be represented in this approach by a formula like

(13) $\forall W$ [K(USER,W0,W) ⊃ T(W,P)]

'W' thereby is a meta-language variable ranging over possible worlds, 'W0' a meta-language constant denoting the world in which the user currently finds himself, and 'P' a meta-language constant denoting some object-language formula like 'loves (peter, mary)'.

'K (USER, W0, W)' is true iff W is accessible from W0 according to what the user knows. 'T(W, P)' is true iff the formula denoted by P is true in W. Such a situation is depicted in Figure 4. Here the user knows the fact expressed by 'loves(peter, mary)' but is uncertain with respect to the fact expressed by 'loves(mary, john)', since there is a world which is accessible to him according to what he knows in which the negation of the latter formula is true.

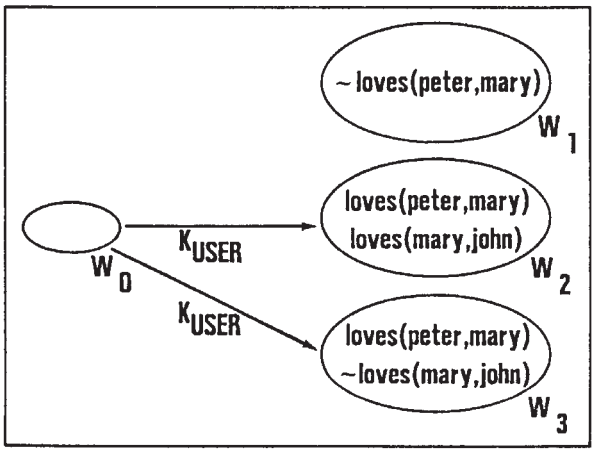


Figure 4. A possible-world representation of 'The user knows that Peter loves Mary and is uncertain whether Mary loves John'

Possible-world semantics has recently been intensively investigated as a means for belief representation [HALP85a, FAGI85, VARD85]. However, as was pointed out in [KOBS85a], as far as the representational needs in the field of user modeling are concerned, the expressive power of this approach is somewhat limited.

Another approach to belief and goal representation is the "*partition approach*", which is applicable to all forms of concept-based knowledge representations. It was first used by Cohen [COHE78] for his special network-oriented representation. The idea behind this approach is to maintain a number of separate partitions to store the system's beliefs about the domain, the system's goals, the system's assumptions about the user's beliefs about the domain, the system's assumptions about the user's goals, the system's assumptions about the user's assumptions about the system's beliefs about the domain, etc. For representing the contents of these beliefs and goals, the standard concept-based knowledge representation schemes (e.g. network-like

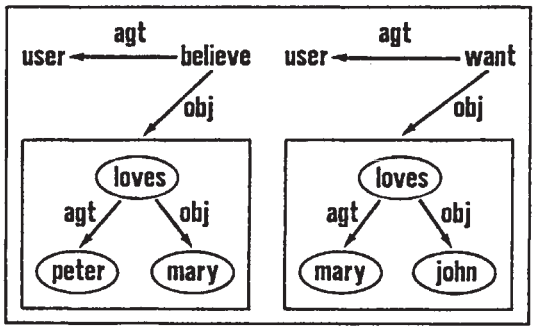


Figure 5. A partition representation of 'The user believes that Peter loves Mary and wants Mary to love John'

representations, predicate calculus) may then be employed. An example of such a representation with a partition for the system's beliefs and another for the system's assumptions about the user's goals is given in Figure 5.

Unfortunately, in the form described here, this approach raises some problems which were spelled out, for example, in [MOOR80, SOUL84, KOBS85a]. In [KOBS85d], a new interpretation is therefore given for this sort of belief and want representation, and new epistemological primitives (the so-called '*acceptance attitudes*') are added to increase its expressive power for more subtle belief and goal types. In the BGP-MS system [KOBS88, ALLG89], this representation scheme is currently enhanced by *partition inheritance* which, for instance, allows the user's and system's mutual beliefs to be stored in a separate partition, the contents of which are inherited by the partitions containing the system's beliefs about the domain and the system's assumptions about the user's beliefs about the domain. Kass [KASS87a] plans to extend this representation in such a way that acceptance attitudes can also be applied to rules, and that beliefs also carry justifications for why the belief is held or not held.

4. Exploiting User Models

4.1. Exploitation Outside of the System

In most systems, user models are applied as aids which help the system in pursuing a particular set of dialog goals or in behaving cooperatively when performing a task together with the user. However, there are conversational settings where the construction of a user model is itself the *central* dialog goal (as is the case with, for example, exams or personnel interviews) and not just a means to an end (in these cases, the exploitation of the user model is left to human interpreters).

An early proposal for such a distinguished function of a user model in an AI system was the NL dialog system ULLY (cf. [HAYE76]). The conversational setting for ULLY was a fictitious cocktail party at which both the user and the program were "guests". The main goal of ULLY was to find out as much as possible about the user. ULLY was designed "to be a friendly and reasonably co-operative conversational partner, who is politely nosey" (cf. [HAYE76], p.138).

The authors assumed that at the beginning of the conversation, ULLY has a certain amount of knowledge about the other party guests, such as their names, what they do, how they are related to each other (e.g. X detests Y), but does not know anything about the user. At the end of the session, ULLY should have a more or less refined idea of what the user is like.

ULLY has never been implemented, since – we think – some of the project's goals were overambitious considering the primitive state of the art of user modeling at that time. Although we have seen some advances in user modeling during the last ten years, even today some of ULLY's maxims point to areas open to further research, for example

- try to increase the level of intimacy without being impolite.
- try to keep the conversation at a level which is understandable.
- treat assertions by the user with "care".

4.2. The Relevance of User Modeling in Various Applications of Natural-Language Dialog Systems

If a system is designed as a cooperative consultant who actively recommends X (e.g. a book in GRUNDY or a hotel room in HAM-ANS) or advises the user against X, a model of the user is a necessary prerequisite for performing the corresponding speech acts (cf. [MORI84, MORI*]). A dialog system which generates only fixed and biased recommendations, on the other hand (e.g. a version of GRUNDY which always suggests the Bible first) has hardly any need for a user model. The following types of conversational behavior relevant to AI applications are listed according to an increasing need for a user model:

(A) question-answering and biased consultation
(B) cooperative question-answering
(C) cooperative consultation
(D) biased consultation pretending objectivity.

A system that responds to the literal meaning of simple questions about a flight schedule or a hotel reservation system which always offers the most expensive room first are examples of category (A). Such systems must possess limited user models only, e.g. for handling discourse phenomena like anaphoric reference and ellipsis appropriately. While all systems of this category implemented to date are simple question-answering systems, it is highly probable that biased systems will be installed as soon as the technology of NL processing becomes further commercialized. The focus of current research is shifting from systems of category (B) (e.g. CO-OP, cf. [KAPS79, KAPS82, KAPS83]) to category (C) (e.g. ARGOT, cf. [ALLE82a, ALLE83b]; UC, cf. [WILE84, WILE86]; XTRA, cf. [ALLG89, COHE*]). Systems of both of these categories include user modeling techniques for the generation of various kinds of over-answering (cf. [WAHL83]) and for the generation and recognition of (indirect) speech acts (cf. [ALLE79, PERR80, APPE82b, APPE85]). Systems

of category (D), which mimic for example professional salesmanship, require a very sophisticated form of user modeling. The experimental dialog system IMP, which will be discussed below, includes some aspects of that kind of dialog behavior.

As was pointed out by Rich [RICH83], user modeling is especially important if the dialog system is used by a heterogeneous group of users and if the system shows some flexibility in what it tells the user. A typical example of this is her GRUNDY system which, in playing the part of a librarian, can select from a large number of books one that it assumes the user might like to read. The selection is made on the basis of a model of the particular user which the system constructs in the course of the dialog.

Another example of a dialog system for a heterogeneous user group is HAM-ANS which, in its hotel reservation application, plays the part of a hotel manager who tries to rent all rooms available in his/her hotel. On the basis of an explicit model of the individual user, the system can select one of four room categories and suggest a vacant room in the selected category. In this conversational setting the user is assumed to have the overall goal of getting a hotel room which meets his/her requirements.

From a greeting like that in Figure 6 (which includes the name and optionally the title, the affiliation and the home or office address of the user) and from further booking information like that provided by the user in Figure 6, the system can form some assumptions about the prospective hotel guest, e.g. his profession, the purpose of his trip (business trip, vacation, etc.) and his solvency. Several stereotypes are associated with each of these dimensions.

As in GRUNDY, stereotypes are used as collections of frequently recurring characteristics of users. In the example above, HAM-ANS may assume that Mayer is Mueller's secretary and that Mueller is a manager in a high-tech company ("Sunshine Computers"), since he holds a Ph.D. and has a secretary. Furthermore, the system assumes that Mueller is on a business trip and, because his expenses are reimbursed, he is able to spend a lot of money for a room. The user-provided information relevant for the consulting procedure is stored as factual knowledge in the user model, whereas the stereotypes triggered by these facts are stored as hypotheses, which can be withdrawn if they are disconfirmed by the subsequent dialog. Note incidentally that if the caller is booking a room for someone else, the model discussed here does not contain information about the actual user of the system, but about the prospective hotel guest instead. Hence, according to the terminology introduced, this is a case of agent modeling in general, and not specifically user modeling.[4] One can imagine future dialog systems which also model the intermediary persons (e.g. for also taking into account private interests of secretaries), but this is beyond the scope of our discussion.

HAM-ANS can characterize a particular hotel guest by several distinct stereotypes. For example, its assumptions about a user named Schmidt may be that he is a university professor who wants to spend his vacation with his family, and that he is not able to spend much money because he has five children. As soon as the system

[4] Sparck Jones [SPAR84, SPAR*] calls the person being talked about in the discourse the *patient*, and the person interacting with the system (i.e. the user in our terminology) the *agent*.

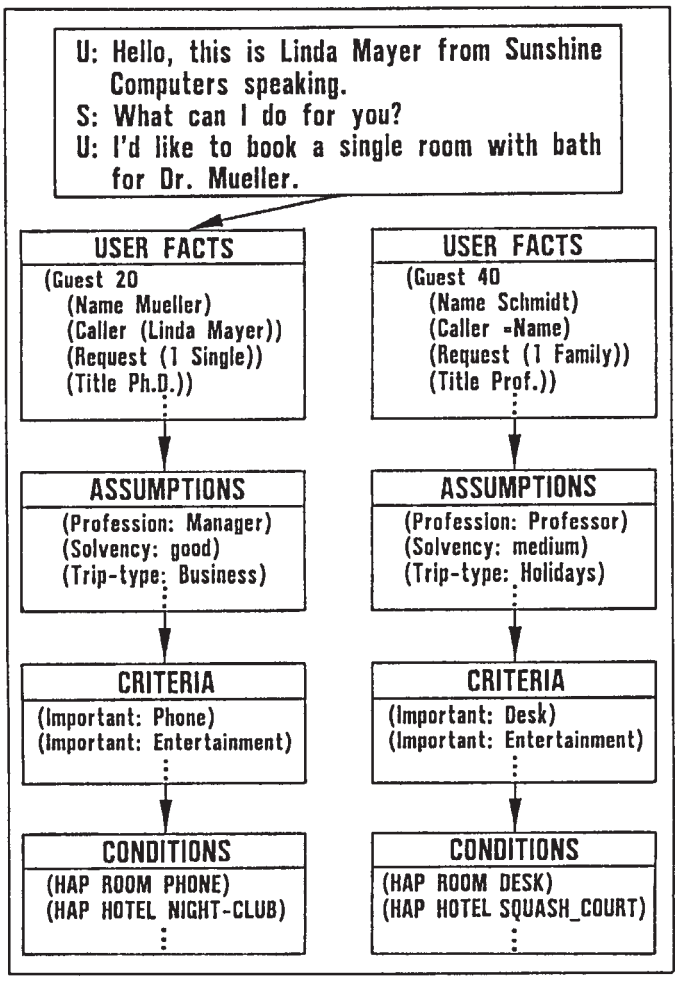


Figure 6. The use of stereotypes in HAM-ANS

has all necessary booking facts at its disposal, it checks whether there is a vacant room which meets the requirements explicitly stated by the user (e.g. period of stay, number of beds). If several rooms belonging to different categories meet these explicit requirements, the system uses the activated stereotypes to predict further demands of the user. For example, HAM-ANS may assume that for Mueller, as a manager on a business trip, a room telephone is very important. Prof. Schmidt on the other hand, being a scientist, actually needs a large desk even on his vacation, but in this particular situation a room telephone is not so important for him. These stereotypical inferences generate a list of criteria rated by different importance values. The set of derived criteria is matched against the available room categories in order to generate a strong or weak recommendation and to indicate which criteria are fulfilled and which are not. However, since individuals differ in the conditions they impose for the fulfillment of a criterion, this matching process again depends on the user profile (cf. [MORI85a,MORI85b]). For example, if the guest plans to stay for a longer period in the hotel, attractive entertainment facilities become very important. This criterion, of course, is evaluated differently by a businessman who travels by himself compared with a professor who spends his vacation together with his family.

4.3. Anticipation Feedback

A special case of exploitation of a user model (which is realized in dialog systems like HAM-ANS, NAOS and IMP) is its use in various forms of *anticipation feedback loops.* Anticipation feedback loops involve the use of the system's language analysis and interpretation components to simulate the user's interpretation of an utterance which the system plans to verbalize. The application of anticipation feedback loops is based on the implicit assumption that the system's procedures for language analysis and interpretation (but not necessarily the contents of the sources of knowledge used by these procedures) are similar to those of the user. Such a similarity assumption seems to be quite common among participants in everyday conversations. Since most people (except, of course, some computational linguists and AI researchers) cannot be assumed to have an exact model of the comprehension capacity of their dialog partners (except for some naive assumptions concerning small children and non-native speakers) it is highly plausible that they use the similarity assumption as a default rule. Although the current state of the art in NL dialog systems in no way provides sufficient evidence for assuming a far-reaching similarity, user modeling based on anticipation feedback loops can considerably improve the quality of present dialog systems.

4.3.1. Local Anticipation Feedback for Ellipsis Generation

The ellipsis generation component of the dialog system HAM-ANS incorporates a local anticipation feedback loop which helps to ensure that the system's utterances are not so brief as to be ambiguous or misleading (cf. [JAME82]). Consider for example the user's question (14).

(14) User: Do five colleagues have more than three weeks vacation?

If five colleagues have in fact not more than three, but more than two weeks vacation or if only two colleagues have more than three weeks vacation, a cooperative system might respond with (15), instead of simply answering 'No'.

(15) System: No, two.

By using its ellipsis resolution component in an anticipation feedback loop, the system can recognize that (15) would be a confusing response since it is so elliptical that, without further context, no unambiguous interpretation can be assigned to it. Acceptable responses like (16) and (17) include more overlap with question (14).

(16) System: No, two weeks.

(17) System: No, two colleagues.

Having rejected (15), the system internally generates the less elliptical responses (16) or (17), depending on the intended meaning. This time, the anticipation of the user's understanding causes a positive feedback, so that the elliptical structure can now be mapped onto a well-formed NL utterance by the surface transformation component of the system (for a more complete account of HAM-ANS's user modeling in ellipsis generation cf. [JAME82]).

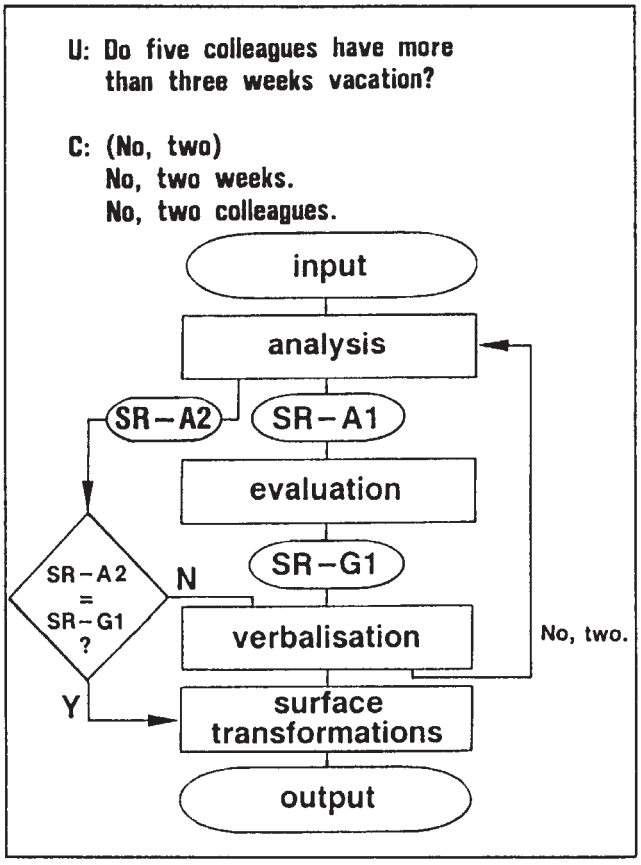


Figure 7. A simple anticipation feedback loop

Figure 7 shows an extremely simplified version of the general structure of a dialog system with such an anticipation feedback loop for user modeling. The parsing of an input results in the semantic representation construction SR-A1, whose evaluation yields SR-G1 as the semantic representation of the intended response. The result of a preliminary verbalization process for SR-G1 (which includes the above-mentioned ellipsis generation) is fed back into the system's analysis component, which is equipped with an ellipsis resolution component. The semantic representation construction SR-A2, which forms the result of the ellipsis resolution process (and which may consist of a disjunction if the input is ambiguous) is then compared with the intended meaning of the response, namely SR-G1. If SR-A2 matches SR-G1, the structure fed into the analysis component is transformed and finally outputted, otherwise an alternative verbalization of SR-G1 is checked in the next iteration of the feedback process.

The anticipation process discussed above is based on a local feedback loop, i.e. one in which the generation procedures of one particular aspect of an utterance are linked with the recognition procedures for the same aspect.

A variety of phenomena, such as the generation of definite descriptions, pronouns, deictic expressions and noun-noun modifications can be handled in a dialog system either by a corresponding multiplicity of local feedback loops (cf. [WAHL83, MARB83]) or by a larger one within which the whole generated utterance would be tested once, just before applying the final surface transformations (see Figure 8).

Such global anticipation feedback loops require dependency-directed backtracking, so that the system returns to the adequate decision point in the generation process whenever the result of the feedback process is negative.

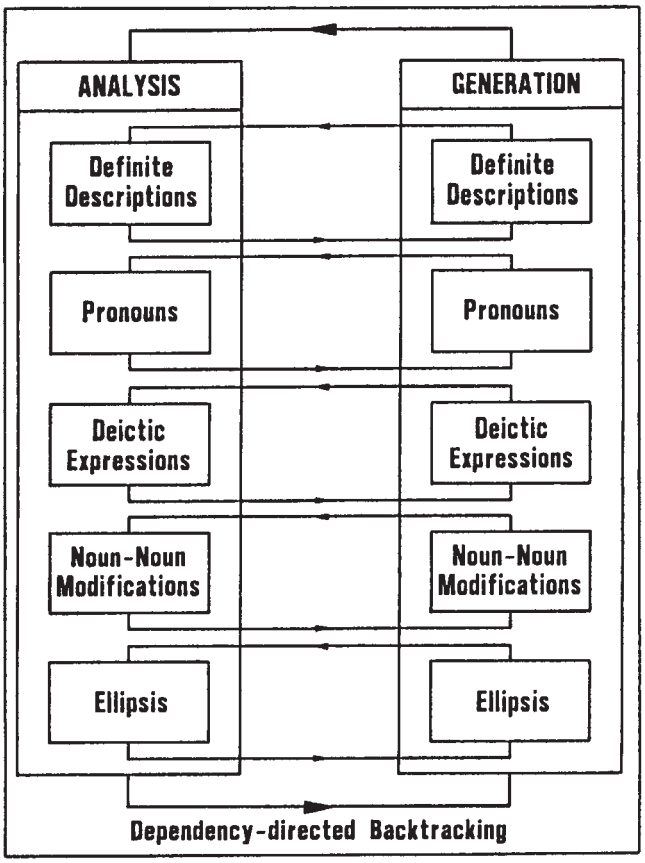


Figure 8. Local and global anticipation feedback loops

4.3.2. A Global Anticipation Feedback Loop

An example of a more global anticipation feedback loop is the imagination component of the system NAOS, which generates NL descriptions of real-world image sequences. [NOVA84] describes the design of a user model for NAOS, which contains a representation of the imagined scene resulting from an anticipation of the user's understanding of a planned utterance. Such a system would take into account the present state of the dialog and the assumed knowledge and dialog goals of the user. On this basis, the representation of the imagined scene (which always contains some uncertainty) and the actual scene observed by the system must be matched in order to check whether the planned utterance will be interpreted by the user in a consistent manner and whether it will generate the intended perceptual impression in adequate detail. For instance, it is quite clear that the image sequence of a car accident in which one was involved is described differently to the police, the lawyer of the opposing party, or a good friend.

This application shows that a user model cannot completely rely on representation schemes only suited for the representation of discrete domains. Instead, they

must also provide means for the representation of continuous domains, e.g. of the fuzzy regions of an object's trajectory which might form part of the image anticipated by the speaker for a planned utterance like (18).

(18) A red BMW passed a small VW.

4.3.3. Anticipation Feedback in Evaluation-Oriented Discourse

Jameson [JAME83, JAME*] has studied a very advanced form of anticipation feedback for user modeling in his IMP system. IMP was designed to answer the questions of a user whose sole dialog goal was assumed to be the assessment of some object. On the basis of explicit assumptions about the user's standards and prior expectations, IMP goes beyond the literal answering of questions by adding unsolicited comments that influence the evaluation of the object under discussion. In evaluation-oriented dialogs (e.g. personnel selection interviews, dialogs with travel agents, hotel managers or other salespeople) the selection of such comments is influenced by the nature of the speaker's bias (e.g. a desire to present something in a favorable light). Since listeners take these facts into account when interpreting such comments, IMP anticipates the interpretation of the user while taking into account the bias which the user possibly ascribes to it. IMP's actual bias can be positive, negative or objective, and it can attempt to maintain a positive, negative or objective image. The actual and projected bias may differ (e.g. usually a salesman's actual bias is positive though he attempts to maintain a neutral image).

On the representational level, IMP's user model includes an explicit record of the user's expectations concerning a particular room offered for rent. This record is updated according to the user's anticipated interpretation of the system's answer. The most interesting procedural aspect of IMP's user modeling is an anticipation feedback loop which takes into account the effect of a planned utterance on the user's impression of the room and helps to filter out comments which would betray the system's bias.

IMP's user model is also used to select expressions like 'but', 'by the way' and 'unfortunately', which precede the additional comments generated by the system. An example is given in (19):

(19) User: What about facilities outside the room?
System: There's a kitchen and there's a bathtub and by the way, the room is very large.
User: Is the room on a quiet street?
System: No, in fact unfortunately there's a lot of noise from the street.
([JAME83], p.618)

For selecting such connectives and sentential adverbs the system compares the shift in the expected value produced by the present comment with that produced by the preceding statement. IMP shows that for an explanation of the complex semantics and pragmatics of such expressions a user model is necessary.

As we have shown, the use of an anticipation feedback loop is an important part of the process of designing a successful utterance. In essence, anticipation on the part of the speaker means answering question (20) (cf. [ROSN81]).

(20) If *I* heard this utterance relative to a set of beliefs B and goals G, where B and G are what *I* assume the hearer believes and wants, respectively, then what would be the effect on me?

It is obvious that a dual question like (21) is useful for the hearer's analysis of utterances.

(21) If *I* had made this statement, on the basis of what beliefs and goals would I have said it?

4.4. Correcting User Misconceptions

In the last few years, a number of techniques have been developed for recognizing misconceptions underlying unsuitable dialog contributions of the user, and for responding to these dialog contributions in an appropriate way (for a brief survey, see [WEBB83]). Each of these methods is restricted to a very particular type of user misconception. The work of Kaplan [KAPS79, KAPS82, KAPS83], for instance, is concerned with the detection of false presuppositions in users' database queries. These include merely "syntactic" presuppositions regarding the existence of the objects mentioned in the discourse, such as

(22) How many female employees in division 3 are managers?
(23) Presupposition: there are female employees in division 3.

Kaplan also considers "indirect" existential presuppositions which can only be inferred using domain knowledge, such as in

(24) User: Which students got a grade of F in CS105 in Spring 1980?
(25) System: CS105 was not given in Spring 1980.

[MAYS80b] is concerned with the recognition of violations of argument restrictions presupposed in users' questions, such as in

(26) User: Which undergraduates teach courses?
(27) System: Undergraduates don't teach courses. Only the faculty teaches courses.

A similar mechanism for detecting "intensional" errors has been implemented in HAM-ANS in its application as an access system to a relational database containing information about research voyages to the antarctic sea [MARB87, MARB88]. The user is informed if relations whose existence is implied by the user's query are not modeled in the database.

All these techniques for detecting and responding to user misconceptions are *not* based on an explicitly represented user model (though an "implicit" user model is hidden in the operation of the mechanisms which implement these techniques). Such an approach is fully justified in limited domains. In more flexible systems,

however, it will become necessary that the assumptions about the user inferred by such techniques be explicitly represented in the user model (e.g. for using them in the determination of the user's level of expertise) and that the handling of user misconceptions be performed on the basis of such an explicit model. Moreover, it can be expected that most types of user misconceptions not described above can only be adequately dealt with on the basis of an explicit user model.

A good example of work along these lines is McCoy's ROMPER system [MCCO85a, MCCO85b, MCCO*] which corrects users' object misclassification (i.e. their subsumption under a wrong superconcept in a concept hierarchy) and the attribution to objects of properties which they do not actually have. Response patterns exist for each type of misconception which enable the system to correct the user. An example of a response to an object misclassification looks as follows:

(28) User: I thought that whales were fish.

(29) System: No, they are mammals. You may have thought they were fish because they are fin-bearing and live in the water. However, they are mammals since, (while fish have gills) whales breathe through lungs and feed their young with milk.

Another example of misconception handling on the basis of a user model is Quilici's [QUIL*] AQUA system. AQUA recognizes user misconceptions about UNIX inferable from his/her reports on problems which s/he has had in interaction with UNIX, and selects appropriate responses. A misconception is discovered by checking each recognized user belief about UNIX against the system's expert knowledge. If the incorrect belief is detected in a catalogue of frequent misconceptions held by the system, the user can be immediately corrected. Otherwise, the type of misconception is determined and associated explanation patterns tested as to whether their preconditions match the recognized user and system belief situation. When an explanation matches completely, the user can be informed of the recognized misconception such as in

(30) User: So to remove a file, I have to own the directory that contains it.

(31) System: No, to remove a file, you need to have write permission on the directory that contains it. You do not need to own the directory that contains it. You need to own that directory when you do not already have write permission on it.

4.5. Some Social Problems in Exploiting User Models

If the user is *aware* that the system with which s/he is interacting is equipped with a user modeling component, then s/he has the chance to apply all the techniques for hiding his/her beliefs, goals and plans which s/he constantly uses in every-day person-to-person interaction. This includes, for example, pretending that one knows less or more about a domain, hiding one's goals and plans, beating about the bush, etc. In some applications (e.g. tutoring or unsolicited advice) the user is necessarily aware of the existence of a modeling component; in other applications its existence might not be so apparent to him/her.

Control of the user over the user modeling component means, for example, that the user can inspect the model which the system has made of him/her in the course of a dialog. However, since the number of assumptions made by the system in a normal interaction can be expected to be enormous (cf. Section 2.3.), is it not possible for the user to inspect them all. Another problem is rendering these assumptions comprehensible to him/her. The representational categories employed for user modeling often do not correspond to the usual "folk-psychological" belief and goal categories. Thus the contents of user models are often very difficult to translate into ordinary English.

Control of the user over the user modeling component may also mean that the user is entitled to alter the assumptions that s/he has inspected. In this case, as well, the problem arises of how to explain the consequences of such alterations to the user. One might argue that such an option might seduce him/her to change actually correct assumptions in order to make them correspond to his/her (more positive) self concept. In the case of cooperative systems and computer-based tutoring, however, it is the user's problem if the system's cooperativeness declines as a result of these manipulations of the user model.

Another interpretation of a user's control over a system's user modeling component is that the user has the possibility to "switch off" this component and to communicate with the remaining "unintelligent" system alone, if s/he does not consent to being modeled by the system. In some cases (e.g. tutoring), this is not possible due to the nature of the application. In other cases (e.g. cooperative information), it is certainly not advisable to do so. However, there are also applications (e.g. unsolicited advice) where such a demand on the part of the user may be justified. The technical question remains, however, whether it is possible to separate the user modeling component from the remaining system in such a way that it is possible to "switch it off".

User modeling therefore involves the risk of *misunderstandings* (as is also the case in normal human communication). It can be expected, however, that the gain in cooperativity will greatly exceed that risk. Often it is only this cooperativity which enables the casual user to gain access to the system at all. In traditional dialog systems, by the way, there is also the problem of misconception, however only on the part of the user (e.g. about the meaning of some command or some technical term, which, when undetected, may lead to a wrong interpretation of the results.) In any case, a necessary characteristic of a user model must be that the assumptions stored in it be revisable if counterevidence is observed (cf. [DOYL80]). As we mentioned already in our introductory definitions (see Section 1.2) it is one of the tasks of a user modeling component to maintain the consistency of the model. Therefore a truth maintenance system (TMS, also known as belief revision or reason maintenance system, cf. [DOYL79, DOYL83]) is an important subpart of a user modeling component. Actually, the TMS is a crucial factor in the effectiveness of recent user modeling shells like GUMS (cf. [FINI86, FINI*]) and TRUMP (a TMS- and Rule-based User Modeling Prototype, cf. [BONA87]).

5. Open Questions and Future Research in User Modeling

5.1. Coping with Collective Beliefs

In the research discussed so far, user modeling was restricted to models of individuals. However, as Wilks & Bien [WILK83] have pointed out, it is necessary in some dialog situations to model the beliefs of organizations (e.g. NSF's view of AI research), of states (e.g. Germany's view of the European Community), or of classes of individuals (e.g. the sales division's view of AI). Sometimes in a conversation, an individual (e.g. a sales representative, a district attorney, a diplomat) has to advocate the beliefs and goals of such a collective third party without referring to his/her personal beliefs or goals. Thus it may be reasonable for a cooperative NL interface to a commercial DBMS to restrict the dialog-independent user model, for example to the point of view of the sales division, the research division, or the production division of a company (the view mechanism in database technology may be seen as a primitive precursor of such a group-oriented user modeling scheme). Based on the assumption that an individual user inherits all beliefs and goals from the division s/he is working for, a simple user model may in such a professional environment ignore the "private" beliefs of the dialog partner, and model only the collective ones.

However, for most conversational settings, this is an oversimplified approach since often in the course of the dialog "private" beliefs and "inherited" beliefs are mixed (see Figure 9). For example, a sales representative may begin a conversation with a client by referring to a "private" belief like (32), but later on express the beliefs and goals of his employer, as in (33).

(32) Last time we met we had an excellent dinner together.
(33) This product is going to be a big seller.

Here the open question is how the system can decide which beliefs are to be regarded as "inherited", and therefore to be passed on to that part of the knowledge base which is reserved for the representation of collective beliefs, and which ones are to be stored in the individual user model. Sometimes even a human listener has difficulty coming to a decision and therefore must ask a meta-question like (34).

(34) Is this your personal opinion or an official statement of your company?

Although our current techniques of plan recognition and speech act interpretation must be greatly refined before we can cope with this problem, a future solution may have great practical importance in certain domains, e.g., in consultative and argumentative dialogs and in knowledge acquisition dialogs with expert systems.

Another open question related to the previous one is: How can the system construct, maintain, and update consistent permanent models of collective beliefs on the basis of dialog sessions with individual users? At least for the sake of storage economy, it is necessary for a dialog system with heterogeneous user groups to identify consistent sets of beliefs and construct inheritance networks of beliefs. A solution to this problem must rely on techniques for machine learning, especially inductive inference and stereotype formation.

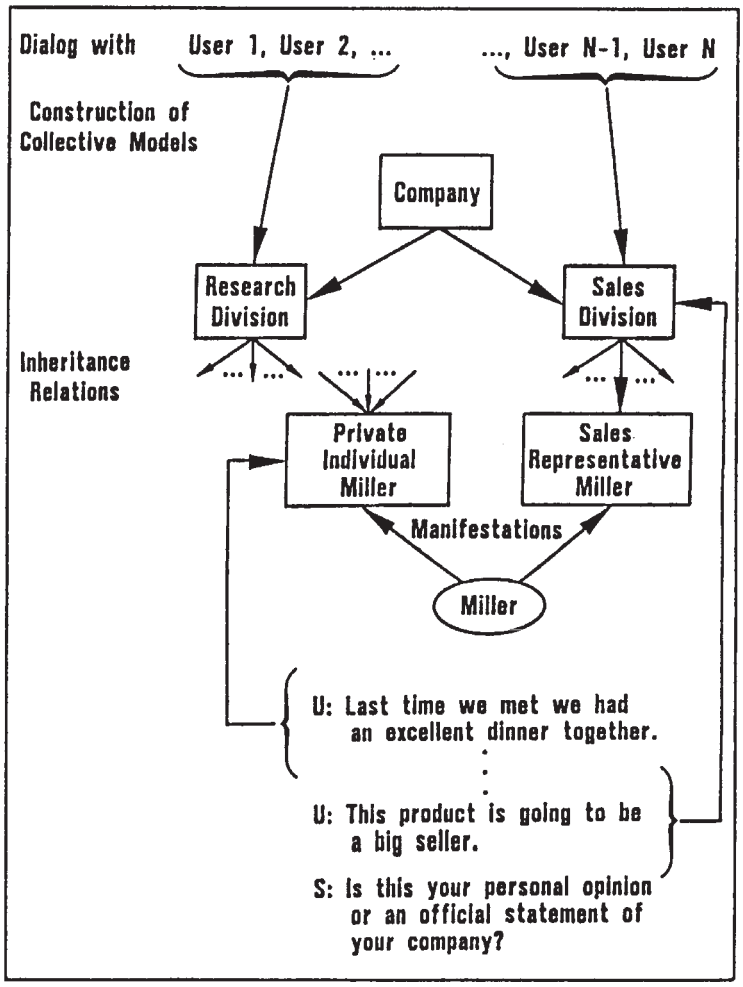


Figure 9. Copying with individual and collective beliefs in a user model

5.2. Adaptability to Diverse Conversational Settings

Today, an important trend in the design of NL dialog systems is the development of *transportable* systems. HAM-ANS (cf. [HOEP83b]), TEAM (cf. [MARP83]) and LDC [BALL84] are examples of transportable systems that can be adapted to different domains by the experts in these fields (rather than by AI or linguistics specialists).

In [HOEP83b] we proposed to go beyond domain-independence by creating dialog systems adaptable to applications that differ not only with respect to the domain of discourse, but also to dialog type, user type and intended system behavior: *transmutable systems* as we call them (see Figure 10).

Such are transportable and adaptable to diverse conversational settings and dialog games (cf. [LEVI77]). As a fictitious scenario for the application of a transmutable system let's suppose that the system's main task is to play the part of a sales representative for a particular company. Sometimes the sales manager interviews the system concerning its job performance. If s/he is dissatisfied with the system's performance, s/he asks an expert in sales promotion to instruct the system.

If you recall the DEC system family XCALIBUR, XSEL and XCON, you will notice that such a scenario is not too unrealistic: XCALIBUR [CARN83c]

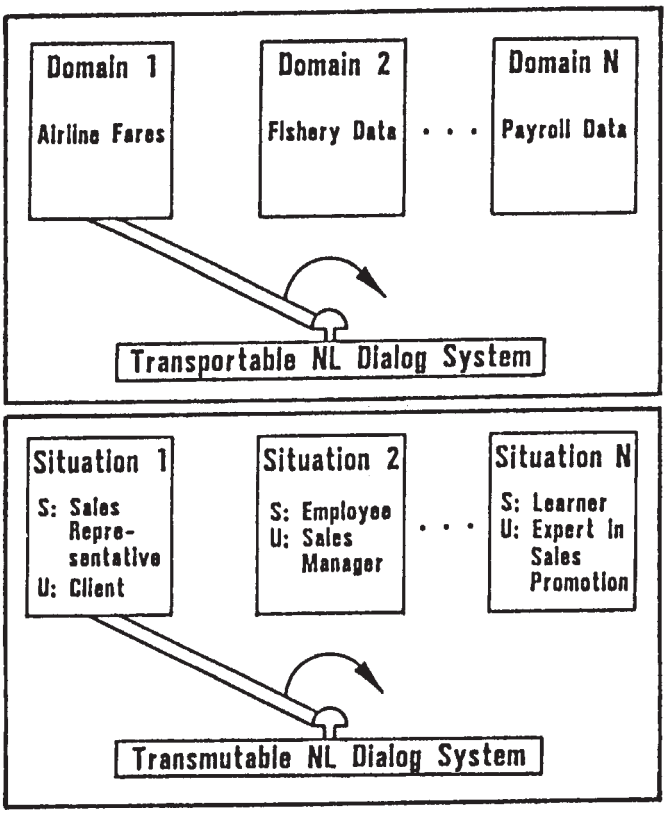


Figure 10. Transportable vs. transmutable systems

is a natural-language interface to XSEL, an automated salesman assistant which advises a customer on the selection of appropriate VAX components. XSEL then produces a sales order and passes it on to the XCON expert system for an automatic configuration of the ordered components.

Nilsson [NILS83] proposed the development of a class of computer systems that are never to be turned off. A dialog system with continuing existence seems to provide a good framework for basic research on transmutable systems. Such a system should have a constantly changing model of its various users and should be able to adapt to diverse conversational settings and to take on various communicative roles.

The major problem that developers of transmutable systems are confronted with is the lack of a representational vocabulary for the declarative description of the relationship between the system and the user, the system's intended dialog behavior, and the associated conversational tactics. Although some of the research discussed in this volume tackles aspects of this problem, a general solution requires lengthy future research on user modeling.

5.3. User Modeling as a Resource-Limited Process

Before asking one's supervisor for a raise in salary, one usually prepares oneself for this dialog by analyzing mutual beliefs and by anticipating the various possible plans

and goals of the dialog partner. But in most everyday conversations, the evaluation of complicated embedded, reflexive, or mutual beliefs and goals as well as a detailed plan analysis is a luxury which one may often not be able to afford. Thus, a computationally plausible outline of a user modeling component must provide some built-in processing restrictions that result from resource-limited real-time understanding of real-life discourse. Rosner [ROSN81] has pointed out that if one carries the user modeling approach to its extreme, there is a very good chance that nothing will get said at all (or if said, not understood) because the speaker and hearer will spend all of their time simulating each other's roles without actually doing anything else. Obviously this process must stop somewhere, but the question is where. Allen & Perrault [ALLE80], for instance, have recognized that their plan-based approach to user modeling may have to be supplemented with a rule-based one. We believe that a promising direction is the compilation of the results of a detailed plan-based analysis of speech act sequences into more efficient "shallow" conversational rules (for a first step in this direction, see the "dialog act plans" in [KOBS85a]).

A related question which so far has not been settled is at what time nested beliefs should be constructed. In the approach proposed by Perrault, Cohen and Allen, most of the possible perspectives on beliefs are already considered as computed (e.g. what one believes one's supervisor believes about one's beliefs about the supervisor's beliefs about a salary increase). Wilks [WILK83, WILK87] adopts the opposite point of view and proposes in his multiple environment approach as a least-effort hypothesis that nested beliefs should only be constructed when needed.

5.4. Prospects for Application-Oriented Research

As can be seen, research on user modeling has been very active in the last few years. A number of interesting proposals have been made concerning representation systems for user models, methods for generating assumptions about the user from his/her natural-language input, and strategies for the planning of the system's dialog contributions on the basis of its model of the user. Also, several prototype systems have been implemented in order to exemplify and further investigate these proposals.

However, a great number of fundamental problems still remain unsolved, some of which have been mentioned in the last few sections. As far as possible applications are concerned, an additional problem arises in that the development of flexible user modeling components is rather costly. Therefore, application-oriented research will probably lead in two directions:

- *Development of user modeling components in help systems for widely used existing systems*

Such a component would observe the user's interaction with the host system, form assumptions about the user's plans and experience, and provide user-specific help on request. Initial efforts along these lines are help systems for UNIX and UNIX-like operating systems [WILE84, WILE86, KEMK86], a help system for the SCRIBE text editor [RICH82], and a system which provides unsolicited advice for users of the VMS operating system [FINI83].

Although work on user modeling to date has concentrated on interfaces to software systems, it should be pointed out that there are possible applications of the methods described above to many electronics-based consumer products, even photocopiers or microwave ovens. In order not to discourage the user by the system's complexity and to help him/her to take full advantage of the capabilities of such multifunctional products, future systems must include an adaptive interface based on an explicit user model. Quinn and Russell [QUIN86] describe a photocopier with an intelligent interface, which analyzes the user's task description (e.g. 'make multiple sets of double-sided copies of an article from a book'), generates a plan to attain the expressed goal, and tells the user what s/he should do next. In testing the system, the critical role of user modeling for plan execution monitoring and the interpretation of the user's behavior (e.g. his replanning after clearing an unexpected misfeed) became clear. The system's individual user model must include information about frequency and recency of particular tasks, frequency and type of successes and failures, points of confusion, and type of help requested and received. The experiments described in [QUIN86] indicate that an intelligent interface must model more than just the operations of the machine and the actions of the user. It must also model the extent of the user's knowledge and plan when to inform the user how to handle problems.

– *Development of general user modeling components*

Another possible line of research seems to be the development of general user modeling components, comparable to the development of expert system shells in the last few years. Such components might provide, for example, a representation scheme for assumptions about a user's beliefs and goals, an inference mechanism for this representation, a mechanism for detecting inconsistencies in the assumptions and for belief revision, etc. Such a shell could then be filled with domain-specific inference rules as well as domain-specific default assumptions about any user, and then be integrated into a dialog system. Beginning research in this direction is the GUMS system [FINI86, FINI*], the TRUMP system [BONA87], the BGP-MS system [KOBS88, ALLG88] and the work of [KASS87a].

6. References

[ALLE79], [ALLE80], [ALLE82a], [ALLE83b], [ALLE86], [ALLE87], [ALLG88], [ALLG89], [APPE82b], [APPE85], [BALL84], [BOGU85], [BONA87], [BOTM87], [CARB83], [CARB*], [CARN83c], [CARR77], [CHIN*], [CLAN79], [CLOW85], [COHE78], [COHE*], [DEDE86], [DOYL79], [DOYL80], [DOYL83], [FAGI85], [FINI83], [FINI86], [FINI*], [GENT83], [GEOR83], [HALP85a], [HAYE76], [HECK88], [HINT62], [HOEP83b], [HORA87], [JAME80], [JAME82], [JAME83], [JAME*], [JOYC83], [KAPS79], [KAPS82], [KAPS83], [KASS87a], [KASS88c], [KASS*], [KEMK86], [KOBS84], [KOBS85a], [KOBS85d], [KOBS86a], [KOBS86b], [KOBS88], [KONO80], [KONO81], [LEVI77], [MARB83], [MARB87], [MARB88], [MARP83], [MAYS80b], [MCCO85a], [MCCO85b], [MCCO*], [MCKE85c], [MOOR80], [MORI84], [MORI85a], [MORI85b], [MORI*], [MURR87a], [MURR87b], [NESS87], [NILS83], [NORM86b], [NOVA84], [PERR80], [POLL82a], [POLL86a],

[POLL86b], [POWE79], [QUIL*], [QUIN86], [RICH79a], [RICH79b], [RICH82], [RICH83], [RICH*], [ROSE82], [ROSE85], [ROSN81], [SCHU83b], [SIDN83b], [SLEE82c], [SLEE85], [SOUL84], [SPAR84], [SPAR*], [SPRE87], [SPRE88], [VARD85], [WAHL83], [WAHL84], [WAHL86b], [WAHL88], [WEBB83], [WERN88], [WILE84], [WILE86], [WILK83], [WILK87]

2 Stereotypes and User Modeling

Elaine Rich

Abstract

A stereotype represents a collection of attributes that often co-occur in people. Stereotypes can play an important role in a user modeling system because they enable the system to make a large number of plausible inferences on the basis of a substantially smaller number of observations. These inferences must, however, be treated as defaults, which can be overridden by specific observations. Thus any stereotype-based user-modeling system must include techniques for nonmonotonic reasoning. This chapter will discuss the role that stereotypes can play in a user modeling system and it will outline specific techniques that can be used to implement stereotype-based reasoning in such systems.

1. Introduction

If one is given the task of constructing a model of another person, one has two choices:

- Construct the model one piece at a time as information is gleaned from experience with the person. In this approach, individual facts about people are viewed as independent events.
- Observe that, empirically, facts about people are not statistically independent events. This suggests that facts can be clustered into groups that frequently co-occur. Now the way to build a user model is by adding to it a whole cluster of facts at a time, as soon as some evidence that is known to be a predictor of the cluster is observed.

This second approach is based on the idea of a stereotype.

The original use of stereotypes was by people forming models of other people. In this chapter, we will extend the idea to include use by programs forming models of people (namely their users). The underlying idea in the two cases is the same: predictive power can be obtained by capturing naturally occurring regularities in human characteristics. Stereotypes are important to people because they permit us to make *predictions* about other people on the basis of an amount of evidence that is sufficiently small that it can be acquired before action is required [MCCA80]. For these predictions to be useful, they do not need to be completely accurate; they

only need to be more accurate than a generic guess, based on no information about the individual, would be. The role of these predictions is not to take the place of specific knowledge about the individual. Instead, their role is to provide a basis for action until such specific knowledge becomes available. As that happens, the model of the individual must be updated, and stereotype-based predictions that conflict with specific observations must be abandoned in favor of the specific facts. This same description is true of the use of stereotypes in programs. Stereotype-based inferences must be treated as default assertions that are defeasible and that may be contradicted by additional specific information as the interaction between the program and the user proceeds.

The use of stereotypes by people can produce negative results. This happens for two reasons. The first is that the stereotypes that people use may be biased so that they make exaggeratedly negative predictions about some groups of people. The second is that subsequent direct observation of an individual may not be used effectively to revise the initial predictions made by the stereotypes. There are many reasons for these problems in people. One of the main ones is that people have cognitive limitations that prevent them both from forming accurate stereotypes and from noticing characteristics that are at variance with the predictions made by stereotypes [TAYL78, HAMI76, CANT76]. In describing stereotype-based user-modeling systems, we will show that it is not necessary for such systems to experience these problems.

The idea of applying stereotypes in a user-modeling system is not new. It was used, for example, in Grundy [RICH79b], which recommends novels to people to read. It was used in UC [CHIN*], which answers questions about UNIX. It was used in HAM-ANS [MORI*], which finds an appropriate hotel room for a user. A very similar idea, that of a *perspective*, was used in the RABBIT [WILL84] system, which allowed users to formulate data base queries by critiquing examples of retrieved objects. Another similar idea, the use of sets of inference rules, each associated with a class of users, was used in UMFE [SLEE85], a general user-modeling front end system.

In the rest of this chapter, we will define what stereotypes are and show how they can be used as components of user-modeling systems.

2. The Definition of a Stereotype

A stereotype is a knowledge structure that can be used by a user-modeling system. A user modeling system, in turn, is part of the user interface component of a complete system, which includes a performance program, which we will call the *back-end system*. Within this context, we will define a *stereotype* to be:

- a *body*, which contains information that is typically true of users to whom the stereotype applies. The form of this information may vary, but it must be compatible with the inference mechanisms that are used in the back-end system.

- a set of *triggers*, which are observable values of attributes. The triggers may be a subset of the values in the body of the stereotype or they may be specific inputs or actions that do not themselves belong in the stereotype.

There may also be a third component to a stereotype:

- a set of relations between it and some number of other stereotypes in the system. The most commonly used relations describe paths in a generalization hierarchy.

SPORTS-PERSON

Triggers (Used-description "athletic")

Facets

Name	Value	Rating
Motivations	*excitement*	600
Character-Strengths	*physical-strength*	900
	perseverance	600
Interests	*sports*	800
Thrill	*5*	700
Tolerate-violence	*4*	600

Generalizations
ANY-PERSON {a canonical user model}

Figure 1. A stereotype from Grundy

Figure 1 shows an example of a stereotype called SPORTS-PERSON. It is adapted from a stereotype used by Grundy in its modeling of users as potential readers of novels. The trigger for this stereotype is that, when asked to give a self description, the user used the term, "athletic". The bodies of all Grundy stereotypes are frames (in the sense of [MINS75]), each of which consists of a set of attributes (sometimes called facets) and their associated values. There are two kinds of facets. Some, such as Motivations and Character-Strengths, can contain multiple values. Also, it turns out that those same facets contain symbolic values, such as *excitement*. Other facets, called *scalar facets*, contain single numeric values, on a scale of -5 to 5. All values have associated confidence measures, called *ratings*. These measures are on a scale of 0 to 1000. This stereotype is related to another one, ANY-PERSON, by the generalization relation. This means that the class of people described by SPORTS-PERSON is a subclass of the class of people described by ANY-PERSON.

Figure 2 shows another example of a stereotype. This one is called BUSINESS-EXECUTIVE and it might be used by a system that did travel planning. The structure of this stereotype is not a frame. It is a set of if-then rules that establish likely user subgoals, given a stated top-level goal. Although the actual structure of the knowledge that is contained in a set of stereotypes may vary, as shown in these two examples, we will need a single way to describe the knowledge about users that

BUSINESS-EXECUTIVE

```
if      goal = plan(business-trip)
then    subgoal = get-flight(first-class)
        subgoal = get-hotel(first-class, quiet, close-to(meeting-place))
        subgoal = get-car(high-quality,small)
        subgoal = get-restaurant-list(expensive)

if      goal = plan(pleasure-trip)
then    subgoal = get-flight(coach)
        subgoal = get-hotel(moderate,adjoining-rooms, pool)
        subgoal = get-car(intermediate,family-size)
subgoal = get-restaurant-list(moderate,take-children)
```

Figure 2. A stereotype for a travel system

is contained in these various structures. For simplicity, we will view all stereotypes as composed of a set of *assertions*, regardless of the actual form in which those assertions are represented.

From our definition and from these examples, it should be clear that the main structural property of a stereotype is that it groups together a set of assertions, specifically a set that typically co-occur in people. That set is made accessible by the triggers, which must correspond to readily observable events or traits. A stereotype may be *activated* whenever one of its triggers is observed (unless there is conflicting information that blocks activation. See Section 5 for a discussion of conflict resolution.) When a stereotype is activated, the assertions that it contains are integrated into the system's model of the current user. Thus, a stereotype-based user-modeling system has the ability to make a first guess at a model for a particular user on the basis of relatively little information about that specific user.

3. The Space of User Models

The term *user model* describes a large class of structures, not all of which can effectively exploit the idea of a stereotype. So before we can examine the role that stereotypes play in user modeling, we need to survey briefly the space of user models. Then we will be in a position to pinpoint the specific kinds of systems to which stereotypes can contribute.

There are several important dimensions in the space of user models. These include the following, whose descriptions are adapted from [RICH83]:

- Canonical vs. individual user models. The simplest model is one of a canonical user of a computer system. Such a model describes the characteristics of a "typical" system user. When such a model is the only one used by a system, it is not usually stored explicitly during execution, since it is the same for all users and does not change during an interaction. Instead, its only purpose is to guide initial

system design. The primary weakness of canonical user models is that they do not account for individual differences among users. Specific traits of specific users are not represented, even when they are highly salient and relevant to the task being performed. To add the ability to model these individual characteristics requires the use of individual user models. But now it becomes necessary to do two new things: user models must be stored explicitly at run time; and some mechanism for inferring the individual models from observed events must be provided.

- Long-term vs. short-term user models. Long-term models describe relatively static characteristics of users, while short-term models describe specific topics and goals in the current discourse. This view of a short-term model probably overlaps with what is usually meant by the term discourse model, but see [SCHU88, CHIN88a, COHE88, KOBS88, MORI88, SPAR88, WAHL88] for a discussion of this issue. This same distinction has also been called static vs. dynamic in [SPAR86].

In this chapter, we will focus primarily on long-term models of individual users, because it is there that stereotypes can play the most important role. Canonical user models do not need the explicit mechanisms that we will present for manipulating models during user-system interactions. Short-term models cannot generally benefit directly from the use of stereotypes because short-term models must contain information not just about general characteristics of the user but rather about details of the current interaction. It is usually necessary to get these details directly from the user or to infer them using techniques for plan and goal detection. (See, for example, [GENE82, CARB*, QUIL*].) There are, however, interesting ways in which a stereotype-based long-term model can interact with a specialized, short-term one. For example, the BUSINESS-EXECUTIVE stereotype of Figure 2 can be used to form a long-term model of the travel preferences of a business executive. That model, once formed, may be stable for a long time. But, during that time, the business executive will present a series of different short-term goals to the system. Each time that happens, the system will need to build an appropriate short-term model that can be used to guide it in helping the user satisfy the current goal. One source of information that the system can use in building these various short-term models is the set of inference rules contained in its long-term model of the same user. So, for example, the assertions that are contained in the BUSINESS-EXECUTIVE stereotype (and thus may also be contained in the long-term model of a particular business executive) may be used to augment the current short-term model with a set of appropriate subgoals that the system should attempt to achieve. Thus, although the major role of the stereotype-manipulating techniques that are discussed in this chapter is in the construction and use of long-term user models, these techniques may also be of use in systems where a major emphasis is placed on the manipulation of accurate short-term models.

4. Stereotypes and User Modeling

Stereotypes play a sort of dual role with respect to user modeling. On the one hand, a stereotype represents a kind of canonical user model since it describes a collection

5. Stereotypes and Contradiction Resolution

The use of stereotypes constitutes a form of default reasoning [REIT80]. It generates plausible but defeasible inferences and thus any system that exploits stereotypes must also exploit methods for contradiction detection and resolution. As in any default reasoning system, conflicts may arise for either or both of the following reasons:

- An incorrect inference was made on the basis of incomplete information and now, with additional information, that inference is shown to be false.
- An inference described something that was true once (presumably when it was made), but the situation has changed and the inference should be retracted. Notice that this second source of conflict can also occur in systems that do not use default reasoning.

We will be concerned here with only the first of these sources of contradiction because of our emphasis on the use of stereotypes in the construction of long-term user models.

Facet	Value	Rating	Justification
Gender	Male	900	Mail name
Thrill	5	900	MAN SPORTS-PERSON
Tolerate-violence	5	866	MAN SPORTS-PERSON
Motivations	Exitement	760	MAN SPORTS-PERSON
C-Strengths	Perseverance Courage Phys-Strength	600 700 950	SPORTS-PERSON MAN MAN SPORTS-PERSON
Interests		800	SPORTS-PERSON

Figure 4. A Grundy model of an individual user

Because conflicts may appear at any time in a stereotype-based system, it is necessary that such a system record not just assertions about users, but also justifications for each of those assertions. Without a record that states clearly why each assertion is currently believed, explicit reasoning about the validity of that belief will not be possible. The method that should be used for storing these justifications depends on the form of the assertions themselves. For example, in Grundy, assertions were represented as values of attributes. In such a system, a list of justifications can be associated with each such value. Figure 4 shows a simplified version of a Grundy user model in which this is done. In this example, two stereotypes, MAN, and SPORTS-PERSON have been activated for the user. Thus either of these stereotypes may

justify a value. In addition, direct observations (such as the fact that the user's name is one that is usually male), may serve as a justification. In other kinds of user modeling systems, other techniques for recording justifications may be used. For example, if a system of logical statements is used, it may be appropriate to use a truth maintenance system (TMS) of some sort [DOYL79] to record dependencies among the statements.

There are two kinds of contradiction that can occur in a stereotype-based system:

1. Contradiction between/among two or more stereotypes. This may occur whenever the system's observations about a user result in the activation of multiple stereotypes that make predictions about a given attribute. This category can be further divided on the basis of the resolution strategy that is appropriate:
   a. Contradiction between one stereotype and another that is either above or below it in the stereotype generalization hierarchy. For example, a COMPUTER-SCIENTIST stereotype might predict a value of HIGH for the attribute COMPUTER-COMFORT. But the more specialized stereotype, THEORETICAL-CS, might predict the value LOW.
   b. Contradiction between any other group of stereotypes. For example, an OFFICE-WORKER stereotype might predict a value of LOW for the attribute PHYSICAL-CONDITION, while, for the same attribute, the MARATHON-RUNNER stereotype would predict a value of HIGH. But both stereotypes may be activated for a single user, and, if they are, a conflict will result.
2. Contradiction between stereotype-based inferences and assertions that are based directly on observations of the user or statements by the user.

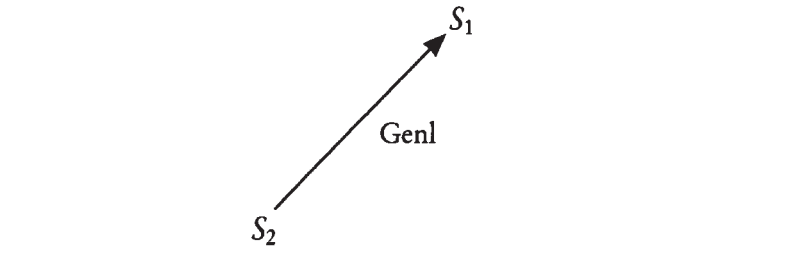


Figure 5. A fragment of the stereotype generalization structure

In case 1a, the conflict resolution procedure is straightforward. The value predicted by the more general stereotype should be ignored and the value predicted by the more specific one should be used. More precisely, let S_1 and S_2 be two stereotypes that predict contradictory values for attribute A. (If more than two stereotypes are involved and they all occur in a single generalization chain, then this procedure can be applied to successive pairs, in any order, to produce the correct result.) If S_1 occurs above S_2 in the generalization hierarchy (see Figure 5) then the

set of users contained in the class described by S_1 is a superset of the set of users contained in the class described by S_2. In this case, use the value for A contained in S_2. Otherwise, use the value contained in S_1. The justification for this procedure is that, since S_2 describes a more restricted set than does S_1, it is interpreted as describing a special case of the class described by S_1. This procedure is analogous to the standard definition of property inheritance [TOUR86], in which values associated with subclasses specify exceptions to general values associated with superclasses.

In the other two cases, the situation is somewhat more complicated. To define a procedure to handle them, it is necessary to notice that a stereotype-based inference is the result of a two-step reasoning process:

- Step 1: On the basis of the trigger, infer that the user is a member of the group described by the stereotype.
- Step 2: On the basis of the stereotype, infer specific values of the attributes mentioned in the stereotype.

When a contradiction arises, either or both of these reasoning steps may be the culprit. If a trigger signals the applicability of its associated stereotype with less than complete certainty, then even when the trigger has been observed, it may be that the corresponding stereotype should not have been activated. Likewise, if a stereotype predicts a particular attribute value with less than complete certainty, then it is possible that, even though a user is a member of the class described by the stereotype (i.e., the stereotype was correctly activated), it is nevertheless the case that this particular prediction is inappropriate for this user.

When a contradiction is detected, two kinds of remedial action may be necessary. The first is to resolve the contradiction within the user model for the current user. When there is a contradiction between two stereotypes (case 1b, above), it may not be clear how the conflict should be resolved. Some possible methods will be discussed below. But when there is a contradiction between a stereotype and a direct observation (case 2, above), the prediction of the stereotype must be ignored in favor of the direct observation. Doing this guarantees that stereotype-based predictions can drive a model only until more direct evidence is available; they can never prevent the use of such evidence once it appears. The second kind of remedial action that is required is to modify the stereotype knowledge base in such a way as to decrease the likelihood of a similar contradiction arising in the future. This step is required in both kinds (1b and 2) of conflict situations. We will describe a system that performs this step in the following section.

Resolving conflicts within a user model is an instance of the more general problem of reasoning in a default system. There are two approaches that can be taken:

- Use some sort of numeric estimates of values and of the system's confidence in those values. Define some combining functions that can be used when more than one piece of evidence relates to a particular value.
- Avoid numeric certainty measures. Instead, select, either arbitrarily or on the basis of some knowledge, one of the competing values and ignore the others.

of attribute values that are typically exhibited by users who belong to the class that the stereotype describes. But, on the other hand, stereotypes provide an excellent mechanism for constructing individual user models. Their contribution then is that they enable one, or a small number of, observable events (corresponding to the stereotype's triggers) to provide the basis for inferring probable values for a much larger set of attribute values. The key to an integrated understanding of these two views of stereotypes lies in the notion of a generalization hierarchy.

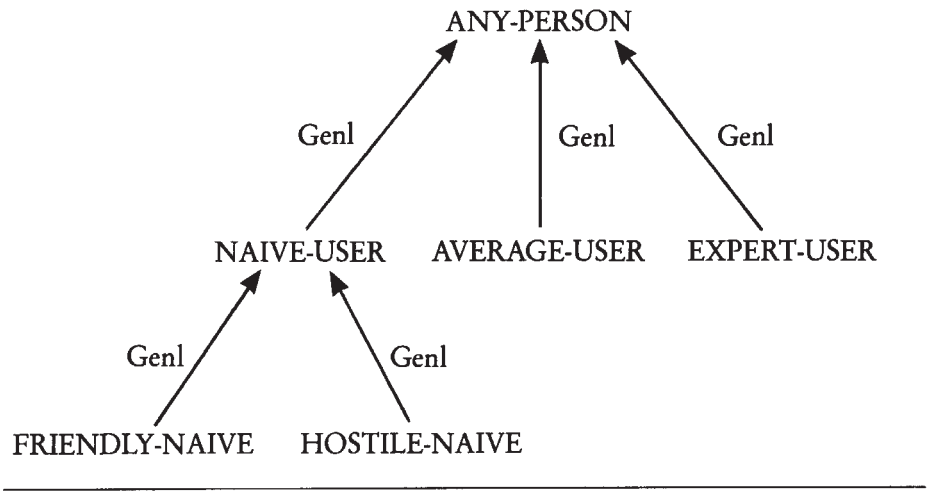


Figure 3. Stereotypes arranged in a generalization structure

As we said in Section 2, a stereotype may have associated with it a set of relations between itself and other stereotypes in the system. The most important and useful of such relations are those that define a generalization hierarchy among a set of stereotypes in a system. The most general stereotype in such a hierarchy then describes a model of a canonical system user. More specific stereotypes describe models of subclasses of system users. See Figure 3 for an example of a set of stereotypes arranged in a generalization hierarchy. In this example, we have used the property Genl as the name for the generalization relation. Other systems have used other names, such as ISA. Such a hierarchy of stereotypes functions in the same way that generalization hierarchies of other sorts function. Specifically, properties are inherited from more general stereotypes to more specific ones so that an efficient representation of shared properties is possible.

All of the stereotypes in a generalization hierarchy must have associated triggers (except the most general one, which can always be assumed to apply if more general information is not available, as it often is not at the beginning of a user's first interaction with the system.) If the only triggers that occur are those that correspond to fairly general stereotypes, then only the corresponding general information will be available for incorporation into the user model. But, if the triggers of more specific stereotypes occur, then a more specific user model will be able to be constructed.

In many systems, a combination of the two approaches may be useful. As we discuss these approaches in the rest of this section, we will have to consider three parts of the process of resolving a conflict within a user model:

- Resolve the current conflict by deciding what assertions should be contained in the user model.
- If certainty measures are being used, decide what certainty to assign to the assertions chosen above.
- If, as a result of resolving the conflict, any assertion has been retracted or modified, find all other assertions that depended on it. Retract or modify them appropriately and propagate any changes to any other dependent assertions. If certainty measures are used and any of their values change, they too must be propagated.

One issue that must be dealt with is how to terminate this third step and how to keep the amount of work spent on the process to a manageable level. Note that this propagation step means that substantial reasoning may be necessary even if the process of resolving the original conflict is trivial (as, for example, in the case (2) of a conflict between a stereotype-based prediction and a direct observation).

5.1. Resolving the Current Conflict

Any reasoning that the user modeling system is capable of performing could, in principle, be used to resolve conflicts between the predictions of different stereotypes. Any such reasoning method has available to it two kinds of information that it can use in making its decision:

- Information about the specific stereotypes that made the contradictory predictions.
- Information about the specific assertions that are in conflict.

Either, or both of these kinds of information may be useful.

Information about the stereotypes that made the conflicting suggestions corresponds to world knowledge about characteristics of people who belong to several groups simultaneously. The easiest-to-use form of this knowledge has already been presented as case 1a above. When the two stereotypes exist in a subclass-superclass relationship to each other, then it is clear that the more specific information should override the more general. Even when this simple situation does not exist, there may still be a way to use information from the generalization hierarchy to help resolve a conflict. Suppose that stereotypes A and B predict conflicting assertions and suppose that the least upper bound of A and B in the hierarchy is C. Then one strategy is to ignore the conflicting predictions of A and B and to use the corresponding prediction of C instead. This corresponds to the notion that perhaps our information about exactly which sublass of C the user belongs in is not accurate so the best we can do is to treat the user just like any member of the more general class C.

A second way that information about stereotypes themselves can be used to resolve conflicts among stereotypes is to recognize that a stereotype contains two kinds of information about people who are members of the class that the stereotype

describes. The first kind is necessary information, i.e., definitional information or information that is prerequisite to definitional information. The second kind is default information. Whenever there is a conflict between these two types, the first will win over the second. As an example, consider the two stereotypes shown in Figure 6, each of which describes a class of computer scientists. The attributes shown in boldface are necessary attributes of the class described by the stereotype.

HACKER	
hacking skill	high
math skill	med
work hours	night

THEORETICIAN	
hacking skill	low
math skill	high
work hours	day

Figure 6. Two stereotypes that may conflict

You cannot be a hacker if you cannot code. You cannot be a theoretician if you cannot do mathematics. Suppose that both of these stereotypes have been activated for a particular user and the system needs to know what the user's level of skill at hacking is. Then the value from the HACKER stereotype should be selected and the value from the THEORETICIAN stereotype should be ignored. Another way of saying this same thing applies if stereotypes are viewed as collections of logical expressions rather than as frames. Then the necessary information will take the form of standard, first-order formulae such as

$$\forall x \; \textit{hacker}(x) \rightarrow \textit{hacking-skill}(x,\textit{high})$$

Default information will be stated as such explicitly, using a higher-order predicate such as Consistent-to-Assume. Thus we might write, for example:

$$\forall x \; \textit{theoretician}(x) \wedge \textit{Consistent-to-Assume}(\textit{hacking-skill}(x,\textit{low})) \rightarrow \textit{hacking-skill}(x,\textit{low})$$

Then the inference rules that are used to form extensions (i.e., consistent theories that contain all ordinary assertions and maximal consistent sets of assertions derivable from the defaults) will prevent the default rules from applying when they would lead to an inconsistency.

So far, we have presented domain-independent ways of resolving conflicts between stereotypes by using information about the stereotypes. Yet another possibility is to augment this general system with a set of domain-specific facts that describe what to do when predictions of specific stereotypes conflict with each other. For example, consider again a travel system that might use a stereotype like the BUSINESS-EXECUTIVE one of Figure 2. Such a system might also possess a stereotype for a young person. This stereotype would differ from BUSINESS-EXECUTIVE in several ways, among them that the user is predicted, when going

on a pleasure trip, to be interested in the cheapest possible fare and probably not to be interested in a hotel room at all since he or she may be planning to stay with friends. A separate piece of knowledge that a user-modeling system in this particular context might have is that BUSINESS-EXECUTIVE should override YOUNG-PERSON. The rationale for this (which does not need to be encoded explicitly in the system at all) is that what really matters in this context is the amount of money that the user can afford to spend and that there are some young people, including those who have managed to become business executives, who have money, even though most do not. But notice that this knowledge is highly domain-specific. If one were building a system that chose the music that should be played to a user who has been put on hold on the telephone, young person should probably override business executive.

In addition to using information about the stereotypes that produced the conflicting predictions, it is possible to resolve contradictions by using knowledge about the predictions themselves. Just as in the case of reasoning about stereotypes, there are general, domain-independent strategies and there may also be more specific, domain-dependent strategies. Domain-independent strategies are those that depend only on the form of the predictions and not their actual content. The following simple, domain-independent strategies are available: (1) If the thing that is being predicted is a numeric value, then predict the average of the competing values. (2) Compute the least upper bound (in a generalization hierarchy) of the predicted values and predict it. This is the analogue of (1) when symbolic values are being predicted. (3) Prefer the older assertion since it is better established. (4) Prefer the newer assertion since it is more likely to be current. (5) Prefer not to believe any assertion that has previously been involved in a conflict (even if it won). (6) Prefer the assertion whose consequences are more attractive. Notice that these general strategies may compete with each other so they cannot all be used simultaneously without some other conflict-resolution strategy. Whenever these general strategies are not adequate, of course, any number of specific inference rules (analogous to the stereotype inference rule that business person should win over young person) can be used to specify that certain predictions should be believed over others.

5.2. Assigning a Certainty Value

It is not necessary for a stereotype-based user-modeling system to exploit explicit certainty measures. But it is often useful to do so in order that the back-end system will know how much to rely on the information in the user model. If certainty measures are used, then the procedure for activating stereotypes and combining them into a consistent user model must compute appropriate certainty values. When only one piece of evidence bears on a particular conclusion, this poses no problem, assuming that some estimate of predictive certainty is associated with each trigger/stereotype pair and with each stereotype/prediction pair. The difficulty comes when there are multiple sources of a prediction about the same thing. If the sources agree, then the resulting certainty should be higher than it would be after only one vote. If the sources disagree, then the degree of uncertainty arising from the disagreement must be computed.

The problem of representing degrees of certainty in a stereotype-based system is not any different from the problem of representing degrees of certainty in any other default reasoning system, and several frameworks for solving the problem exist. The simplest one mathematically, although not usually computationally, is to use Bayes' theorem [CHAR83], which describes the relationships that hold among various conditional probabilities. Another possibility is to use the Dempster-Shafer [SHAF76] model, in which certainty estimates are viewed not as points but as ranges of likelihood. A third possibility is to use a somewhat more *ad hoc* set of combination rules, such as the ones that were defined for the Mycin [SHOR76] system. This last approach worked well, for example, in Grundy.

5.3. Propagating the Change

Default reasoning of all sorts, including the kind introduced through the use of stereotypes, is nonmonotonic in the sense that when a new assertion is added to the system, some other assertions may need to be removed or modified. Thus it is critical to the success of a stereotype-based system that it be able to propagate the necessary modifications throughout a user model whenever any change is made.

It is useful to distinguish two kinds of propagation that may need to occur. The first is of symbolic objects. These objects may correspond either to assertions about the current user or they may correspond to justifications of those assertions. The most straightforward way to propagate symbolic objects is to build the user-modeling system on top of a reasoning system, such as a TMS [DOYL79] or an ATMS [DEKL86], that provides the mechanism for recording justifications and for restoring consistency when necessary. Here again, the needs of a stereotype system are not fundamentally different from the needs of any other system that does default reasoning.

The second kind of propagation that may need to occur is in numeric objects. For example, suppose that trigger T1 suggests stereotype S1 with some rating R1. Then S1 suggests assertion A1 with some rating R2. Thus (in the absence of any conflicting information), A1 will be asserted with some certainty, which is a function of R1 and R2 (perhaps just their product). Now suppose trigger T2 is observed and it also suggests stereotype S1, this time with the rating R2. As a result, the combined rating of stereotype S1 must be changed. But if this rating is changed, then the rating of assertion A1 must also be changed, since it was a function of the rating of S1. Now, if some other assertions were made as a result of A1, their ratings must also be changed and so forth. An issue arises here because of the way that numeric values are usually combined. Specifically, if two values are combined to form a third, then the influence of either of the originals on the third is usually such that, if one of the originals changes, then the change in the third will be smaller than the change to the original one was. This is the case, for example, if the two values are combined by multiplying them together and if they are both less than 1 (which they often are if they are viewed as certainty measures). This means that, as a change is propagated through the system, it becomes smaller and smaller. But, of course, the amount of work required to propagate it does not decrease correspondingly. Thus there often comes a point at which substantial effort is being expended on the propagation of

insubstantial changes. It is often useful to define some threshold, δ, such that any change with absolute value less than δ will not be propagated further.

6. Adaptation of Stereotype Systems

Adaptation is an important part of an effective stereotype-based system because of the difficulty of defining accurate stereotypes *a priori*. A variety of learning and adaptation techniques can be applied to stereotypes, depending on exactly the form in which the stereotypes and their triggers are represented. In the last section, we presented two broad classes of techniques for dealing with the issue of conflict resolution within a particular user model: general, domain-independent techniques that rely only on the structure of the objects being examined and not on their content, and specific techniques in which domain-specific information (such as a preference for using the BUSINESS-PERSON stereotype over the YOUNG-PERSON one) was exploited. The same two kinds of approaches are also possible for dealing with the problem of stereotype adaptation. In this section we will deal only with the first of these strategies, since it is the only one that can be described in a general way.

Because of the inherent uncertainty in stereotype-based inferences, one natural adaptation technique applies to stereotypes (such as the one shown in Figure 1) that contain information to which explicit measures of certainty have been attached. In this case, adaptation consists of modifying the certainty measures on the basis of each experience with a user. In this section, we will describe a modification system (that was used in Grundy) that takes this approach.

If we are going to use concrete interactions with users as a basis for adapting the certainty factors associated with a stereotype system, we must look for two kinds of feedback from the interactions:

- Evidence that the predictions made by a stereotype are correct and should be made again in the future in similar circumstances.
- Evidence that the predictions made by the stereotype are incorrect and should not be repeated in similar circumstances in the future.

The adaptation algorithm must then combine these two kinds of evidence into a single, coherent strategy for updating the certainties associated with stereotypes and their triggers. This strategy must deal adequately with both of the following issues:

- How should credit and blame be assigned among all the factors that made a given prediction?
- How much weight should be given to the initial values in the stereotypes and to each additional piece of evidence?

In defining Grundy's adaptation strategy, the following answers were used. For the first question, the approach is not to try to apportion credit or blame intelligently. Instead, all contributing factors are modified. This means that some inappropriate modifications will occur. But on the average, the factors that are actu-

ally responsible for the prediction will have their values changed in the appropriate direction. The others will fluctuate some, but will not show any noticeable drift. The answer to the second question is somewhat *ad hoc*, and is shown in the formulae given below. When values are being modified, the old value is weighted by a system-defined constant (which was set to 100) and then combined with the new value. When ratings are being modified, the weight given to the new value depends on the magnitude of the old rating (so low ratings can easily be changed but higher ones change only slowly). More sophisticated answers to this question are possible. For example, one might record, with each value, some measure of the total amount of evidence that has thus far contributed to it. Then the relative influence of each new piece of evidence could decrease over time. The most interesting thing about Grundy from this point of view is not that it uses a sophisticated method but that it uses an unsophisticated one that does, nevertheless, work.

In Grundy, adaptation of the stereotype system could occur every time a new fact or inference (i.e., something other than a stereotype-based prediction) was added to some facet of an individual user model. Every stereotype that had made a prediction about the value of that facet was modified. If the new fact supported the stereotype's prediction, then the stereotype was reinforced. If the new fact contradicted the stereotype's prediction, then the stereotype was weakened.

Three kinds of changes are necessary to cause this to happen. The first is to modify the actual value that the stereotype predicts for the facet. This was only done for facets with numeric values (such as Thrill and Tolerate-violence in Figure 1). For these facets, the following formula describes the change that was made:

$$\textit{NEW-STEREOTYPE-VALUE} = \frac{\textit{OLD-STEREOTYPE-VALUE} * \textit{CONSTANT} + \textit{NEW-INFO-VALUE}}{\textit{CONSTANT} + 1}$$

The second kind of change that must be made is to the rating attached to the prediction in the stereotype. When a confirmation of a stereotype's prediction occurred, the rating attached to that prediction was modified according to the following formula:

$$\textit{NEW-RATING} = \textit{OLD-RATING} + \frac{\textit{NEW-INFO-RATING}}{\textit{OLD-RATING}}$$

When a contradiction to a prediction occurred, the rating attached to that prediction was modified similarly except that the change was subtracted from rather than added to the rating:

$$\textit{NEW-RATING} = \textit{OLD-RATING} - \frac{\textit{NEW-INFO-RATING}}{\textit{OLD-RATING}}$$

The third kind of change that must be made is to the rating of all of the triggers of the stereotype that made the prediction. Recall that this is necessary because it is not possible to tell whether the problem is that the stereotype was appropriate for the user but it made an inaccurate prediction or whether the problem has arisen

	Number of users interacted with so far					
	0		10		20	
Facet	Value	Rating	Value	Rating	Value	Rating
Thrill	4	700	.9	699	.1	702
Personalities	−2	500	−1	503	.9	504
Suspense	4	700	2.6	700	1.2	700

Figure 7. Three snapshots of the Grundy MAN stereotype

because the stereotype should not have been activated for this user. The trigger ratings were modified using the same formulas that were used to modify prediction ratings.

Grundy was never run as a large-scale performance program, so there was no opportunity to monitor the evolution of its stereotypes over a very large number of users. But some interesting adaptation did occur. Recall that Grundy's task was to recommend novels to people to read. So the information contained in its stereotypes corresponded to preferences for certain kinds of books. Figure 7 shows three snapshots of part of the MAN stereotype that Grundy used. The first value and rating are the ones that were initially provided to Grundy (based on a best guess about the characteristics of the population that the stereotype would describe.) The second value and rating correspond to the computed numbers after complete interactions with ten users to whom the stereotype applied. The last value and rating correspond to the computed numbers after complete interactions with twenty such users. The Thrill facet was intended to describe the user's level of interest in a thrilling plot (on a scale of −5 to 5). The Personalities facet was intended to describe the user's level of interest in well-developed characters. The Suspense facet was intended to describe the user's level of interest in a suspenseful plot. Roughly, the original stereotype described a user who wanted action as opposed to character development. The modified stereotype described a more balanced user.

Other approaches to the adaptation problem are also possible. For example, stereotypes can be viewed as concepts, and then they can be learned with statistical concept-learning methods. Another approach that could work is to train a neural network [RUME86] to recognize instances of stereotypes.

7. Parallel Models of System Knowledge and of Users

Stereotypes provide a mechanism for organizing information about classes of users in a way that makes possible rapid inference about users. One of the important things to know about a user of a knowledge-based system is how the user's knowledge compares to the system's model of knowledge in the relevant domain. One way to make inferences of this sort possible is to organize system knowledge in a parallel

structure to that in which user knowledge is organized. In other words, something like a set of stereotypes can be used to group together pieces of system knowledge in much the same way they are used to group together knowledge about users. The user modeling system in UC [CHIN*] does exactly this in order to be able to guess quickly whether a particular user knows about a particular feature of UNIX.

8. Summary

This chapter has two main points. The first is that the idea of a stereotype, viewed as a cluster of assertions about possible users, can be useful in a user-modeling system. The second is that stereotype-based inference is a form of default reasoning. It thus possesses all of the difficulties that such systems normally possess. And some of those difficulties may be overcome by using various of the techniques that have been proposed for reasoning in default systems.

Acknowledgements. I am indebted to Jim Miller and Alfred Kobsa for substantial comments on an earlier version of this chapter.

9. References

[CANT76], [CARB*], [CHAR83], [CHIN88a], [CHIN*], [COHE88], [DEKL86], [DOYL79], [GENE82], [HAMI76], [KOBS88], [MCCA80], [MINS75], [MORI88], [MORI*], [QUIL*], [REIT80], [RICH79b], [RICH83], [RUME86], [SCHU88], [SHAF76], [SHOR76], [SLEE85], [SPAR86], [SPAR88], [TAYL78], [TOUR86], [WAHL88], [WILL84]

3 A Taxonomy of Beliefs and Goals for User Models in Dialog Systems [1]

Alfred Kobsa

Abstract

In this chapter, a taxonomy of beliefs and goals is proposed, the purpose of which is threefold: it should allow for a classification of current user models with regard to *what is being modeled* by them; it should support the designer of a user model in investigating what types of beliefs and goals are relevant to his/her particular application; and it should allow the evaluation of the expressive power of representation schemes for beliefs and goals. For each belief and goal type, examples are presented for their occurrence in application domains of dialog systems. In addition, a number of existing systems are compared with respect to what beliefs and goals are being modeled by their user modeling components, and the consequences of an observed dichotomy of user models with respect to complexity is discussed.

1. Introduction

In [WAHL*], a user model in an artificial intelligence dialog system has been defined as a knowledge source which "contains explicit assumptions on all aspects of the user that may be relevant for the dialog behavior of the system". This chapter concentrates on that part of the user model which contains the system's assumptions about the user's *beliefs, goals* and *plans*. For a dialog system, assumptions about the user's beliefs are necessary, for example, to discover misconceptions on the part of the user or to adapt its answers to the level of his/her expertise in the domain of discourse. Assumptions about the user's goals and plans can be used, for example, for providing better recommendations or for supplying additional information not explicitly requested by the user.

There are several application domains in which it is necessary that the system also take into account "objective facts", about the user. Morik & Rollinger's

[1] This research was supported by the German Science Foundation in its Special Collaborative Research Program No. 314 on AI and Knowledge-Based Systems. The development of the VIE-DPM system reported herein was partly funded by the Austrian Ministry of Science. I am indebted to David Chin, Mark Line, Gudula Retz-Schmidt and several anonymous reviewers (mostly participants of the User Modeling Workshop in Maria Laach) for their comments on earlier versions of this chapter.

[MORI85b] system, for instance, stores the information which the user provides about the number of occupants of the apartment s/he wants to rent, about the accepted price range, the age of his/her children, etc. HAM-ANS, when run in the hotel reservation domain, makes assumptions about the customer's profession, solvency, and the type of his/her trip [MORI85a, MORI*]. In most cases, however, such "objective" entries in the user model are not directly used for planning the system's dialog contributions, but for inferring probable *goals* of the user.

A look at proposed or implemented user models demonstrates that rather different types of user's beliefs and goals are being modeled in this field. In most systems, however, only a few of these belief and goal types are typically modeled, and the expressive power of the representation scheme is restricted to these particular beliefs and goals alone. In Rich's [RICH79a, RICH79b] GRUNDY system, for instance, the user model consists only of feature-value pairs which are weighted by certainty factors. In Sleeman's [SLEE85] UMFE system, concepts are labeled with markers that indicate whether the system believes / does not believe / is uncertain whether the user knows the concept. In the user model of HAM-ANS, only those beliefs of the form 'the user believes p' and those goals of the form 'the user wants p' can be represented in which p is a fully instantiated proposition about the situation. Only in systems specifically devoted to speech act recognition or dialog planning (ARGOT [COHE79, ALLE83b]; KAMP [APPE85]; VIE-DPM [KOBS85a, KOBS85d]) is a much greater number of belief and goal types modeled.

In this chapter a taxonomy of beliefs and goals is proposed, the purpose of which is threefold: it should allow the classification of current user models with regard to *what is being modeled* by them (and possibly help to avoid some confusion which seems to have recently come up); it should support the designer of a user model in investigating what types of beliefs and goals are relevant in his/her particular application; and it should allow the evaluation of the expressive power of representation schemes for beliefs and goals, since it seems that a great number of belief and goal types should be representable in user modeling components of more general natural-language dialog systems – particularly, of course, in user modeling shells designed for NL dialog systems. For each belief and goal type, examples are presented for their occurrence in application domains of dialog systems. Moreover, a number of existing systems are compared with respect to what beliefs and goals are being modeled by their user modeling components.

2. Beliefs, Goals and Plans

This is certainly not the right place to investigate what beliefs, goals and plans "really are" or should be considered to be. As a matter of fact, the ascription of these and similar "intentional" states to people, small children, animals, computers, thermostats, etc. is abundant in our commonsense explanation of behavior. We explain Prince Andrew's marrying Sarah Ferguson by attributing to him the *goal* of marrying her. And we explain the stampede of gerbils upon sighting an airplane by ascribing to them the *belief* that there is a bird of prey in the air.

In the scientific psychology of cognition, these notions have had a rather checkered fate. While fully recognized in German 19th century psychology, they were banned together with all other mental terms by behaviorism, which regarded them as unscientific since not observable. Now, in the era of cognitive science, their good reputation has been restored, though they have recently been attacked anew, this time from the side of the philosophy of psychology (cf. [STIC83]).

In the field of user modeling, the assumption that users have beliefs, goals and plans which must be modeled in dialog systems is doubtlessly a very fruitful one. Also in other fields of artificial intelligence research or computer science in general, the modeling of beliefs has been the object of increasing interest in the last few years. This is particularly the case in the fields of *multiple-agent planning systems, distributed databases* and *intelligent computer-aided instruction*. In the first field (cf. [POWE79, KONO80, ROSE82, GEOR83, ROSE85, WERN88]), a group of robots (possibly equipped with divergent perceptual capabilities) is to construct and execute a plan for the achievement of some goals. In the field of distributed databases [LEHD84, HALP85b], independent processors (each working with one database) interact with each other in order to answer queries. In both fields it seems desirable for a robot/processor to be able to represent the fact that some other robot/processor knows who knows some fact, whether there exists one who knows some specific fact, etc. In the field of intelligent computer-aided instruction [SLEE82c, DEDE86], the system makes assumptions about what a student already knows of the material to be taught, about what false beliefs s/he has, etc.

For modeling beliefs, plans and goals it thus makes sense to investigate what *types* of beliefs and goals should be discerned, how complex beliefs and goals are *composed*, and how the discerned beliefs and goals can be put into a *systematic order*. Quite in the spirit of artificial intelligence planning, we regard plans as sequences of intermediate goals, and action operators for achieving these goals, and will therefore disregard them for the remainder of this chapter. Since, in the field of user modeling, beliefs and goals can be found both on the side of the system and on the side of the user, the term *'agent'* will be employed to refer to both of them. This term was chosen deliberately since our commonsense attribution of beliefs and goals to individuals essentially depends on whether they are judged to play an active role in events of our world. The agent who models a belief or goal will be referred to as the *modeling agent*, and the agent whose beliefs or goals are being modeled will be called the *modeled agent*.

When a classification of beliefs and goals is to be defined, means are necessary for characterizing what beliefs and goals should be grouped together. In this chapter, *linguistic descriptions* which are commonly used for attributing these beliefs and goals to other agents will be employed for this purpose. However, we must be somewhat careful with such an approach since, as was pointed out by many authors in the fields of epistemic logic and linguistics, a number of problems and ambiguities are hidden in certain kinds of linguistic belief and goal descriptions. Moreover, recent research in several disciplines has revealed that there are types of beliefs that should be discerned for which no everyday description exists (for these, the technical terms "infinite-reflexive" and "mutual beliefs" have been introduced).

One of these ambiguities, which has long puzzled philosophers and which has been most prominently exposed by Quine [QUIN56], is that many linguistic descriptions of beliefs and goals can be interpreted in two different senses, namely a so-called *de dicto* and a *de re* reading. For example, when John watches a man with a pipe stealing into Buckingham Palace, a speaker might describe John's beliefs from John's perspective (i.e., *de dicto*) as follows:

(1) John believes that the man with the pipe is a spy.

Assume that the speaker, unlike John, believes that the suspicious character is no less a person than the famous Captain Maigret solving his latest case. Seen from the speaker's perspective, John's belief then can also be described *de re* by

(2) John believes that Maigret is a spy.

The problem is that, in this particular situation, (2) is true only when interpreted in a *de re* sense, and false when interpreted in a *de dicto* sense. The general problem is that belief and goal descriptions can be interpreted both from the speaker's perspective, and from the perspective ("belief context") of the individual whose beliefs or goals are being described.[2] In this chapter, all linguistic belief descriptions (such as 'agent a *does not believe that* p' or 'agent a *does not want* p') should be read in a *de dicto* sense, i.e. from the point of view of the individual a. Belief descriptions de dicto "essentially involve the believer's conception of the issue at hand" [BURG78: 341]. For checking whether the beliefs or goals attributed to agents by the use of such a description have been correctly identified, readers should put themselves into the position of this agent and test whether they could exhaustively *report* these beliefs or goals by replacing the variable 'a' in the description with the pronoun 'I' (which yields 'I *do not believe that* p' and 'I *do not want* p' in our example).[3]

Another problem with linguistic belief descriptions concerns the meaning of the concept 'know'. Philosophers have argued that a belief description such as 'a knows that p' implies that p is the case. In the field of user modeling, however, the point of reference for judging the correctness of a belief is not the "world", but the beliefs of the system. It makes no sense to represent *within* the system beliefs of the system describable by 'The system believes that ... But actually ...' or 'The system believes that ... But the user knows correctly that ...'. Constellations like these are only observable from *outside* the system. Thus the concept 'know that p' is used in a somewhat different sense in this chapter: an agent a1 believes that an agent a2 knows p if a1 believes a2 believes p and a1 also believes p. Definitions for 'know whether p', 'know the x such that p(x)', 'want to know the x such that p(x)', etc. will be given below.

---

[2] For related views, see [MOOR82, KRAU83, RAPA84, BARN86]. As was pointed out by Stich [STIC83], however, this "ambiguity" of belief or goal descriptions is mostly due to the fact that they are seen in isolation. When investigated in a situational context, there is only one possible reading.

[3] Here I ignore the problem of the substitution of so-called 'quasi-indicators' such as personal pronouns and demonstratives. See [RAPA84] for a discussion.

3. Basic Beliefs

Basic beliefs of an agent are beliefs which are *not* about goals or beliefs of the same or some other agent, but about facts in a certain domain (the so-called *situation*). Beliefs which *are* about other beliefs and goals will be called *complex*. (The system's beliefs about the situation, for instance, form basic beliefs, whereas the system's beliefs about the user's beliefs or goals are already complex.) We will see in the next few sections, however, that complex beliefs and goals can be recursively defined and contain one basic belief or goal of a given agent (e.g. the user). It therefore makes sense to investigate basic beliefs and goals from the user's perspective as well, as we do below.

Situational facts may be individual or general. Individual facts comprise single events in the past, present or future. General facts concern general properties of objects in the situation, general relations between objects, general rules, etc. Examples of both will be given below. The symbol 'p' will be used to refer to arbitrary individual or general facts.

3.1. Basic Saturated Beliefs

Saturated beliefs are beliefs about a fact p, with p being a proposition the formalization of which in predicate calculus contains no free variables or existential quantifiers. There exist 4 types of basic saturated beliefs (the three periods at the beginning of each belief description have to be replaced with the name of an agent):

(1.1.1.) ... *believes that* p.
(1.1.2.) ... *does not believe that* p.
(i.e. believes that not p.)
(1.1.3.) ... *is uncertain whether* p.
(1.1.4.) ... *knows whether* p.
(i.e. has a definite attitude as to whether p.)

Type (1.1.1.) forms the most frequent kind of belief that is represented in current user models. At the level of general facts, such beliefs are modeled for the purpose of representing what concepts, rules or procedures a user knows about. At the individual level, they represent the objects and events that occurred explicitly or implicitly in the preceding dialog.

Examples include:

(3) The user knows the meaning of the concept 'DIPLOCOCCUS-PNEUMONIAE' (general fact; [SLEE85]).
(4) The user knows the rule 'Elevated cyclic AMP leads to increased osteoclast activity' (general fact; [WALL84]).
(5) The user believes that the function LIST combines two lists into a single one (general fact; [REIS85])
(6) The user believes that Tony has a tennis game in the afternoon (individual fact; [MARS83])

(7) The user believes that John loves Mary (e.g., since the system told him/her so; individual fact; [KOBS85a])

All basic beliefs and goals to be discussed here should be regarded as being embedded in a complex belief or goal of the system (a full description of the first belief would read, for instance, that the system believes that the user knows the concept 'DIPLOCOCCUS-PNEUMONIAE').

Beliefs of type (1.1.2.) express that an agent definitely does not believe that a fact p holds. An example would be

(8) ... the user *does not believe* that the system is printing file F1.

This belief would be a pragmatic presupposition of the user's command 'Print F1' and is inferred by default in the VIE-DPM system when such a command is entered. Remember that all belief descriptions are to be interpreted *de dicto* here, and that p describes a fact in the situation. Descriptions of the form 'a does not believe that p' and 'a believes that not p' are then equivalent.[4] Seen from a's perspective, the belief reports 'I do not believe that p' and 'I believe that not p' are equivalent, apart from possible pragmatic differences regarding the emphasis of assertion and the strength of the negative force (cf. [HORN78]). (8) is thus equivalent to

(9) ... the user believes that the system is not printing file F1.

Moreover, it should be stressed that entries in the user model expressing the assumption that a user does not know a concept [SLEE85, FINI86, FINI*, CHIN86, CHIN*] do not form beliefs of type (1.1.2.), but are "not-beliefs" *of the system* about beliefs of the user (more precisely, they are beliefs of the type S (3.1.2.) U (1.1.1.),[5] which will be dealt with later on).

Beliefs of type (1.1.3.) occur when the assumption that the user is uncertain about some fact is to be represented in the user model. In VIE-DPM, such an assumption is made, for example, based on the user's question to the system *whether p is the case*. This explicit representation of a *user's uncertainty* must be discerned from the case that there is no explicit information in the user model about whether the user believes or does not believe p. In the latter case, it can at best be *inferred* that the *system* is uncertain whether the user believes, does not believe, or is uncertain whether p (these are beliefs of type S (3.1.3.) U (1.1.1.) – (1.1.3.),[6] which will be discussed below). In most representation systems, neither kind of uncertainty can be explicitly represented.

Beliefs of type (1.1.4.) are most important in the case of Y/N-questions of both the user and the system. In ARGOT [ALLE80, ALLE83b] and VIE-DPM [KOBS85a], a precondition for the applicability of a Y/N-question is the belief

[4] This equivalence does not hold if p, instead of being a description of a fact in the situation, is itself a belief or goal description (which must also be interpreted *de dicto*). In this case, the general rule applies that 'a does not believe that p' can be inferred from 'a believes that not p', but not vice versa (cf. [PART70, KOBS85a]).

[5] 'a1 (3.1.2.) a2 (1.1.1.)' is a concatenated belief description which characterizes a *subtype* of type (3.1.2.) beliefs.

[6] '(1.1.1.) – (1.1.3.)' signifies that '(1.1.1.)', '(1.1.2.)' or '(1.1.3.)' can be selected at this point.

that the addressee knows whether p. Thus, when the user enters such a question whether p, the system assumes s/he believes that the system knows whether p. On the other hand, before posing such a question to him/her, the system checks whether it believes the user to know whether p.

3.2. Basic Unsaturated Beliefs

Basic unsaturated beliefs are similar to saturated ones except for the fact that existential quantification occurs in their formalization in predicate logic. In other representational schemes, e.g. that of VIE-DPM [KOBS85c], no specific additional construct need be introduced for their representation. Basic unsaturated beliefs can be divided into 5 types:

(1.2.1.) ... *believes that* ∃ x p(x).
(1.2.2.) ... *does not believe that* ∃ x p(x).
(i.e. believes that not ∃ x p(x).)
(1.2.3.) ... *is uncertain whether* ∃ x p(x).
(1.2.4.) ... *knows whether* ∃ x p(x).
(i.e. has a definite attitude as to whether ∃ x p(x).)
(1.2.5.) ... *knows the x such that* p(x).
(i.e. knows more about the x such that p(x).)

Beliefs of type (1.2.1.) are among the presuppositions of wh-questions. For example, when the user asks the VIE-DPM system for the x such that p(x), the system infers, among other things, that the user *believes that* ∃ x p(x). This presupposition in the user's question could then be checked against the system's knowledge (cf. [KAPS79, KAPS83]). Beliefs of types (1.2.2.)–(1.2.4.) are rather similar to their saturated counterparts (1.1.2.)–(1.1.4.). Beliefs (1.2.2.) and (1.2.3.) have not been dealt with in user modeling components developed to date, though both also occur in domains which are currently being investigated. A belief of type (1.2.3.), for instance, is among the presuppositions of the Y/N-question whether there is an x such that p(x) (e.g., "Is there a train to Toronto tomorrow morning?"). Cooperativity requires that somebody who poses this question must be *uncertain whether* ∃ x p(x).

Beliefs of type (1.2.5.) are also among the presuppositions of wh-questions. When the user asks the VIE-DPM system for the x such that p(x), the assumption is made that the user *believes* the system to "*know* this x". Conversely, before asking the user such a question, the system checks whether it *believes* the user to know this x. There has been a long debate in the fields of epistemic logic and the philosophy of language as to what it means for a modeling agent to attribute to a modeled agent the knowledge of the x such that p(x). Kaplan [KAPD69], Konolige [KONO81] and Appelt [APPE85] argue that this means that the modeling agent assumes the modeled agent to possess a *"standard name"* or *"rigid designator"* for x, i.e. a term denoting x in all possible worlds. Allen [ALLE83a] and Haas [HAAS86] convincingly refute such a view and demonstrate that the information an agent must possess about an x in order to *"know"* it (e.g. its name, its address, ways to find it, etc.) is essentially context-dependent. The only necessary (ergo context-independent)

requirement for the information which the modeled agent is assumed to have is that it would allow the modeling agent to uniquely identify x in the situation. Thus the belief description (1.2.5.) should be regarded as being synonymous with "...believes more facts about the x such that p(x), which would allow the modeling agent to identify x".

4. Basic Goals

Basic goals are similar to basic beliefs in that they are about a situation and not about beliefs and goals of other agents. Unlike beliefs, however, they are not about the current situation but about a desired situation. Thus, for an agent to have a goal p means to desire a situation in which p is the case. As is the case for the above belief descriptions, goal descriptions are also to be read in a *de dicto* sense.

4.1. Basic Saturated Goals

Basic saturated goals are about a fact p in a desired situation. As is the case for basic saturated beliefs, p's formalization in predicate logic must not contain free variables or existential quantifiers. Four types of basic saturated goals can be discerned:

(2.1.1.) ... *wants* p.
(2.1.2.) ... *does not want* p.
(i.e. wants p not to be the case.)
(2.1.3.) ... *has no preference with respect to* p.
(2.1.4.) ... *wants to know whether* p.
(i.e. wants to have a definite attitude as to whether p.)

Goals of type (2.1.1.)–(2.1.3.) are rather similar to the corresponding beliefs (1.1.1.)–(1.1.3.). In HAM-ANS [MORI85a] and [MORI*], goals of type (2.1.1.) are attributed to the user in order to predict (and possibly influence) his/her decisions with respect to hotel rooms to be rented. In ARGOT [COHE79, ALLE83b], KAMP [APPE85] and TRACK [CARB83, CARB85b, CARB*], assumptions about such basic goals are compiled into hypothesized user plans. Similar to the belief description (1.1.2.), the goal description (2.1.2.) is equivalent to 'wants not p', when read in the *de dicto* interpretation. Such goals occur in Martins' [MARS83] MBR system which represents, for example, that an agent does not want the faculty meeting to be held in the morning. (Actually, Martins only talks about "belief spaces"; however, some of the formulas contained therein may be more profitably regarded as descriptions of an agent's goals.) It seems that goals of type (2.1.3.) have not been employed up till now in any user modeling components. Assumptions relating to this type might be important in a consultation process in which the system tries to determine whether the user will agree with the side-effects of a plan proposed to him/her for the achievement of his/her principal goal.

Goals of type (2.1.4.) are fundamental in the field of speech act recognition and planning dialog (ARGOT, VIE-DPM). System beliefs about such user goals

are generated, for example, from Y/N questions of the user; they then form the starting point for appropriate search and answer generation processes of the system. In ARGOT and TRACK they are also used for inferring further goals of the user, for example that the user wants p or does not want p (beliefs (2.1.1.) or (2.1.2.), respectively). In UC [WILE86, CHIN*], the assumption that the user does not know p (i.e., a belief of type (3.1.2.)) is inferred.

4.1.1. Basic Unsaturated Goals

Besides basic saturated goals, there exist 5 types of basic unsaturated goals:

(2.2.1.) ... *wants* ∃ x p(x).
(2.2.2.) ... *does not want* ∃ x p(x).
(i.e. wants it not to be the case that ∃ x p(x).)
(2.2.3.) ... *has no preference with respect to* ∃ x p(x).
(2.2.4.) ... *wants to know whether* ∃ x p(x).
(i.e. wants to have a definite attitude as to whether ∃ x p(x).)
(2.2.5.) ... *wants to know the x such that* p(x).
(i.e. wants to know more about the x such that p(x).)

Goals (2.2.1.)–(2.2.4.) are rather similar to goals (2.1.1.)–(2.1.4.), the desired situation merely being "vaguer" in the unsaturated goals. System assumptions about user goals of type (2.2.5.) are important for user modeling since they form the basis for dealing with wh-questions of the user (cf. [ALLE80, ALLE83b, CARB83, CARB85b, APPE85]). In VIE-DPM, the conceptual representation of a user's NL wh-question is transformed into a representation of a user's belief of type (2.2.5.) [KOBS85a, KOBS86a]. Such a belief, wherever it stems from, then gives rise to a search process in the system's knowledge base, and eventually to the generation of an appropriate response. As was already discussed for beliefs of type (1.2.5.), however, no further context-independent interpretation can be given to goal description (2.2.5.) other than that the respective agent wants to believe more facts about x, so that x can be discerned from all other objects the agent knows about. What exactly an agent wants to know about such an x (its name, its address, ways to find it) can only be inferred from the context (if at all), and can only then be explicitly represented.

5. Beliefs with Respect to Other Agents' Beliefs and Goals

The beliefs which an agent holds may refer not only to the current situation, but also to beliefs and goals of other agents. The beliefs of a dialog system about the user's beliefs and goals are of that kind. Beliefs about beliefs can, in principle, be arbitrarily embedded, i.e. an agent a1 may hold beliefs about the beliefs/goals of an agent a2 about the beliefs/goals of an agent a3, etc. One may suppose, however, that arbitrarily complex beliefs do not occur in dialog situations, due to resource limitations. Current work in user modeling and dialog planning also provides practical

evidence for the existence of such a limitation: in the planning of dialog contributions, the depth of nesting hardly ever goes beyond the levels of 4-5 (cf. [SEAR69, COHE79, ALLE83b, AIRE83, AIRE84, KOBS85a]).

The symbol '(1.1.1.)–(4.2.5.)' at the ends of the belief descriptions to follow may be replaced by any belief or goal description. Thus, arbitrarily complex descriptions can be generated. Every description of such a complex belief, of course, must include the description of a *basic* belief or goal.

The general case of belief and goal embedding will probably not be so frequent in the field of user modeling. For some examples of beliefs about "third-party beliefs", i.e. agent beliefs in general, see [WAHL*, SPAR84, SPAR*]. Much more important is the special case of reflexive beliefs, in which a1 and a3 are identical.[7] Their most frequent subtypes include the so-called *infinite-reflexive* and, more specifically, *mutual beliefs*, which will be dealt with later on.

5.1. Beliefs with Respect to Beliefs and Goals of Known Agents

There exist 5 types of beliefs with respect to beliefs and goals of known agents:

(3.1.1.) ... *believes that* a (1.1.1.)–(4.2.5.)

(3.1.2.) ... *does not believe that* a (1.1.1.)–(4.2.5.)
(i.e. believes that not a (1.1.1.)–(4.2.5.))

(3.1.3.) ... *is uncertain whether* a (1.1.1.)–(4.2.5)

(3.1.4.) ... *knows whether* a (1.1.1.)–(4.2.5.)
(i.e. has a definite attitude as to whether a (1.1.1.)–(4.2.5.))

(3.1.5.) ... *believes that* a (1.1.1.)–(4.2.5.) *and believes that* a *believes that* ... (3.1.5.)

Beliefs (3.1.1.)–(3.1.4.) correspond to beliefs (1.1.1.)–(1.1.4.) except for the fact that they are about beliefs or goals, not about situations. It should be noted that a belief of the form a1 (3.1.1.) a2 (1.1.2.) is different from a1 (3.1.2.) a2 (1.1.1.), and a1 (3.1.1.) a2 (1.1.3.) is different from a1 (3.1.3.) a2 (3.1.1.). Thus, for example, the following pairs of belief descriptions express different facts when interpreted in our standard *de dicto* sense:

(10) S *believes that* U *does not believe that* Johnsons Inc. have ordered three F34 engines.

(11) S *does not believe that* U *believes that* Johnsons Inc. have ordered three F34 engines.

(12) S *believes that* U *is uncertain whether* Johnsons Inc. have ordered three F34 engines.

(13) S *is uncertain whether* U *believes that* Johnsons Inc. have ordered three F34 engines.belief types

[7] I will not deal with so-called "meta-beliefs" here (i.e. beliefs which can be described, for example, by "a1 believes that a1 believes ...") since their role in user modeling is still rather unclear (but see [CHIN*]).

In cases (10) and (12), S believes that U's beliefs could be described from U's point of view by the descriptions "I do not believe (I am uncertain whether) Johnsons Inc. have ordered three F34 engines". In cases (11) and (13), the system does not believe (is uncertain whether) this could be done. The inferences that can be drawn in each case are different. In possible-world representations (e. g. [HALP85a, MOOR80, APPE85]), which are mostly concerned with "knowledge" rather than beliefs, beliefs such as (10) can be represented, whereas beliefs such as (11) cannot.

A specific constellation of reflexive beliefs is type (3.1.5.), the so-called *infinite-reflexive* belief [KOBS84], which seems to be quite fundamental in belief and goal recognition as well as dialog planning. An implicit requirement in the belief description of (3.1.5.) is that the agent who instantiates 'a' must be different from that entered at '...'. An example for an infinite-reflexive belief is

(14)(a) S *believes that* U *wants to know whether* John loves Mary, and
(b) S *believes that* U *believes that* S *believes that* U *wants to know whether* John loves Mary,
(c) etc.

(14) is part of the inferences which the VIE-DPM system draws about the user when s/he enters the Y/N-question "Does John love Mary?". Beliefs of type (3.1.5.) should not be seen as an actual infinite set of reflexive beliefs but rather as *primitive beliefs* from which each member of this set can be inferred when needed.

A specific belief constellation composed of infinite-reflexive beliefs and a basic belief is the so-called *mutual belief*. In our notation, such a belief can be defined as follows:

(15)(a) a1 (1.1.1.)
(b) a1 (3.1.5.) [a2 (1.1.1.)]
(c) a1 (3.1.1.) a2 (3.1.5.) [a1 (1.1.1.)]

The elements in square brackets characterize the type of belief description that must be entered in (3.1.5.) at the place denoted by '(1.1.1.)–(4.2.5.)'. An example for such a mutual belief would be:

(16)(a) S *believes that* Peter loves Mary, (from 15a)
(b) S *believes that* U *believes that* Peter loves Mary, (from 15b)
(c) S *believes that* U *believes that* S *believes that* Peter loves Mary, (from 15c)
(d) S *believes that* U *believes that* S *believes that* U *believes that* Peter loves Mary, (from 15b).
(e) etc.

It can be shown easily that definition (15) is equivalent to the shorter classical one which reads as follows:

(17)(a) a1 *believes that* p.
(b) a1 *believes that* a2 *believes that* p, (a) and (b).

Mutual beliefs have been intensively investigated in the fields of philosophy of language [LEWI69, SCHI72], [CLHH81, SMIB82], psychology [CLHH83], artificial intelligence [PERR81, JOSH82, NADA83], and distributed databases [LEHD84,

HALP85b]. In the field of user modeling, however, the concept of mutual beliefs seems to be somewhat too narrow. For example, the inferences which can be drawn about a user's goals and beliefs from his/her dialog contributions do not form mutual, but infinite-reflexive beliefs. The same holds true for the assumptions which the user makes about the system's beliefs and goals as a result of its dialog contributions (these assumptions must be anticipated by the system in the process of dialog planning).

Since infinite-reflexive beliefs are so fundamental in dialog systems equipped with a user modeling component, a user model should provide simple means for representing them. Currently, infinite-reflexive beliefs can be modeled in ARGOT and VIE-DPM only. Appelt's [APPE85] KAMP system and, to a limited extent, Konolige's [KONO81] representation allow mutual beliefs to be expressed. Also, the beliefs that are modeled in HAM-ANS are interpreted by Hoeppner & Morik [HOEP83a] as being mutual beliefs.

5.2. Beliefs with Respect to Beliefs and Goals of Unknown Agents

There exist 5 types of beliefs with respect to beliefs and goals of unknown agents:

(3.2.1.) ... *believes that there exists an a1 such that* a1 (1.1.1.) - (4.2.5.)

(3.2.2.) ... *does not believe that there exists an a1 such that* a1 (1.1.1.) – (4.2.5.)
(i.e. believes that no a1 exists such that a1 (1.1.1.) – (4.2.5.))

(3.2.3.) ... *is uncertain whether there exists an a1 such that* a1 (1.1.1.) – (4.2.5.)

(3.2.4.) ... *knows whether there exists an a1 such that* a1 (1.1.1.) - (4.2.5.)
(i.e. has a definite attitude as to whether there exists an a1 so that a1 (1.1.1.) – (4.2.5.))

(3.2.5.) ... *knows the a1 such that* a1 (1.1.1.) – (4.2.5.)
(i.e. knows more about the a1 such that a1 (1.1.1.) – (4.2.5.))

In the field of user modeling, these types of beliefs are probably not so important. They might be relevant in the fields of multiple-agent planning systems and distributed databases since it seems desirable for a robot/processor to be able to represent the fact that some other robot/processor knows who knows some fact, whether there exists some agent who knows some fact, etc. Currently, such assumptions can be represented only in VIE-DPM, but are not put to use.

6. Goals with Respect to Other Agents' Beliefs and Goals

6.1. Goals with Respect to Beliefs and Goals of Known Agents

There exist 4 types of goals with respect to beliefs and goals of known agents:

(4.1.1.) ... *wants* a (1.1.1.) – (4.2.5.)

(4.1.2.) ... *does not want* a (1.1.1.) – (4.2.5.)
(i.e. wants it not to be the case that a (1.1.1.) – (4.2.5.))

(4.1.3.) ... *has no preference with respect to whether it should be the case that* a (1.1.1.) – (4.2.5.)

(4.1.4.) ... *wants to know whether* a (1.1.1.) – (4.2.5.)
(i.e. wants to have a definite attitude as to whether a (1.1.1.) – (4.2.5.))

Except for (4.1.3.), all these goals are very important in user modeling. They frequently occur in the recognition process for user intentions and in the generation of dialog contributions of the system. In ARGOT, for instance, the user's assertion *that p* leads to the mutual belief that the user wants p to be believed by the system (i.e. U (4.1.1.) S (1.1.1.)). In VIE-DPM, a user's question *whether p* leads to the assumption that the user wants it to be mutually believed whether or not the system believes p (i.e. ... U (4.1.4.) S (1.1.1.)). On the other hand, in the process of dialog planning, VIE-DPM produces an assertion that p upon discovering the following beliefs in its knowledge base:

(17)(a) The system *believes that* the user *wants to know whether* the system *believes* p (i.e. S (3.1.1.) U (4.1.4.) S (1.1.1.)).

(b) The system *believes* p (i.e. S (1.1.1.)).

Goals of type (4.1.3.) currently do not seem to be relevant in user modeling. However, as soon as one takes aspects of privacy into account, it might become interesting to model the assumption that the user doesn't care if some third party knows some fact about him/her. At the moment, such beliefs can be represented in KAMP and VIE-DPM only.

6.2. Goals with Respect to Beliefs and Goals of Unknown Agents

5 types of goals with respect to beliefs and goals of unknown agents can be discerned:

(4.2.1.) ... *wants* ∃ a1 *such that* a1 (1.1.1.) – (4.2.5.)

(4.2.2.) ... *does not want* ∃ a1 *such that* a1 (1.1.1.) – (4.2.5.)
(i.e. wants no a1 to exist such that a1 (1.1.1.) – (4.2.5.))

(4.2.3.) ... *has no preferences with respect to whether there should be an a1 such that* a1 (1.1.1.) – (4.2.5.)

(4.2.4.) ... *wants to know whether there exists an a1 such that* a1 (1.1.1.) – (4.2.5.)
(i.e. wants to have a definite attitude as to whether there exists an a1 such that a1 (1.1.1.) – (4.2.5.))

(4.2.5.) ... *wants to know the a1 such that* a1 (1.1.1.) – (4.2.5.)
(i.e. wants to know more about the a1 such that a1 (1.1.1.) – (4.2.5.))

The goals of this group do not seem to be relevant for user modeling since beliefs or goals of unknown third parties currently do not belong to the domain of discourse of artificial intelligence dialog systems. This might change, however, as soon as privacy considerations are taken into account. Goals of type (4.2.4.) and

(4.2.5.) are probably interesting for the fields of multiple-agent planning systems and distributed databases when a robot or processor is searching for some other who shares its assumptions. Currently, such goals can be represented only in VIE-DPM, but are not put to use.

7. A Classification of Existing User Models

In this section, a tentative classification of several existing user models is presented with respect to the proposed belief and goal taxonomy (cf. Table 1). Its purpose is to render transparent the sorts of beliefs and goals being modeled in the respective systems, so that designers of user models can compare their application with that of existing systems. Emphasis was placed on systems in which a greater number of beliefs and goals have been modeled.

This classification should not be regarded as the last word in the matter, since it is somewhat difficult to judge whether a certain belief or goal type is modeled in a given system. A necessary condition was that the representational scheme underlying the user model provide formal means for an *explicit* representation of the particular belief or goal type. This, of course, does not automatically mean that the representation is complete or fully adequate. For example, some systems (GUMS, HAM-ANS, UMFE) provide means for representing the system's beliefs about a user's beliefs or goals (i.e. beliefs of the form S (3.1.1.) U (1.1.1.) or U (2.1.1.)); their representation, however, does not extend to beliefs of the user about third parties' beliefs or goals. In other systems (e.g. KAMP, VIE-DPM), a number of belief and goal types that can be represented in the user model do not actually occur in the current operation of the system. Therefore, a third category has been introduced in Table 1 to mark those belief and goal types that are representable in the knowledge base of the user modeling component, but not yet put to use.

The most striking result of the comparison in Table 1 is that systems which exploit the user model for *explicit* speech act recognition and dialog planning (i.e., ARGOT, KAMP, VIE-DPM) have to model far more beliefs and goals than systems which only employ the user model for interacting with the user in a more personalized way (e.g. by taking into account his/her presumed goals or background knowledge). Systems in the latter application domain use an explicit representation only for beliefs of the form S (3.1.1.) – (3.1.3.) U (1.1.1.) & (2.1.1.). An intermediate case is the Unix Consultant (UC), which analyzes and generates a restricted number of speech acts only [WILE86, CHIN86, CHIN*].

This observation might have implications for the design of general user modeling systems, i.e. shell systems that are to be employed in a large number of different dialog systems (cf. [FINI*]). It seems that designers of such tools must decide whether or not their system should support the explicit recognition and planning of dialog contributions as well. In this case, the system must possess a considerably greater expressive power for belief and goal representation than is required otherwise.

Table 1. A comparison of existing user models with respect to the belief and goal taxonomy

	ARGOT Speech Act Recognition and Planning in Train Domain	GUMS Inferences about Users' Beliefs; Shell System	HAM-ANS Cooperative Dialog in Hotel Reservation	KAMP Planning Referential Expressions in Assembly Domain	MBR Conflict Resolution in Time Scheduling	TRACK User Plan Recognition and Cooperative Response in Course Scheduling	UC Help System for Unix	UMFE Personalized Explanation of Medical Reasoning	VIE-DPM Inferring Users' Beliefs and Goals; Speech Act Planning; Shell System
(1.1.1.)	×	×	×	×	×	×	×	×	×
(1.1.2.)	×	—	—	×	×	×	×	—	×
(1.1.3.)	—	×[1]	—	×	—	—	—	—	×
(1.1.4.)	×	—	—	×	—	—	—	—	×
(1.2.1.)	×	—	—	•	—	×	—	—	×
(1.2.2.)	×	—	—	•	—	—	—	—	×
(1.2.3.)	—	—	—	•	—	—	—	—	•
(1.2.4.)	×	—	—	•	—	—	—	—	×
(1.2.5.)	×	—	—	×	—	—	—	—	×
(2.1.1.)	×	—	×	×	×[4]	×	×	—	×
(2.1.2.)	×	—	—	×	×[4]	•	×	—	×
(2.1.3.)	—	—	—	×	—	—	—	—	•
(2.1.4.)	×	—	—	•	—	×	•	—	•
(2.2.1.)	×	—	•	•	—	—	—	—	×
(2.2.2.)	×	—	—	•	—	—	—	—	×
(2.2.3.)	—	—	—	•	—	—	—	—	•
(2.2.4.)	×	—	—	•	—	—	—	—	•
(2.2.5.)	×	—	—	×	—	×	×	—	×
(3.1.1.)	×	×	—	×	×	×	×	×	×
(3.1.2.)	×	×	—	—	—	×	×	×	×
(3.1.3.)	—	×	—	×	—	—	×	×	×
(3.1.4.)	—	—	—	—	—	—	—	—	•
(3.1.5.)	×	—	×[2]	×[3]	—	—	—	—	×
(3.2.x.)	—	—	—	—	—	—	—	—	•
(4.1.1.)	×	—	—	×	—	—	×	—	×
(4.1.2.)	×	—	—	—	—	—	×	—	×
(4.1.3.)	—	—	—	•	—	—	—	—	•
(4.1.4.)	—	—	—	•	—	—	—	—	•
(4.2.x.)	—	—	—	—	—	—	—	—	•

× modeled by the user modeling component

• representable in the user model but not put to use

— not modeled

[1] Envisaged.

[2] According to Hoeppner & Morik (1983), all beliefs of the system about the user are to be regarded as mutual beliefs.

[3] Mutual beliefs only.

[4] Martins only talks about "belief spaces"; some of the formulas contained therein may be more profitably regarded as descriptions of an agent's goals.

8. Discussion

The belief and goal classification presented in this chapter should be primarily regarded as having a more *pre-theoretical* character. A theoretically satisfying classification would have to prove that the belief and goal types discerned are both *necessary* and *sufficient* for user modeling in dialog systems. Evidence for their necessity has been given by demonstrating where they occur in current user modeling components, or could occur in future systems. A proof for the sufficiency of the proposed taxonomy (i.e. a proof that there is no other belief or goal type apart from (1.1.1.) – (4.2.5.)), however, would presuppose a deeper understanding of belief and goal formation processes in person-to-person communication than we currently have.

Of course, the above taxonomy can be refined by also taking into account *additional classification criteria.* In the above belief and goal descriptions, for example, the *content* of the beliefs and goals has been totally ignored (i.e. the structure of p has not been further analyzed). However, there is certainly a difference between believing, say, general vs. individual facts (cf. (3) – (7)). Rich [RICH79a, RICH79b], for instance, has proposed that one discern whether an entry in the user model represents a highly specific piece of information or longer-term, more general information. Other possible dimensions in terms of which p can be classified might be suggested by classifications of the underlying representation of the facts described by p. Different representational instruments are employed when p contains quantifiers, disjunction, conjunction, etc. More on this can be found in the literature on belief and goal representation (cf. [HALP85a, KOBS85a, HALP86, WAHL*] for surveys).

The proposed taxonomy imposes demands on representational schemes for beliefs and goals in that all of the above belief and goal types should be expressible in more general representational systems. As a matter of fact, formal representation requires rendering the structure of the above belief and goal types with even more precision. For example, the above-mentioned analysis of what it means to know the x such that p(x) was initiated when inconsistencies in superficial *ad hoc* formalizations of this belief type were detected. One may suppose that, in the long run, we may knock over the ladder after having climbed it and replace the familiar and cherished belief and goal talk by more precise representational notions. Also, an intermediate "epistemological" level as, for example, possible-world theory or that proposed in [KOBS85c] might be an interesting alternative. At the moment, however, there is little consensus as to "the" correct belief and goal representation, and the explanatory power and compactness of linguistic belief descriptions is still so great that this step does not yet seem advisable.

9. References

[AIRE83], [AIRE84], [ALLE80], [ALLE83a], [ALLE83b], [APPE85], [BARN86], [BURG78], [CARB83], [CARB85b], [CARB*], [CHIN86], [CHIN*], [CLHH81], [CLHH83], [COHE79], [DEDE86], [FINI86], [FINI*], [GEOR83], [HAAS86], [HALP85a], [HALP85b], [HALP86],

[HOEP83a], [HORN78], [JOSH82], [KAPD69], [KAPS79], [KAPS83], [KOBS84], [KOBS85a], [KOBS85c], [KOBS85d], [KOBS86a], [KONO80], [KONO81], [KRAU83], [LEHD84], [LEWI69], [MARS83], [MOOR80], [MOOR82], [MORI85a], [MORI85b], [MORI*], [NADA83], [PART70], [PERR81], [POWE79], [QUIN56], [RAPA84], [REIS85], [RICH79a], [RICH79b], [ROSE82], [ROSE85], [SCHI72], [SEAR69], [SLEE82c], [SLEE85], [SMIB82], [SPAR84], [SPAR*], [STIC83], [WALL84], [WAHL*], [WERN88], [WILE86]

Part II

Building User Models

Introduction

The recognition of relevant facts about the user, including most prominently his/her beliefs, background knowledge, goals and plans, is a crucial research issue in the field of user modeling. The effectiveness of a user modeling component in enhancing a system's cooperativity essentially hinges on its ability to draw correct assumptions about the user in the course of the dialog, and to decide which of them are relevant to the dialog purpose at hand.

As can be seen from [WAHL*], a number of construction techniques have been developed in the field of user modeling (possible limitations of which are discussed in [SPAR*] and [KASS*]). The chapters in this part give a detailed account of how such techniques have been put to use in a number of user modeling components developed for various application domains. In addition, an outline is presented as to how the acquired assumptionsassumptions, about the user are exploited by the system. The focus, however, lies on recognition aspects.

David Chin is concerned with the development of KNOME, the user modeling component of the UNIX Consultant (UC) system. UC is an online natural-language advisory system for the UNIX operating system (a related application domain is presented in the work of [QUIL*] which is summarized below). Users can ask UC how to do things in UNIX, request definitions of UNIX terminology, and obtain help debugging problems with command usage. In its interaction with the user, UC should take his/her level of expertise into account.

KNOME's user model contains two sorts of assumptions about the user's knowledge concerning UNIX, namely the expertise stereotype probably applying to the user, and additional individual assumptions about him/her. In the KNOME approach, four expertise stereotypes (novice, beginner, intermediate, expert) are discerned along which users are classified, and four difficulty classes exist for the classification of UNIX commands, command formats, terminology and other relevant information. Assumptions are associated with each expertise stereotype concerning the plausibility of a user belonging to a certain stereotype knowing UNIX-related information of a certain difficulty level. Chin thus uses a *double* stereotype approach (cf. [RICH*]).

The individual part of the user model contains assumptions about the user's UNIX knowledge that were directly inferred from his/her natural language input. Chin discerns six deduction types for inferring what the user probably knows or does not know (including, for instance, the rules 'user mentions the concept ?x → user knows the concept ?x' and 'user wants to know ?x → user does not

know ?x'). In the early stages of the dialog, the inferred assumptions are used for determining the plausibility that a given stereotype applies to the user, and, after enough evidence has been collected (typically after three interchanges), they are used for making the ultimate decision on the most plausible stereotype. As soon as this decision has been made, new individual assumptions may only override the stereotype, but not cause its replacement by some other. Also, individual assumptions are then explicitly represented only when different from the stereotype.

The assumptions which are represented in KNOME's user model are exploited by various UC components. They are used, for example, in determining the correct interpretation of a (possibly indirect) speech act, for expressing information in terms of only such concepts with which the user is already familiar, or for pruning away those parts of an answer that the user already knows. "Meta-knowledge" about whether the system knows all options of a UNIX command (i.e., local closed world assumptions) can be used for correcting users' inquiries regarding options unknown to the system.

Alex Quilici is also involved with the development of a help system for UNIX. His envisaged system should allow the user to report problems which s/he has had in interaction with UNIX, and provide cooperative advice. Quilici concentrates on the recognition of misconceptions about UNIX inferable from these reports, and on the selection of appropriate responses. In particular, he considers misconceptions involving plan applicability conditions (whether a particular plan should be used to achieve a goal), enablements (whether a particular state must exist before a plan can achieve a goal), and effects (whether a state will exist as a result of a plan execution).

The basic knowledge representation scheme in Quilici's model is a plan representation containing action operators and associated applicability conditions, enablements, and effects. Various forms of knowledge are expressed within this scheme:

1. the system's knowledge about preconditions and effects of UNIX commands, including knowledge about the effect of command usage in the case of unfulfilled preconditions;
2. the user's beliefs and goals as inferred from his/her input;
3. a number of beliefs about UNIX commands which the system does not hold, particularly frequent misconceptions of the user (i.e., some sort of "bug catalog" as used in the field of computer-aided instruction – cf. [KASS*]); this knowledge is stored along with (1);
4. an uninstantiated user belief pattern and a number of so-called 'explanations' for each type of user misconception discerned (namely misconceptions concerning command applicability, enablements and effects); each explanation consists of a number of uninstantiated system belief patterns which together "contradict" the user belief pattern (e.g. the system belief patterns 'Ap causes So' and 'Ao causes Sp' may explain why the system does not share a user belief of the form 'action Ap causes state Sp').

Quilici's model assumes that the user's natural-language report about his/her problems in the use of UNIX has already been analyzed, and that a number of user beliefs and goals have been recognized and entered into the user model. For each

user belief, the system first determines whether it shares it by searching its knowledge base for the belief, or whether it is a common misconception by searching the misconception catalog for its negation. If neither search is successful, the system assumes that the user's belief is wrong (see [CHIN*], however, for a critique of this so-called 'closed world assumption').

When an incorrect belief is detected in the misconception catalog, the system can immediately correct the user. Otherwise, the misconception type is determined by matching the user belief patterns with the incorrect belief. The explanations associated with the determined misconception type are then tested as to whether all their system belief patterns (which became partly instantiated in the first step) have a match in the system's knowledge base. When an explanation matches completely, a divergence between system knowledge and user beliefs has been recognized about which the user can be informed.

The contribution by **Sandra Carberry** is concerned with the recognition of users' plans in an ongoing dialog. In her model, a number of domain-independent heuristics are used for inferring candidate goals and plans from user statements (these heuristics are extensions of inference rules proposed by [ALLE80] and [ALLE83]). Focusing strategies are then employed for selecting the candidate goals and plans most related to the current dialog context and for integrating them into a model of the user's actual plan. These strategies are based on recent research on dialog organization and focusing done, for example, by [MCKE85a, MCKE85b, GROS86].

Even before all its details have been determined, the assumed user plan can be exploited for generating expectations about dialog contributions of the user. Carberry's model uses this potential for handling two forms of linguistically problematic dialog contributions, namely pragmatically ill-formed sentences and intersentential ellipsis (only the former is described in this contribution, however). Pragmatically ill-formed user sentences, in her definition, are syntactically and semantically correct, but do not fit the system's model about the domain. Carberry discusses pragmatic ill-formedness resulting from attribute confusions and missing specifications in a user's question, and proposes repair strategies based on the expectations from the user model.

Up till now, nearly all plan recognition models have assumed that the system's model of the user's plan is perhaps incomplete, but never wrong. In a final section, Carberry discusses how her analysis could be extended to be able to detect clues indicating disparities between the system's model of the user and the user's actual goals and plans, reasoning on the system's model and the system's domain knowledge to form hypotheses as to the source of these disparities, responding to the user and negotiating with him/her to isolate the errors, and appropriately repairing the user model.

The work of **Jill Fain Lehman** and **Jaime Carbonell** demonstrates that the field of user modeling cannot confine itself to models which only include assumptions about "objective facts" concerning the user, or reconstruct his/her beliefs, goals and plans (of which the user is probably aware). Instead, other kinds of cognitive competence (which are possibly unconscious to the user) must sometimes also be reconstructed by the system to enable it to adapt to the user in a cooperative way.

An example of such a system is the CHAMP parser for natural language that adapts to the idiosyncratic linguistic usage patterns of a user and stores them in a user model. Its central knowledge base consists of a so-called "kernel grammar", which comprises a lexicon, case-frame concept definitions, default semantic values and syntactic forms that are present prior to interaction with any user. The latter constitute a sort of ANY-PERSON stereotype (cf. [RICH*]) which contains the grammatical regularities in the input of all users. If a user's input cannot be parsed correctly because it is ungrammatical according to the syntactic forms currently known, recovery strategies are applied in a least-deviant-first manner both to letters in input words (this amounts to spelling correction) and to sentence constituents. The strategies used consist of a combination of four basic operations, namely insertion, deletion, substitution, and transposition. The parsing process is split for each recovery strategy that applies. If the application of a small number (typically not more than two) of such operations results in a successful parse, the user is asked to confirm the system's interpretation. If the interpretation is correct, the new syntactic form is subjected to a conservative generalization process, and then added to the user model.

The practicality of the CHAMP approach essentially hinges on the assumption that users' idiosyncratic language patterns vary a great deal (otherwise the kernel grammar could simply be extended to cover all users) and that the patterns exhibited by an individual user are bounded, given a specific task at hand. Both hypotheses were confirmed in a number of hidden-operator experiments carried out by Fain Lehman. In addition, counter-evidence was obtained through control experiments against the assumption that users would naturally adapt to the system's limitations faster than the system can usefully adapt to his/her idiosyncrasies.

4 KNOME: Modeling What the User Knows in UC [1]

David N. Chin

Abstract

KNOME is the user modeling component of UC, a natural language consultation system for the UNIX operating system. During the course of an interactive session with a user, KNOME infers the user's level of expertise from the dialog and maintains a model of the user's knowledge of the UNIX domain. KNOME's model of the user makes use of a *double-stereotype* system in which one set of stereotypes represents the user's expertise and another represents the difficulty level of the information. KNOME is used in UC to help disambiguate the user's statements, avoid telling the user something that the user already knows, take advantage of prior user knowledge in presenting new information, and detect situations where the user lacks pertinent facts or where the user has a misconception. UC also models its own knowledge of UNIX with *meta-knowledge* (explicit facts about the limitations of the system's own knowledge base), which is used to help in correcting user misconceptions.

1. Introduction

UC, the UNIX Consultant [WILE84, WILE86], is a natural language computer consultant system under development at the University of California at Berkeley. UC advises users on using the UNIX[2] operating system. Users can ask UC how to do things in UNIX, get definitions of UNIX terminology, and get help debugging problems in using commands.

To better accommodate a greater range of potential users, UC contains an explicit user modeling component called KNOME (KNOwledge Model of Expertise). Some of the early results of this research were reported in [CHIN86]. KNOME contains information about what users know. Using such information, UC can:

---

[1] This research was sponsored in part by the Defense Advanced Research Projects Agency (DoD), ARPA order No. 4871, monitored by the Space and Naval Warfare Systems Command under contract N00039-84-C-0089, and by the Office of Naval Research under contract N00014-80-C-0732.

[2] UNIX is a trademark of Bell Laboratories

1. avoid telling the user something that the user already knows
2. tailor explanations to the user's level of understanding
3. utilize the user's background knowledge in interpreting what the user says

KNOME is also helpful in detecting cases where the user's knowledge is incorrect (misconceptions) and cases where the user lacks needed information. In such cases, *UCEgo* [CHIN87], the component of UC that implements an intelligent agent for UC, can take the initiative and try to correct the user's misconception, or try to inform the user of the needed information.

1.1. Some UC Examples

To see how KNOME can infer a user's level of expertise from a conversation with the user, and how this information can be used advantageously by UC in helping the user, consider the two UC sessions shown in Figure 1 and Figure 2.

Welcome to UC (Unix Consultant) version 3.23
To a '#' prompt, please type in your questions
about the UNIX file system in English.
To leave, just type '^D' or '(exit)'.

Hi.
How can I help you?

How can I delete a file?

Use rm.
For example, to delete the file named foo, type 'rm foo'.

What does ruptime do?

ruptime is used to list the uptime of all machines on the network,
list the number of all users on them, and list their load average.

Figure 1. UC session with User1

The user of the first session, User1, is apparently a novice who knows very little about UNIX. After all, User1 does not even know something as simple as how to delete a file (the subject of User1's first question). On the other hand, the user in the second session, User2, knows what an inode is, a relatively complex UNIX concept. So User2 is probably a much more advanced user of UNIX than User1. KNOME uses this type of evidence about particular facts that users know or do not know in order to deduce their level of expertise.

Based on the different levels of expertise of the two users as deduced by KNOME, UC is able to provide qualitatively different answers to the two users.

```
Welcome to UC (Unix Consultant) version 3.23
To a '#' prompt, please type in your questions
about the UNIX file system in English.
To leave, just type '^D' or '(exit)'.

Hi.
How can I help you?

# How can I find out the inode of a file?

Use ls -i.

# What does ruptime do?

ruptime is like uptime, except ruptime is for all machines on the
network.
```

Figure 2. UC session with User2

In the first answer to User1, KNOME gives an example of the format of the rm command (simply "rm" followed by the name of the file to be deleted). UC gives an example of the format to User1, since a user as unsophisticated as User1 probably would not know the format of rm. On the other hand, User2 is a more advanced user who undoubtedly already knows the format of ls. Hence UC is able to give a very concise answer to User2's first query, "Use ls -i," that does not include an example of the format of ls. This illustrates how KNOME is used to avoid telling the user something that the user already knows.

The second query of both users is the same. For the more advanced user, UC was able to explain the ruptime command in terms of a similar command, uptime, that the more advanced user presumably already knows. Such a simile would not work for User1, since a novice user probably would not know the uptime command, and so could not exploit the simile to understand ruptime. KNOME's modeling of the user's knowledge allows UC to choose whether or not to use different forms of presentation like similes based on whether or not the user will understand them.

1.2. Problems in Modeling

There are many difficulties in building and maintaining a model of what the user knows. The system must first determine what the user knows, and then represent this information internally. To determine the user's knowledge of the domain, the system could quiz the user exhaustively. However, for most domains, such a quiz would be extremely time consuming and most users would not consent to such extensive quizzing. Fortunately, users tend to learn domain knowledge in a predictable order, so the system can make additional inferences about what a user is likely to know or not know based on a partial user model. That is, the user modeling system should be able to *predict* a user's knowledge based on partial information about

what the user knows or does not know. Exactly which predictions/inferences are allowable is problematic. This is the key problem for any user modeling system that wishes to take advantage of the order in which users typically acquire knowledge in a domain.

Besides the basic question of which inferences are allowable, there is also the problem of uncertainty. Making inferences about the user based on limited prior knowledge about the user is a form of default reasoning. Such default reasoning is not always valid, so the system needs to be able to handle the fact that such inferences are uncertain. Usually this requires that the user modeler be able to judge the certainty of its default inferences. Also, as the system obtains more concrete information, it must be able to update its user model. The new information may contradict previous predictions, so maintenance is not simple.

Another problem concerns the organization of the inferences. Many inferences tend to be grouped together, because people tend to go through distinct stages of learning (see [KAYD85]). People at a particular stage of learning tend to have similar knowledge. Once a system has determined its user's level, it can assume that this user knows about as much as other users at that level. A good user modeling system needs to group such inferences together in a simple and efficient manner.

Another problem in user modeling is how to represent the model internally. The simplest method is the *overlay* model [CARR77] wherein the user's knowledge is represented as a subset of the system's knowledge. However, a simple overlay model cannot predict what a user might know based on partial information. An overlay model does not represent the order in which users typically learn information in a domain. What is needed is a model that represents such ordering information in a manner useful for prediction of what a user might know based on partial information.

1.3. Other User Models

User models have appeared frequently in Intelligent Tutoring Systems (e.g. [BURT76, SLEE82c]). [KASS*] provides a survey of user models in ITS. Typically, ITS use some variant of the simple overlay model [CARR77]. Overlay models encode the user's knowledge as a subset of the system's expert knowledge base. One variant is to encode users as a collection of differences from the system's built-in expert. Differences may include a lack of user knowledge relative to an expert. This is frequently augmented by a library of buggy procedures/rules that are common among users (e.g. [BROW78, STEV79, SLEE81, JOHN84, REIS85, ANDJ85b]). These types of user models are not designed to *predict* a user's knowledge based on partial information about what the user knows or does not know. This sort of default inference is not so appropriate for ITS where the system needs to keep an accurate user model. On the other hand, systems usually do not interact with a single user for as much time as tutoring systems, so they do not have the time to build a complete user model. In such cases, default reasoning is needed. Therefore, consultation systems need a different type of user model than is commonly found in ITS.

An exception is the hierarchical overlay model of USCSH [MATT86] which does some limited prediction. USCSH predicts the user's proficiency for parent nodes in the hierarchy by taking a weighted sum of the proficiency ratings of the children nodes. This approach allows prediction of highly related topics, but does not extend to prediction of the user's knowledge of less related topics. For example, such a system could predict that a user has high proficiency in process monitoring when the system knows that the user has high proficiency in using the ps, who and kill commands. However such a system could not predict the proficiency of the user in an unrelated topic, such as using the cat command, based on the system's knowledge of the user's proficiency in process monitoring commands.

Other user models have been built to model a number of aspects of the user (see [WAHL*] for a survey). For example, Grundy [RICH79b, RICH83, RICH*] modeled user personality traits. Grundy used user *stereotypes* to group default inferences about user traits, then used these traits for suggesting books to the user for reading. To determine which stereotypes were applicable to a particular user, Grundy asked the user for a self description at the start of a session. Individual phrases (e.g. self descriptions such as athletic or feminist) would *trigger* stereotypes that were likely to apply to that user. Such an approach worked well for modeling personality traits, but does not carry over to modeling domain knowledge. A user's self description as a "beginner" often will not coincide with the system's idea of a "beginner." Also, stereotype triggers cannot be used to make inferences such as inferring that the user does not know something when the user asks about it. Stereotype triggers are not sufficient for inferring what users know.

Another use of user models is for representing user preferences. The hotel reservation system of HAM-ANS [HOEP84b, MORI*] and the Real Estate Agent [MORI85b] chose different hotel rooms or apartments based on users' needs and preferences. These systems expect users to mention at the start of their session their requirements and preferences for a room/apartment. Such an approach works well in domains where users tend to specify their needs and preferences at the start of a session. However this is not the case in consultation domains where users are unlikely to mention their level of expertise in the domain at the start of a session.

Many systems including UC have modeled the user's plans and goals (e.g. [ALLE80, CARB83, CARB*, LITM84, JOHN84, WILE86]) Such systems infer the user's plans and goals by matching the user's actions to steps of prestored plans. This type of inference works well for detecting the user's plans, but does not extend to determining what the user does or does not know.

2. Internal Representation of Users

KNOME represents what UC believes users know about UNIX. In this sense, KNOME's model of the user is not necessarily meant to accurately model actual users, but is meant to conform to the way human UNIX consultants model their clients. So in developing KNOME's user models, no attempt was made to psychologically profile what different users know. Instead, former UNIX consultants (UC's implementors) were informally surveyed to determine how they viewed users.

2.1. Double-Stereotypes

Humans use categorization to organize inferences (see[ROSC77,ROSC78,ROSC83]). These categories serve as reference points or prototypes for judging objects. Once an object has been identified as belonging to a particular category, default inferences about the object can be made based on the object's membership in the category.

An approach similar to human categorization is used in KNOME to organize inferences concerning users. In KNOME, users are grouped according to their level of expertise in using UNIX. Each group is represented by a prototype-style category. There are four such categories in UC: *novice, beginner, intermediate,* and *expert.* Each of the categories represents an increasing mastery of UNIX information.

Individual users are represented as members of one of these categories and inherit properties from their category. Specific information about individual users overrides inherited information. These user categories are also termed stereotypes after the stereotypes of Grundy [RICH79b, RICH83, RICH*], which pioneered categorization of users in computer programs. However, unlike the stereotypes in Grundy, which consist of collections of attributes, the stereotypes in KNOME are implemented as categories within the KODIAK representation system [WILE87]. As such, KNOME's stereotypes are integrated into KODIAK's multiple inheritance hierarchy and its default inheritance mechanisms.

Besides categorizing users, KNOME also categorizes information into levels of difficulty. Commands, command-formats, terminology, and other relevant information are categorized according to their level of difficulty. These stereotype categories include *simple, mundane,* and *complex.* Information is grouped into categories based on their typical location on the learning curve, in other words, based on when the typical user would learn the information. For example, some simple concepts include the rm, ls, and cat commands, the technical term "file," and the simple file command-format (the name of the command followed by the name of the file to be operated upon). These simple concepts are learned early in the experience of a typical user. Somewhat more advanced are mundane level concepts. Some examples are the vi, diff and spell commands, the technical term "working directory," and the -l option of ls. Examples of complex concepts are the grep, chmod, and tset commands, the term "inode," and the fact that write permission on the containing directory is a precondition for using the rm command for deleting a file. Complex concepts are learned relatively late by the typical user.

In addition to the three previously mentioned levels of difficulty, KNOME has the category *esoteric* that includes information that is not typically learned at any particular stage of experience. Such concepts are learned only when users have special needs. Some may be familiar to beginners, yet unfamiliar to many experts. An example of an esoteric command is spice, a program used for electronic circuit simulation. Only people who need to perform circuit simulations would know how to use spice. This group would likely include beginning users as well as intermediate and expert users. On the other hand, there are many expert UNIX users who do not know about spice, because they have never needed to perform semiconductor circuit simulations.

Once information has been classified into levels of difficulty, inferences based on user stereotypes can easily be represented as relations between the various user

stereotypes and difficulty levels. For example, Figure 3 shows a graphical KODIAK representation of one such relation. This Figure shows the KODIAK representation of the fact that intermediate users know most facts that have the difficulty level of mundane. The predicate of MOST is used, because there is uncertainty as to whether any particular intermediate user will know any particular mundane level fact. However, any intermediate level user will know most mundane level facts. Table 1 summarizes the relation between different user stereotypes and different knowledge difficulty levels.

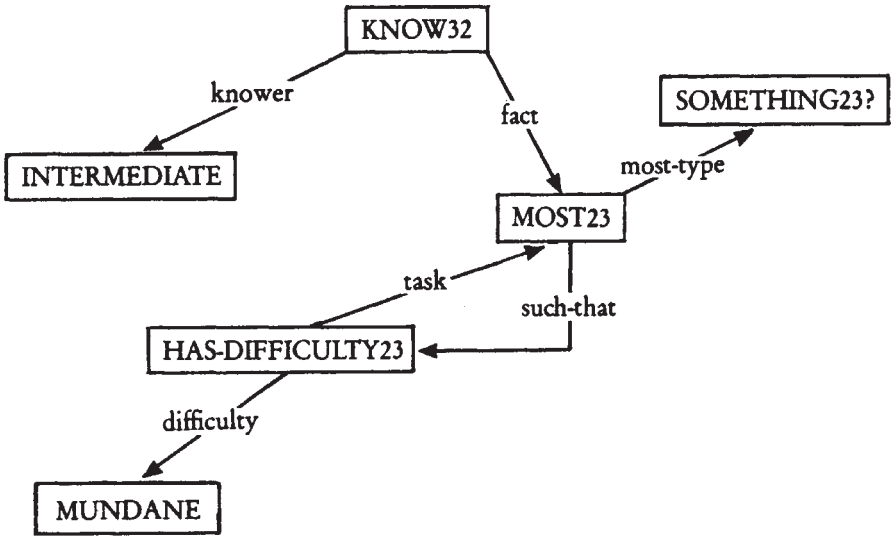


Figure 3. KODIAK representation of: intermediates know most mundane facts

Table 1. Relation between user stereotypes and knowledge difficulty levels

User stereotype	Knowledge difficulty level			
	simple	mundane	complex	esoteric
expert	ALL	ALL	MOST	—
intermediate	ALL	MOST	AFEW	—
beginner	MOST	AFEW	NONE	—
novice	AFEW	NONE	NONE	NONE

The use of stereotype categories for both users and information is an example of a *double-stereotype* system. Single stereotype systems like Grundy used stereotypes for users (e.g., FEMINIST, INTELLECTUAL, SPORTS-PERSON) but not for information (i.e., Grundy did not have stereotypes for books such as GOTHIC-ROMANCE, DETECTIVE-STORY, or SCIENCE-FICTION). KNOME is the first user modeling system to utilize double-stereotypes.

The use of stereotypes allows KNOME to *predict* what users are likely to know or not know based on a partial user model. Once KNOME has accumulated a small

number of facts (typically 3)[3] about what users do or do not know during the course of a UC session, it can infer the user's level of expertise (see Section 3). Then based on the user's level of expertise, KNOME can predict whether or not the user is likely to know other facts. The small set of facts that allows KNOME to infer the user's level of expertise constitutes a partial user model, and the inference that the user belongs to a particular stereotype represents a set of default inferences that are made on the basis of the partial user model.

The use of stereotypes allows KNOME to efficiently group these default inferences together. A system that did not use stereotypes would need to represent individually every partial user model and all the default inferences from that partial user model. Since a partial user model is simply a set of facts about a user, there is a combinatorically large number of partial user models. So, representing all such partial user models and their default inferences would be extremely inefficient. However many partial user models have similar default inferences. These can be grouped together efficiently using user stereotypes. A double-stereotype system is even more efficient, since it allows the system to encode for each fact only its level of difficulty rather than having to encode its relationship to every user stereotype. The rest of a double-stereotype system can then be encoded in a small number of statements relating the two levels of stereotypes. This relationship in KNOME between user stereotypes and knowledge difficulty levels is summarized in Table 1.

KNOME currently has only one double-stereotype range for users. This range represents users' knowledge of UNIX file manipulation commands, UNIX information gathering commands, and UNIX concepts and terminology. This corresponds to UC's domain of expertise within UNIX. If UC were extended to cover more of UNIX or even other operating systems, then KNOME would need more ranges of double-stereotypes. For example, if UC also covered usage of the vi and emacs editors, then KNOME would need separate expertise ranges for each of these domains. KNOME would also need to encode how the user's level of expertise in one domain might be used to predict the user's level of expertise in another domain. So, an expert in using vi would undoubtedly have a high level of expertise in manipulating the UNIX file system. On the other hand, a high level of expertise in using emacs does not necessarily indicate a high level of expertise in using UNIX, since the emacs editor is commonly found in other operating systems.

2.2. Modeling Individual Users

Individual users usually do not completely fit any one stereotype. As a result, a user model that uses stereotypes must also encode how individual users differ from the stereotypes. KNOME stores specific information about what individual users know and do not know. Such information is stored as a collection of propositions about what a user knows or does not know. These propositions are represented

[3] The actual number varies, because facts may mutually reinforce KNOME's evaluation of the user's level of expertise, in which case fewer facts will be needed; or facts may provide contradictory evidence, in which case more facts will be needed to evaluate the user's level of expertise.

using KODIAK. In reasoning about what a user knows, this collection of individual facts is checked before resorting to reasoning from the user's stereotype category.

To avoid storing too many individual facts about what users know, only those facts that cannot be inferred with high likelihood from a user's category are stored. For example, the fact that a particular intermediate user knows the rm command (a simple command) does not need to be stored explicitly, since it is directly inferable from the fact that intermediate users know *all* simple facts. Also, the fact that a particular beginner knows the rm command does not need to be stored. This is because beginners know *most* simple facts and hence it is likely that any particular beginner would know rm. On the other hand, the fact that a particular novice knows the rm command does need to be stored explicitly, since that is not inferable with high likelihood. Because novice users know just *a few* simple facts, it is unlikely that any particular novice would know rm.

2.3. Types of Knowing

In UC, two senses of "know" are used. The first is the classical sense of know: the knower believes that some proposition P is true, and P is true. An example of this sense of knowing is: " the user knows that the cd command is used to change the current working directory." There is also another sense of know that is found in UC. This sense of know is closer to the colloquial usage wherein knowing an object means being familiar with that object. However, in UC this vague usage is made precise. For UC, knowing a command such as the UNIX-RM-COMMAND implies knowing that there exists a command with the name rm, knowing the main purposes of this command, knowing the format of the command, knowing the main effects of the command, and knowing the main preconditions of the command.

The main purposes of a command are represented in UC using the *planfor* [CHIN83, WILE86] relation, which relates a goal and the plan that is typically used to satisfy that goal. The planfors of a plan are different from the effects of the plan, since not all of a plan's effects can be considered goals that the plan is typically used to satisfy. All goals in the planfors of a command are also effects of the command, but some side effects of the command may not be part of its planfors. For instance, calling the game program rogue with a file argument has a side effect that is not found in the planfors of rogue. When rogue is given a file argument that is not a rogue-format save file, then rogue arbitrarily removes the file. This is a side effect of the rogue program, but it is not in any planfors, since users do not typically run rogue in order to remove files.

2.4. Dealing with Uncertainty

In any user model, the inferences that the model makes about the user contain some degree of uncertainty. In KNOME, an attempt was made to avoid the use of numeric representations of uncertainty, because their use tends to lead systems to overestimate the accuracy of their ratings of uncertainty. So, UC uses a fixed number of simple rating levels instead: LIKELY, UNLIKELY, VERY-LIKELY, VERY-UNLIKELY, SOMEWHAT-LIKELY, SOMEWHAT-UNLIKELY, TRUE, FALSE,

and UNCERTAIN (cf. fuzzy logic [ZADE65]). KNOME also uses predicates such as AFEW, MOST, ALL, and NONE (cf. [ZADE82]). Such predicates and ratings are used by KNOME to express the certainty of its inferences. For example, when KNOME is asked whether a particular user knows some particular fact, it will return either TRUE, LIKELY, UNLIKELY, FALSE, or UNCERTAIN. TRUE and FALSE indicate that KNOME has complete confidence. LIKELY and UNLIKELY indicate some uncertainty. An UNCERTAIN result indicates that KNOME does not have enough information to guess whether the user does or does not know that fact.

To deduce whether a particular user knows some particular fact, KNOME uses a multi-step inference process. First, KNOME checks the individual user model for that user. If the individual user model records that this user does or does not know this fact, then the answer is, respectively, TRUE or FALSE. If there is no specific information in the user's individual user model, KNOME next checks the stereotype category of the user. If users of the user's stereotype do or do not know that fact, then the answer is again TRUE or FALSE, respectively. If these checks fail, KNOME resorts to inference based on the difficulty level of the fact. If the user's stereotype knows ALL facts of that difficulty level, then the answer is TRUE; if the stereotype knows MOST facts of that difficulty, then the answer is LIKELY; and if the stereotype knows AFEW facts of that difficulty, then the answer is UNLIKELY. Finally, if this process fails due to lack of information, then the answer is UNCERTAIN. This algorithm is summarized in the flow chart of Figure 4.

3. Deducing the User's Level of Expertise

One of the main problems in user modeling is how to build up a model of a particular user. In a user modeling system based on stereotypes, this means determining which stereotype(s) best fit the current user and how that user differs from the stereotypes. In UC, this means that KNOME must determine the user's level of expertise in using UNIX.

3.1. Other Model Acquisition Systems

Various approaches to building up models of users have been used. Some systems such as the Real Estate Agent [MORI85b] expect users to mention enough self-characteristics at the start of a session so that the system can build up the user model. This works for situations such as a real estate agent interchange with a client where the client is likely to mention what he/she needs/wants at the beginning of a dialog. However, this does not extend to other consultation situations where the user usually will not mention what he/she knows at the start of a session. In such consultation situations the user typically begins by describing his/her problem to the consultant.

Another approach is to ask the user for a self-description at the start of a session as in Grundy [RICH79b] This is a reasonable approach for judging the user's

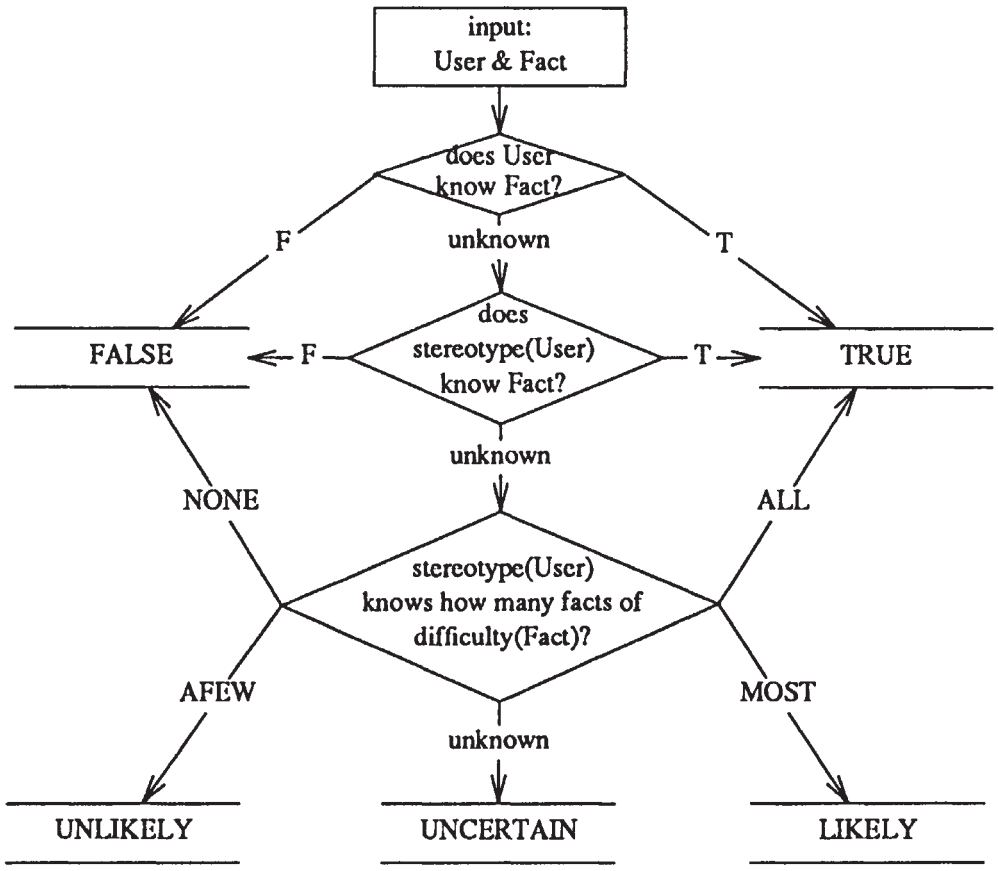


Figure 4. Algorithm for determining whether User knows Fact

personality traits, but does not work well for determining what the user knows. For judging the user's level of expertise, this approach is inaccurate, since the user's concept of different levels of expertise like "intermediate" may differ considerably from the system's concept or from other users' concept of an "intermediate." Also, this requirement is not very practical for occasional users, since this increases the overhead of using the system.

A third approach is to quiz users at the start of a session so that the system may estimate the user's expertise. For example, the MAP system [MACM83] helps a user to specify a correct user model of how that user interacts with an operating system. Such an approach is extremely time-consuming. For tasks like intelligent operating system interfaces for which MAP was designed, such a protracted process is somewhat more acceptable, since the user will realize its benefits over a long period of time. However for interacting with a consultation system like UC where typical sessions are short, such a process would be too time-consuming. The casual user who only needed the answer to a single question would be better off without a user model rather than having to put up with such a lengthy process.

The most common approach in ITS (Intelligent Tutoring Systems) is to compare the user's performance with what a built-in expert would do. Differences in performance can then be attributed to either a lack of user knowledge or differences

in user knowledge such as improper procedures or rules. However such an approach is only applicable if the system can "look over the shoulder" of the user. Consultation systems often cannot observe the user at work. Even when this is possible, consultation systems should not be expected to constantly observe the user, since human consultants do not do so.

To properly emulate a human consultant, the system should be able to assess the expertise level of the user while talking to the user. None of the previously mentioned approaches model this ability, which is a difficult problem in natural language understanding. One system that has addressed this problem is the VIE-DPM user modeling system [KOBS85d]. However, VIE-DPM uses only a syntactic interpretation of the user's statements to derive the user's beliefs. VIE-DPM makes assumptions about the user's beliefs based on the form of the user's input (a yes/no question, a wh-question, or a command) [KOBS85d, KOBS86a] or the presence of linguistic particles (e.g. "not") in the user's input [KOBS85b]. Such a syntax-only system would not be able to properly interpret phenomena such as indirect speech acts [SEAR69]. For example, "Can you pass the salt?" is a yes/no question, but one should not assume from this question that the speaker does not know whether the listener can pass the salt. Despite such limitations, VIE-DPM is a step in the right direction.

3.2. KNOME's Approach

KNOME uses the approach of inferring the user's level during the course of a session. At the beginning of a session, UC has no information about the user. In such cases, KNOME remains open to the possibility that the user may belong to any stereotype. KNOME maintains a list of candidate stereotypes to which the user may belong. KNOME also rates the likelihood that the user may be a member of any particular stereotype. At the start, the likelihood ratings for membership in any stereotype is neutral, except for the beginner stereotype. The likelihood rating for beginner is initially slightly higher than neutral, since more of UC's users tend to be beginners rather than novices, intermediates, or experts. As more information is gathered about the user during the session, the stereotypes may rise or fall in likelihood.

KNOME deduces the user's level of expertise using a two phase process. First, during the course of the conversation, KNOME deduces individual facts about what the user does or does not know. As such facts are gathered, KNOME combines the evidence to adjust the likelihood ratings for the different levels of expertise. Eventually KNOME identifies the stereotype level of expertise to which the user belongs.

3.3. Collecting Evidence

The first step in the process of deducing the user's level of expertise is to collect evidence on that subject. Typically, such evidence takes the form of individual facts about what the user knows or does not know. Such facts and other forms of evidence can be deduced from what the user says (or does not say) and from an analysis

of the user's goals. KNOME distinguishes six classes of inferences that were found to be useful in UC for determining the user's level of expertise. These inference classes are summarized in Table 2, which lists the inference class and the kinds of inferences that are commonly found in each inference class.

Table 2. Taxonomy of deduction types for inferring what users know

Inference class	Kinds of inference
Claim	user states that user does (not) know ?x → user does (not) know ?x
Goal	user wants to know ?x → user does not know ?x
Usage	user uses ?x → user knows ?x
Background	user mentions his/her background → user knows as much as the stereotype indicated by the background
Query-Reply	system asks if user knows, user replies → user's reply
No-Clarify	system uses new terminology, user does not ask for clarification → user understands terminology

The first class of inferences, Claim inferences, applies when the user claims to know or not know something. The second class, Goal inferences, applies when the system can infer that the user wants to know something. In such cases, the system can also infer that the user must not know whatever is wanted, since if the user already knew it, then the user would not have the goal of knowing it. The third class, Usage inferences, are made when the user properly uses some command or terminology. KNOME can then deduce that the user must know the command or terminology. The fourth class, Background inferences, are made when the user mentions some relevant information about his/her background. The fifth class of inference, Query-Reply, applies when the system queries the user about the user's knowledge and the user replies. The last type of deduction, No-Clarify, is made when the system uses some new terminology that the user may or may not know. If the user does not ask for a clarification of the terminology, then the system can assume that the user probably already knew the terminology. Each of these classes of inference is described in greater detail below.

Claim inferences are concerned with direct statements by the user about what the user does or does not know. Examples of such statements and the inferences that KNOME makes based on the statements include:

I already know about mkdir.
→ the user knows the mkdir command

I don't know how to move a file.
→ the user does not know how to move a file

→ the user does not know that mv is a plan for moving files

→ the user does not know the mv command

In the first example, the user claims to already know the mkdir command. This statement is not as straightforward as it might seem, since the user and the system may not agree on what it means to know a command. A reasonable interpretation of what a user might mean is that the user knows how to use mkdir, knows the main purpose of mkdir and knows the main effects and preconditions of mkdir. This interpretation is the same as what UC means when it encodes that a user knows a command (see Section 2.3).

Of course, it may be that the system should not believe the user's statement about knowing mkdir. It is possible that the user may be confused and does not actually know mkdir. But, there is always doubt about what someone might really know, no matter how strong the evidence. For example, even if the system were to watch the user use the mkdir command successfully, it is possible that the user may have actually wanted to create a file instead of a directory. However, in none of these cases is there any reason to doubt the user. Unless KNOME obtains evidence to the contrary, it assumes that the user really does (or does not) know something when the user makes such a claim.

The second example requires somewhat more complex processing than the first example. In the second example, the user claims not to know how to move a file. Since UC knows that mv is used to move files, KNOME is able to deduce that the user does not know that mv is a plan for moving files. Finally, because the user does not know one of the main purposes of the mv command, KNOME deduces that the user must not know mv.

KNOME only stores information about levels of difficulty for objects and not for descriptions, so only facts about a user knowing (or not knowing) specific objects (e.g. commands, concepts, etc.) can be used in deducing the user's level of expertise. This approach eliminates redundancy, since any object will have many possible descriptions. For example, the user could have referred to mv by "how to rename a file" rather than "how to move a file." So, the fact that the user does not know mv is useful for deducing the user's level of expertise, but the descriptive fact that the user does not know how to move a file cannot be directly used. UC must first determine that this description applies to the mv command before it can be used for deducing the user's level of expertise. In UC, descriptions of the purpose of commands are interpreted by UC's UNIX planner [WILE86]. After UC's UNIX planner has determined the command referred to by the command description, KNOME can use this information in deducing the user's level of expertise.

Goal inferences apply when UC can deduce the user's goals. If UC can infer that the user wants to know something, ?x, then KNOME can deduce that the user does not know ?x and also does not know the answer to ?x. For example, if the user asks "How can I delete a file?" then UC can deduce that the user's goal is to know how to delete a file. From this goal, KNOME can deduce that the user does not know how to delete a file. Then, after UC computes that a plan for deleting a file is to use the rm command, KNOME can infer that the user does not know this planfor relation between deleting files and the rm command. Finally, since the

planfor relation is central to UNIX commands, KNOME can deduce that the user does not know rm in the sense that the user is not familiar with rm (see Section 2.3).

In UC, the initial deduction about the user's goal is made by UC's goal analysis component, PAGAN [WILE86]. PAGAN computes the user's plans and goals from the dialog, and can handle phenomena such as indirect speech acts. In some cases, PAGAN utilizes KNOME's model of the user's knowledge in deducing the user's goals. For example, if the user asks, "Do you know how to compact a file?", then the choice between the direct interpretation (the user wants to know if UC knows how to compact a file) and the indirect interpretation (the user wants to know how to compact a file) depends partly on whether or not KNOME believes the user already knows the compress command. If KNOME does believe that the user is likely to know compress, then that will bias PAGAN toward the direct interpretation that the user is testing UC (not uncommon), and vice-versa.

This process may seem circular, since PAGAN's analysis of the user's goal depends on KNOME, while KNOME's analysis of the user's knowledge depends on PAGAN. However, KNOME's initial evaluation of whether the user knows is only a prediction from a partial user model. This prediction does not determine PAGAN's interpretation, because PAGAN must also take into account other factors in its analysis. One such factor is the frequency of usage of the direct versus indirect interpretations. PAGAN is biased toward adopting the more frequent usage. Another factor is the the context of the dialog. When UC is configured in knowledge acquisition mode as UCTeacher [MARJ85], then the direct interpretation (i.e. the user is testing UC's knowledge), is inherently more likely than the indirect interpretation. KNOME's beliefs about the state of the user's knowledge is only one factor in PAGAN's analysis, which then leads KNOME to its own inference about the state of the user's knowledge.

Besides questions on how to do something, Goal deductions apply to many other kinds of queries such as:

What is a directory?
→ the user wants to know what is a directory (goal)
→ the user does not know what is a directory
→ the user does not know the directory concept

Can you tell me how to find out who is on the system?
→ the user wants to know who is on the system (goal)
→ the user does not know who is on the system

What does ruptime do?
→ the user wants to know the effects of the ruptime command (goal)
→ the user does not know the effects of the ruptime command
→ the user does not know the ruptime command

All of the above queries represent Goal deductions, since PAGAN can deduce that the user wants to know some information. Based on this initial deduction, KNOME can infer that the user does not know the information that is sought.

Usage deductions are made when the user applies some plan/procedure correctly, or uses some specialized terminology correctly. Examples include:

I tried typing "rm foo," but I got the message, "rm: foo not removed."
→ the user knows the rm command

How can I find out the inode of a file?
→ the user knows the inode concept

I tried typing "del foo", but I got the message "del: Command not found."
→ the user knows the DOS del command
→ the user does not know the UNIX rm command

In the first example, KNOME can deduce that the user knows the rm plan. In the second example, KNOME deduces that the user knows the meaning of the term "inode." This is the main type of deduction used in ITS in which the system observes the user at work solving problems. This type of deduction is suggested by [RICH83] for consulting situations.

In general, whenever the user uses specialized terminology correctly, KNOME infers that the user must know the concept denoted by the terminology. In UC, terms include UNIX command names and operating system terminology such as "file protection", "directory," and "inode." Checking that the user used the terms *correctly* is not very sophisticated in KNOME. The first check is simply that the term must be used in an English sentence that UC can parse and understand. However, the fact that the user used a term in an understandable sentence does not automatically mean that the user knows the meaning of the term. For example, if the user asks, "What does rm do?", KNOME should not infer that the user knows the rm command. The same principle applies when the user states, "I don't know the rm command." In these types of sentences, KNOME actually first infers that the user does know rm and then reverses its inference later based on the fact that the user has the goal of knowing the effects of rm, which is a Goal type inference. In cases when KNOME has evidence both that the user knows something and that s/he does not know it, KNOME always assumes that the user doesn't know, since it is always possible that the user just seems to know and in fact doesn't know.

The last example is somewhat more complex than the previous examples. In this case, KNOME first infers that the user knows the DOS del command. Then, since the DOS del command is used to delete files and the user mistakenly tried to use the del command, KNOME can deduce that the user does not know the UNIX command for deleting files, that is, the rm command. In general, whenever the user tries to use a foreign command in UNIX, then KNOME assumes that the user does not know the corresponding UNIX command. This correspondence is found by looking at the goal of the foreign command and finding a UNIX command that is a plan for the same goal.

Background deductions are made when the system learns some relevant fact about the user's background. Examples include:

I am a fourth year computer science graduate student.

→ the user is a member of the 4TH-YEAR-CS-GRAD stereotype

→ the user is LIKELY to be an EXPERT

In my CS2 class, ...

→ the user is a member of the CS2-STUDENT stereotype

→ the user is LIKELY to be a BEGINNER

In the first example, KNOME infers that the user is a member of the 4TH-YEAR-CS-GRAD stereotype. This allows KNOME to further infer that the user is LIKELY to be an expert, because most 4th year CS graduate students are experts. In the second example, UC's Concretion mechanism infers from "my CS2 class" that the user is a CS2-STUDENT. Based on this, KNOME then infers that the user is a member of the CS2-STUDENT stereotype, and so is likely to be a beginner.

To make Background deductions, KNOME uses a collection of rules, one for each stereotype related to the user expertise levels. For example, the rule for CS2-STUDENT is: if the user is a CS2-STUDENT, the the user is LIKELY to be a BEGINNER.

Query-Reply deductions are made after the system asks whether or not the user knows some fact. The user's answer usually provides information about whether or not the user knows the fact. For example, if a system were to ask the user, "Do you know the rcp command?" and the user answers "Yes," then the system can infer that the user knows the rcp command. This type of deduction has not been implemented in KNOME, since the current version of UC does not ask the user whether the user knows particular facts.

No-Clarify deductions show how a system might obtain information based on what the user does *not* say. If the system uses some terminology that the system is not sure of the user knows, then the system can infer that the user does know this terminology provided that the user does not ask for a clarification and the meaning of the terminology is not evident from the context. Such deductions are more complex than the previous ones, since they require that the system keep track of the conversation. This type of deduction has not been implemented in KNOME, since UC tries to avoid the use of terminology unfamiliar to the user.

In KNOME, deductions are made using a rule-based system. The actual rules were shown in Table II, with the exception of the Background type of inference. The Background type inference is actually implemented by a collection of rules. For every type of user background that gives some clue about the user's level of expertise in using UNIX, there is a rule. Since there are potentially many types of user backgrounds that are relevant, there will be a corresponding number of such rules. For example, if the user is a CS2 student, then that is a relevant piece of background information which provides some clue about the user's level of expertise in using UNIX. On the other hand, the facts that the user is a scuba diver, likes blueberry pies, or is a bachelor are irrelevant. The particular Background rules that have been implemented in KNOME are shown in Table 3, which also lists the auxiliary inferences which support the major inferences that were shown in Table 2.

Table 3. Other rules used to infer what users know

user is a fourth year CS graduate student	→ user is likely to be an expert
user is a CS2 student	→ user is likely to be a beginner
user does (not) know ?x, where ?x is a description, and ?y is the referent of ?x	→ user does (not) know ?x
user does (not) know a planfor of command ?x	→ user does (not) know ?x
user does (not) know	→ user does (not) know ?x

Such rules are implemented using *if-detected daemons* [CHIN87], which are daemons that detect specific configurations of knowledge in UC's KODIAK knowledge bases. For example, the first rule shown above would normally fire after PAGAN has inferred that the user wants to know something. Then the action part of the rule allows KNOME to infer that the user does not know some fact.

These classes of deductions were found to be important in the domain of a natural language consultation system. Other domains may provide other clues as to what a user knows. For example, in a domain where the user provides speech input, a system may be able to make deductions about the user's knowledge when the user mispronounces technical words.

3.4. Combining Evidence

After collecting evidence that the user does or does not know certain facts, the system must combine the evidence to determine the user's level of expertise. Facts are particular pieces of information about what the user does or does not know as described in the previous section. Note that a single user statement or query may produce several facts. Each unique fact is treated equally in determining the user's level of expertise. When KNOME deduces that the user does or does not know something more than once (perhaps from different user statements), only the first such fact is used in deducing the user's level of expertise. The rest only reconfirm the first fact, so they do not contribute to determining the user's level of expertise.

At the start of a session, KNOME has very little idea about the level of expertise of the user. KNOME's beliefs about the user's level are encoded as ratings of the likelihood that the user is a member of a candidate stereotype category. As evidence is gathered during the dialog, KNOME continuously adjusts its ratings concerning the likelihood that the user might be a member of any particular stereotype. Eventually, KNOME gathers enough evidence to peg the expertise level of the user. Typically, this takes about three interchanges. After making this decision, KNOME

does not change its estimation of the user's level. This works well in UC where typical sessions are short and it is advantageous to quickly guess the user's level of expertise. A more flexible approach is to keep a running account of likelihoods as in SC-UM [NESS87], a user modeling system that is based on the methodology demonstrated in KNOME. However, a system that does not commit to a particular interpretation of the user's level of expertise cannot avoid the need to store information that can be predicted from the user's stereotype. This becomes inefficient as more is learned about the user.

There are two types of conclusions that may be drawn from any particular fact about what the user knows or does not know. First, the evidence may be enough to eliminate some stereotypes from consideration. An example is when the user does not know a simple fact such as the rm or ls commands. In these cases, KNOME can rule out the possibility that the user may be an intermediate or an expert, since all intermediates and experts know all simple facts (see Table 1). So, if the user does not know a simple command like rm or ls, then the user could not possibly be more advanced than a beginner.

Even when the evidence is not enough to rule out a category, the system can still infer that the category is either more likely or less likely. KNOME does this by increasing or decreasing the likelihood rating of candidate categories.

Likelihood ratings are combined using the following linear scale: FALSE, VERY-UNLIKELY, UNLIKELY, SOMEWHAT-UNLIKELY, UNCERTAIN, SOMEWHAT-LIKELY, LIKELY, VERY-LIKELY, TRUE. The UNCERTAIN rating is neutral. It implies that the system does not believe that membership in that category is either likely or unlikely. Likelihood ratings combine linearly according to the scale. For example, a rating of LIKELY when combined with another LIKELY rating produces a TRUE rating. Also a SOMEWHAT-LIKELY rating combined with a VERY-UNLIKELY rating produces an UNLIKELY rating.

To see why KNOME might change a likelihood rating, consider what happens when KNOME determines that a user does not know some mundane level fact. In this case, KNOME infers that the user cannot be an expert, since experts know all mundane facts. Also, KNOME considers it somewhat less likely that the user might be an intermediate, since intermediates know most mundane level facts. Similarly, the user is considered somewhat more likely to be a beginner, since beginners know a few mundane facts. Finally, since the stereotypical novice does not know any mundane facts, the user is considered more likely to be a novice.

Table 4 shows the basis for these deductions. Given that a user does or does not know a fact of some difficulty level, and given that a particular user stereotype knows none/afew/most/all of the facts of that difficulty level, then KNOME can deduce that the user belongs to that stereotype with the additional likelihood shown in Table 4.

Combining Table 4 with Table 1, yields Tables 5 and 6. Given the particular stereotypes used in KNOME, these two tables show the deductions that are made when KNOME determines that a user does (Table 5) or does not (Table 6) know some fact that has the difficulty level indicated. Deductions consist of eliminating categories (FALSE) or increasing/decreasing the likelihood ratings of categories.

Table 4. Deductions when user does (not) know some fact

Stereotype knows difficulty level	Likelihood (user ∈ stereotype)	
	User does know fact	User does not know fact
NONE	FALSE	LIKELY
AFEW	SOMEWHAT-UNLIKELY	SOMEWHAT-LIKELY
MOST	SOMEWHAT-LIKELY	SOMEWHAT-UNLIKELY
ALL	LIKELY	FALSE

Since these two tables are easily derivable from the previous two, KNOME does not actually store the information of these two tables internally. Table 1 is represented declaratively as a KODIAK network that is accessible to the rest of UC. Only Table 4 is stored internally in KNOME. This table is used when needed to derive the likelihood rating differences shown in Tables 5 and 6.

Table 5. Deduction when user knows some fact

User stereotype	Difficulty level of fact			
	simple	mundane	complex	esoteric
Novice	SOMEWHAT UNLIKELY	FALSE	FALSE	FALSE
Beginner	SOMEWHAT-LIKELY	SOMEWHAT-UNLIKELY	FALSE	—
Intermediate	LIKELY	SOMEWHAT-LIKELY	SOMEWHAT-UNLIKELY	—
Expert	LIKELY	LIKELY	SOMEWHAT-LIKELY	—

When the likelihood rating of a stereotype reaches TRUE, it is selected as the user's category. The user's category can also be deduced by elimination. Candidate stereotypes are eliminated when their likelihood drops to FALSE. When all but one candidate have been eliminated, the remaining candidate is selected. During the period before the final decision, KNOME works under the provisional assumption that the user belongs to the category with the highest current likelihood. In the case of a tie (i.e. when there are two or more candidate categories with the same highest likelihood rating), the lower level of expertise is chosen to represent the user temporarily. Using the lower level helps prevent KNOME from overestimating the user before KNOME is sure about the user's level of expertise. Underestimating the user's level may result in more verbose descriptions, but does not lead KNOME to

Table 6. Deduction when user does not know some fact

User stereotype	Difficulty level of fact			
	simple	mundane	complex	esoteric
Novice	SOMEWHAT-LIKELY	LIKELY	LIKELY	SOMEWHAT-LIKELY
Beginner	SOMEWHAT-UNLIKELY	SOMEWHAT-LIKELY	LIKELY	—
Intermediate	FALSE	SOMEWHAT-LIKELY	SOMEWHAT-LIKELY	—
Expert	FALSE	FALSE	SOMEWHAT-UNLIKELY	—

leave out information that the user may not know. Also, inferences that are made based on these provisional stereotype assumptions are marked as having greater uncertainty.

3.5. Examples

To see how KNOME deduces a user's level of expertise, consider the following traces from the UC sessions shown in Figures 5 and 6. At the start of either session, UC does not have any information about either user. In such cases, UC starts with a slight bias toward believing that the user is a beginner, because more UC users tend to be beginners. Hence the likelihood rating for the user being a beginner is SOMEWHAT-LIKELY, whereas the likelihood ratings for the user being a novice, intermediate or expert are all UNCERTAIN (the neutral likelihood rating).

In Figure 5, KNOME deduces that the user is a novice, because the user does not know either the more or lpr commands, which are both simple commands, nor the rwho command, which is of mundane difficulty. When KNOME first deduces that the user does not know the more command, KNOME can draw a number of inferences based on the fact that the user does not know a simple command. Since experts and intermediates know all simple commands (see Table 1), KNOME rules out the possibility that the user might be an intermediate or an expert. Also, since beginners know most simple facts, KNOME lowers the likelihood rating for the user being a beginner from SOMEWHAT-LIKELY to UNCERTAIN. Likewise, since novices know only a few simple facts, KNOME raises the likelihood rating for the user being a novice to SOMEWHAT-LIKELY.

When KNOME next determines that the user does not know the simple command, lpr, KNOME makes additional inferences. KNOME raises the likelihood rating of the user as a novice to LIKELY and lowers the likelihood rating of the user as a beginner to SOMEWHAT-UNLIKELY. Finally when KNOME finds that the user does not know the mundane command rwho, the likelihood rating for the

```
Welcome to UC (Unix Consultant) version 3.23
To a '#' prompt, please type in your questions
about the UNIX file system in English.
To leave, just type '^D' or '(exit)'.

Hi.
How can I help you?

# What will display a file on my terminal?

KNOME:  Asserting *USER* does not know ACTION12?
KNOME:  Asserting *USER* does not know PLANFOR40
KNOME:  Asserting *USER* does not know UNIX-MORE-COMMAND0
KNOME:  Since UNIX-MORE-COMMAND0 is a UNIX-MORE-COMMAND,
        asserting *USER* does not know UNIX-MORE-COMMAND
KNOME:  UNIX-MORE-COMMAND has difficulty SIMPLE, so deducing:
KNOME:  ruling out *USER* = INTERMEDIATE
KNOME:  ruling out *USER* = EXPERT
KNOME:  *USER* is SOMEWHAT-LIKELY to be NOVICE
        =) likelihood(*USER* = NOVICE) = SOMEWHAT-LIKELY
KNOME:  *USER* is SOMEWHAT-UNLIKELY to be BEGINNER
        =) likelihood(*USER* = BEGINNER) = UNCERTAIN

Use more.
For example, to display the file named foo, type 'more foo'.

# How can I print a file on the lineprinter?

KNOME:  Asserting *USER* does not know UNIX-DLPR-COMMAND0
KNOME:  Since UNIX-DLPR-COMMAND0 is a UNIX-DLPR-COMMAND,
        asserting *USER* does not know UNIX-DLPR-COMMAND
KNOME:  UNIX-DLPR-COMMAND has difficulty SIMPLE, so deducing:
KNOME:  *USER* is SOMEWHAT-LIKELY to be NOVICE
        =) likelihood(*USER* = NOVICE) = LIKELY
KNOME:  *USER* is SOMEWHAT-UNLIKELY to be BEGINNER
        =) likelihood(*USER* = BEGINNER) = SOMEWHAT-UNLIKELY

Use lpr.
For example, to print the file named foo, type 'lpr foo'.

# What does rwho do?

KNOME:  Asserting *USER* does not know STATE11?
KNOME:  Asserting *USER* does not know HAS-EFFECT23
KNOME:  Asserting *USER* does not know UNIX-RWHO-COMMAND0
KNOME:  Since UNIX-RWHO-COMMAND0 is a UNIX-RWHO-COMMAND,
        asserting *USER* does not know UNIX-RWHO-COMMAND
KNOME:  UNIX-RWHO-COMMAND has difficulty MUNDANE, so deducing:
KNOME:  asserting *USER* = NOVICE

rwho is used to list all users on the network, list their tty,
and list their login time.
```

Figure 5. UC session with an novice

```
Welcome to UC (Unix Consultant) version 3.23
To a '#' prompt, please type in your questions
about the UNIX file system in English.
To leave, just type '^D' or '(exit)'.

Hi.
How can I help you?

# Can you tell me how to find out the inode of a file?

KNOME:  Asserting *USER* knows INODE1
KNOME:  Since INODE1 is an INODE,
        asserting *USER* knows INODE
KNOME:  INODE has difficulty COMPLEX, so deducing:
KNOME:  ruling out *USER* = NOVICE
KNOME:  ruling out *USER* = BEGINNER
KNOME:  *USER* is SOMEWHAT-UNLIKELY to be INTERMEDIATE
        => likelihood(*USER* = INTERMEDIATE) = SOMEWHAT-UNLIKELY
KNOME:  *USER* is SOMEWHAT-LIKELY to be EXPERT
        => likelihood(*USER* = EXPERT) = SOMEWHAT-LIKELY
KNOME:  Asserting *USER* does not know ACTION15?
KNOME:  Asserting *USER* does not know PLANFOR390
KNOME:  Asserting *USER* does not know UNIX-LS-i-COMMAND0
KNOME:  Since UNIX-LS-i-COMMAND0 is a UNIX-LS-i-COMMAND,
        asserting *USER* does not know UNIX-LS-i-COMMAND
KNOME:  UNIX-LS-i-COMMAND has difficulty COMPLEX, so deducing:
KNOME:  *USER* is SOMEWHAT-LIKELY to be INTERMEDIATE
        => likelihood(*USER* = INTERMEDIATE) = UNCERTAIN
KNOME:  *USER* is SOMEWHAT-UNLIKELY to be EXPERT
        => likelihood(*USER* = EXPERT) = UNCERTAIN

Type 'ls -i'.

# What does rwho do?

KNOME:  Asserting *USER* does not know UNIX-RWHO-COMMAND0
KNOME:  Since UNIX-RWHO-COMMAND0 is a UNIX-RWHO-COMMAND,
        asserting *USER* does not know UNIX-RWHO-COMMAND
KNOME:  UNIX-RWHO-COMMAND has difficulty MUNDANE, so deducing:
KNOME:  ruling out *USER* = EXPERT
KNOME:  only one candidate left, so asserting *USER* = INTERMEDIATE

rwho is like who, except rwho is for all users on the network.
```

Figure 6. UC session with an intermediate

user being a novice rises from LIKELY to TRUE. TRUE is the threshold (nothing can have a higher likelihood than TRUE), so KNOME concludes that this user is a novice.

In Figure 6, KNOME deduces that the user is an intermediate. First, KNOME deduces that the user knows what an inode is, since the user mentions "the inode of a file." Since knowing about inodes is a complex fact, KNOME is able to eliminate the novice and beginner categories as possibilities. KNOME also lowers the likelihood rating of the user as an intermediate and raises the likelihood rating of the user as an expert. However, these changes in likelihood ratings are canceled out by KNOME's deduction from the same user query that the user does not know the ls -i command, which is of complex difficulty. Next, KNOME deduces that the user does not know the rwho command, which is of mundane difficulty. This allows KNOME to eliminate the possibility that the user might be an expert, leaving KNOME to conclude that the user is an intermediate, since that is the only possibility left.

Even before KNOME has completely determined the expertise level of the user, KNOME's partial user model is still useful to the other components of UC. In the dialog with the novice, UC chooses to provide the user with examples of command-formats based on KNOME's initial deduction that the user is SOMEWHAT-LIKELY to be a novice. On the other hand, UC does not provide examples of simple command-formats to the intermediate user who presumably would already know such simple formats. Also, UC is able to use a simile to explain to the intermediate user the rwho command in terms of the who command. The novice user was given a complete explanation of rwho, since KNOME believed that the user would not know the who command and so would not understand the simile.

4. Modeling UC's Knowledge

Besides modeling what users know, KNOME also models what UC itself knows. Such a model allows KNOME to differentiate between cases where UC lacks knowledge and where the user has a misconception.

4.1. Open vs. Closed World Models

An important problem in AI knowledge bases is to define the limitations of the knowledge base. One solution is to use a closed world model, which states that the knowledge base knows everything (see [REIT78]). In a closed world model, anything that is neither in the knowledge base nor deducible from the knowledge base is deemed false. Such a model works only for completely self-contained microworlds such as airline reservation systems that know all airline flights. For real world applications, a closed world model is untenable, since real world knowledge bases cannot know everything even in just their own area of expertise. So a system with a closed model would be prone to giving out false information such as claiming

that objects that the system did not happen to know about do not exist, claiming that actions that the system could not figure out cannot be done, etc.

The other extreme is an open world model, which states that anything that is neither in the knowledge base nor directly deducible from the knowledge base is not known. This model has a significant problem with ruling out things that do not exist. For example, a knowledge base might list all the participants in any particular relation. Using a closed world model, the system would know that these are the only participants of that relation. However, in an open world model, the system cannot rule out that there might not be other participants unless the knowledge base explicitly encodes for all other possible participants that they do not participate in that relation. Since there are usually many more such negative facts than positive facts, encoding such facts in a knowledge base is extremely inefficient.

Neither model is quite adequate. A closed world model allows a system to rule out spurious hypotheses easily, but is prone to ruling out true facts that were not in the knowledge base. An open world model allows the system to be non-committal about hypotheses that are not covered by its knowledge base, but does not allow it to rule out spurious hypotheses when the system *does* have complete information. The ideal system needs to combine the best of both models. It needs to be assertive in ruling out spurious hypotheses when the system does have complete information, yet remain non-committal when it does not have complete information. To do this, a system needs to know the limitations of its own knowledge.

4.2. Meta-Knowledge

UC uses an open world model augmented by knowledge about the coverage limitations of the knowledge base. By using an open world model, UC is able to remain non-committal about information that is not in its knowledge base. This allows UC to profess ignorance about things outside its area of expertise. On the other hand, when UC does have complete information about an area, this is explicitly encoded in KNOME's model of UC's knowledge. In a sense, this is a model of what UC itself knows. Such knowledge is called *meta-knowledge.*

The term meta-knowledge was previously used for AI knowledge bases by [BARR77, DAVI77]. [BARR77] argues that AI knowledge bases need metaknowledge, i.e. the system's knowledge about "the extent and limit of available knowledge, what facts are relevant in a given situation, how dependable they are, and how they are to be used." [DAVI77] describes meta-knowledge in the TEIRESIAS system and its use in knowledge acquisition and in selecting which rules to try applying first through the use of *meta-rules.* [SMID83] use meta-level reasoning to find all of the solutions to a problem. Although they use a closed world assumption, they suggest that other systems for which the closed world assumption is not valid can encode limits on the number of solutions and use these limits to cut off inference. Such limits can be considered a type of meta-knowledge.

In KNOME, meta-knowledge is not primarily used for the same purposes as in TEIRESIAS; rather it is primarily used in dealing with user misconceptions. So KNOME's meta-knowledge describes the coverage limitations of UC's knowledge

base. Chapter III, Section 4 describes how user misconceptions are detected and how meta-knowledge is used in correcting the user's misconceptions.

4.3. Representation of Meta-Knowledge

Meta-knowledge in KNOME is represented explicitly as a collection of KODIAK statements about what UC knows. Examples of meta-knowledge include the fact that UC knows that it knows all file manipulation commands and the fact that UC knows all possible side effects for all known simple commands, most known mundane commands, and a few known complex commands. The fact that UC does not know all of the side effects for known commands is due to not enough programming and the knowledge limitations of UC's programmers rather than any inherent limitation of UC.

Figure 7 shows the representation of KNOME's meta-knowledge about simple commands. KNOW1 states that UC knows all of the effects of simple commands, and KNOW2 represents the fact that UC knows all of the options of simple commands.

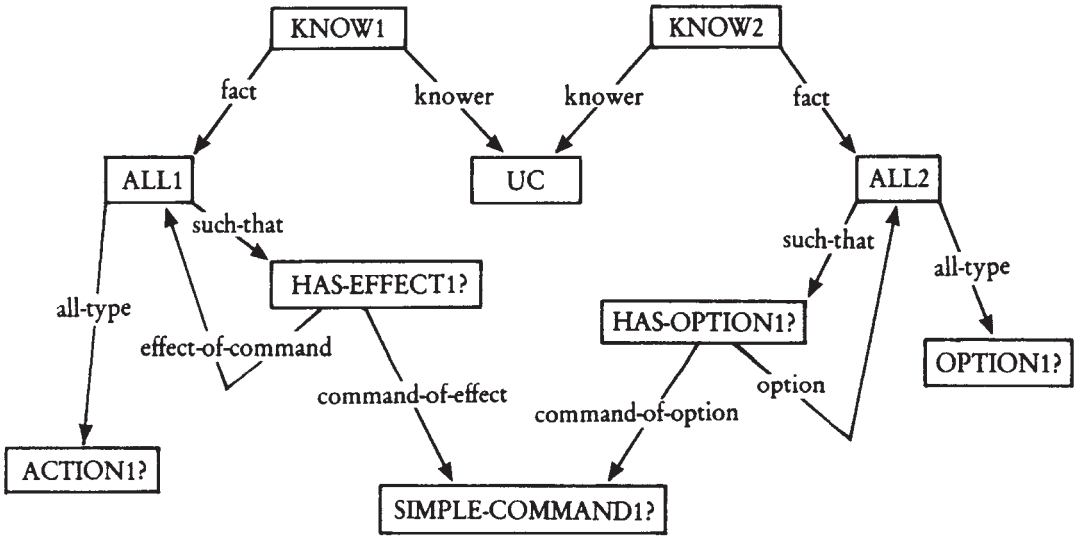


Figure 7. KODIAK representation of some of UC's meta-knowledge

Besides the explicit meta-knowledge modeled by KNOME, UC also has meta-knowledge that is represented implicitly within the KODIAK representation language. For example, any aspectual of a relation can be constrained to be filled by only certain classes of objects. This effectively constrains the way in which different objects may be related to other objects in KODIAK. Of course, these constraints are sometimes violated when users speak metaphorically.

4.4. Other Approaches to User Misconceptions

Other systems that have dealt with user misconceptions use one of two common approaches. The first approach compiles an a priori list of possible user misconceptions for use in detecting misconceptions. This is common among ITS (Intelligent Tutoring Systems) (e.g. [BROW78, STEV79, SLEE81, JOHN84, REIS85, ANDJ85b]). These systems are not very flexible, since they cannot deal with any user misconceptions that are not listed among their precompiled a priori lists.

Another approach attempts to detect classes of misconceptions. Among the best of these is Kaplan's CO-OP natural language database-query system [KAPS83], which handles a class of faulty user presumptions. An example from CO-OP is shown in Figure 8.

User: Who advises projects in area 36?

CO-OP: I don't know of any area #36.

Figure 8. CO-OP example with hedged response to misconception

The user in the CO-OP example believes that there is an area 36. In processing the user's query, CO-OP finds that there is no area 36 listed in its database, so it detects a possible user misconception: the user may mistakenly presume that there is an area 36 when in actuality there might not be an area 36. Note that CO-OP's reply does not actually *correct* the user's misconception, rather it *hedges,* claiming that it does not know of any area #36. By hedging, CO-OP cannot correct the user's misconception, but can only *suggest* to the user that the user might be mistaken. CO-OP is unable to take corrective action, because it does not assume either an open or closed world model for its database. As a result, it cannot tell from the fact that there is no area 36 in its database whether there really is no area 36, or if area 36 was just left out of its database. Unlike UC, in which the KNOME subcomponent models UC's own knowledge with meta-knowledge, CO-OP does not have a model of its own database.

A corrective response is shown in Kaplan's example of a cooperative human, shown in Figure 9.

User: Which students got a grade of F in CS105 in Spring 1980?

Human: CS105 was not given in Spring 1980.

Figure 9. Kaplan's example of a cooperative human respondent

Because CO-OP does not adopt either an open or closed world model, it would not be able to deny that CS105 was given in Spring 1980 as in Kaplan's example of a cooperative human. CO-OP would only be able to suggest a misconception by saying something like, "I don't know of any CS105 in Spring 1980." To be able to actually correct the user, CO-OP would need to know that the absence of a CS105 among the Spring 1980 courses implies that there was no CS105 offered then. That is, CO-OP would need meta-knowledge. Since CO-OP was designed to be easily transportable among different databases, it did not have such non-transportable meta-knowledge.

The DAIQUIRY module of HAM-ANS [MARB87, MARB88] extended the misconception approach of CO-OP to include conceptual failures in which the user asks about objects not modeled by the database. Because DAIQUIRY, like CO-OP, did not have meta-knowledge, it too could only hedge its replies to the user. For example, it would reply, "The system has no knowledge about 'SHIP'," when SHIP was not in its conceptual hierarchy.

[MAYS80b, WEBB83] report efforts to deal with user misconceptions that can be found by constraint violations on relations. For example, if the TEACH relation is constrained to relate only FACULTY to COURSES, then the system can detect a misconception if the user asks about UNDERGRADUATES that TEACH. This type of relational constraint can be considered a kind of implicit meta-knowledge. It differs from the explicit meta-knowledge of KNOME in that relational constraints can only be used to detect misconceptions about what can be the case, and not misconceptions about what is the case. For example, relational constraints can be used to deduce that only commands can have command-options, but not whether any particular command has any particular command-option.

After a user misconception has been detected, it must be corrected. [MCCO85a, MCCO*] discuss strategies for correcting object-oriented misconceptions. [QUIL*] discusses correcting plan-oriented misconceptions in AQUA. AQUA assumes a closed world model for its knowledge base of plans, called an "advisor model." As a result of this assumption, AQUA can only model perfect advisors that have complete knowledge of the domain. If AQUA's advisor model were incomplete, then AQUA would tend to mislead the user. For example, if an obscure side effect of a plan were inadvertently left out of AQUA's knowledge base, then AQUA would mislead the user to believe that the plan did not have this a side-effect. Some form of meta-knowledge would be required to allow AQUA to model an advisor without perfect knowledge of the domain.

4.5. Detecting Misconceptions in UC

User misconceptions are detected by UC during the processing of the user's query. Currently, UC only handles relational misconceptions, that is, misconceptions in which the user believes a relation holds between two objects when in fact, such a relation cannot hold or does not happen to hold between those particular objects. UC does not handle object-oriented misconceptions, such as those for which [MCCO85a] discusses correction strategies.

In processing the user's query, UC checks to see whether all relations mentioned by the user in the user's query have a counterpart in UC's knowledge base. For example, if the user asks, "What does ls -e do?", then UC's parser/understander understands part of this as a HAS-OPTION relation relating an instance of the ls command and an instance of the -e option. While looking for the effects of the command, UC finds that there is no corresponding HAS-OPTION relation between the ls command and the -e option in UC's knowledge base. At this point, UC suspects that the user may have a misconception.

In order to see whether the user actually has a misconception, or whether UC just lacks information about this particular option of this command, UC checks with the KNOME subcomponent, which models the limitations of UC's knowledge base using meta-knowledge. If KNOME has meta-knowledge that UC knows all of the options of this command, then UC knows that the user has a misconception. On the other hand, if the options of this command are not covered by KNOME's meta-knowledge, then UC cannot assume that the user has a misconception. Instead, UC must assume that it lacks information about this command/option combination.

4.6. Correcting Misconceptions

After UC has determined either that the user has a misconception or that UC lacks knowledge, UCEgo decides how UC should reply to the user. When the user has a misconception, UCEgo tries to correct this misconception by denying that what the user mistakenly believes is the case. On the other hand, if UC determines that it lacks information, then UCEgo apologizes to the user for not knowing the answer to the user's query. These two types of responses are shown in the UC session of Figure 10.

```
Welcome to UC (Unix Consultant) version 3.23
To a '#' prompt, please type in your questions
about the UNIX file system in English.
To leave, just type '^D' or '(exit)'.

Hi.
How can I help you?

# What does who -b do?

who does not have a -b option.

# What does ruptime -t do?

I'm sorry, I do not know that.
```

Figure 10. UC session showing different UC responses to possible misconceptions

In the first query, UC corrects the user's misconception that who has a -b option. It does this by first noticing that the user's usage of "who -b" translates into a HAS-OPTION relation between a UNIX-WHO-COMMAND and a -b-OPTION. There is no equivalent HAS-OPTION relation in UC's knowledge base, so UC suspects a possible user misconception. To see if this is the case or if UC just lacks knowledge about this particular option, UC consults the meta-knowledge stored in KNOME. The appropriate meta-knowledge in this case is the fact that UC knows all the options of all simple commands. Since who is a simple command, and since the -b-OPTION is not listed among the options of UNIX-WHO-COMMAND in UC's knowledge base, KNOME can infer that there is no such option for who. Next, UCEgo decides that UC should correct the user's misconception by denying the existence of a -b option for who.

On the other hand, in the second query, UC professes ignorance about the -t option of the ruptime command. As in the previous query about who, UC detects a possible misconception when it does not find a -t option listed for ruptime in its knowledge base. However, in this case, ruptime is a complex command, so the previous meta-knowledge does not apply. There is no meta-knowledge about the options of complex commands (due to not enough programming by UC's implementors rather than any inherent limitation of UC), so UC cannot tell if ruptime has a -t option. In order to be polite, UCEgo apologizes to the user for not knowing.

5. Exploiting KNOME

KNOME is used by several different components of UC. This section will provide a survey of these applications, since a detailed analysis is beyond the scope of this chapter.

One application of KNOME is in goal analysis (see [WILE86]). KNOME can be used to choose between different interpretations of the user's statements. For example, consider the user's query "Do you know how to delete a file?" There are two possible interpretations of this query: a direct interpretation where the user wants to know if UC knows how to delete a file; and an indirect speech act interpretation where the user wants to know how to delete a file. If the goal analyzer knows that the speaker already knows how to delete a file, then the system can rule out the indirect interpretation. Similarly, if the system believes that the user does not know how to delete a file, then the indirect interpretation is more likely.

Another use for the user model is to keep UC from telling the user something that the user already knows, and to express information in terms of concepts with which the user is already familiar. For example, in expressing the answer, the UC-Express component of UC prunes those parts of the answer that the user already knows. So in the following UC dialog, UCExpress gives an example of the format of the vi command to a novice, since the novice likely does not know the simple-file-format. To beginners and other non-novices, UCExpress does not provide the example, since UC can assume that the user will know that to use vi, one types "vi" followed by the name of the file to edit.

User: How can I edit a file?
UC to novice: Use vi.
For example, to edit the file named foo, type 'vi foo'.
UC to beginner: Use vi.

KNOME is also used by UCExpress when determining whether or not to use an analogy in explaining the usage of a command. Consider the following UC dialog. UCExpress needs to know that the user already knows the uptime command before it can choose to use the analogy.

User: What does ruptime do?
UC: ruptime is like uptime, except ruptime is for all machines on the network.

UCEgo [CHIN87], the agent component of UC, uses KNOME when deciding whether or not to provide additional information that the user may need (cf. [JOSH84a, JOSH84b] for other theoretical approaches). For example, in the UC dialog shown in Figure 11, UCEgo does not actually answer the user's question.

User: Who is on the system?

UC: To find out who is on the system, type 'who'.

Figure 11. UC volunteers information that the user does not know

UC does not actually tell the user who is on the system (e.g. by saying "John, Mary, and David are on the system"). Instead, UCEgo decides to tell the user how to find out this information, namely use the who command. However, if the user already knows the who command, then UC's answer would be superfluous. So UCEgo first consults KNOME to see if the user lacks such knowledge before providing the additional information.

6. Conclusion

6.1. Summary

The user modeling system of KNOME is extremely simple, yet it provides enough functionality for the needs of UC. KNOME can infer what the user knows from a consulation dialog with the user. Based on only a few such facts about what a user does or does not know, KNOME can also *predict* many other default facts about what the user might or might not know. KNOME's stereotypes serve to efficiently group those default inferences that commonly co-occur. Using only a few such stereotypes, KNOME is able to provide enough differentiation among potential

users so that UC can use these differences in answer expression, goal analysis, and detection of situations when the user lacks knowledge or has misconceptions. The double-stereotype system is not only space efficient, but also allows easy addition of knowledge to KNOME. When a new fact is added to UC, the knowledge engineer needs only to classify the new fact according to its difficulty level.

Although no attempt has been made yet to apply the methodology to other areas, it would seem that many of the techniques should transfer well. Stereotype categories are used in UC in similar fashion to the way humans use categories, that is as reference points for reasoning about individual members (see [ROSC77, ROSC78, ROSC83]). Thus stereotypes should be useful for approximate reasoning wherever human categorization has proven to be useful. The double-stereotype system extends the use of computer stereotype categories to cover more of human categorization, that is, to include categories for knowledge as well as categories/stereotypes for people/users. The double-stereotype system can be applied straightforwardly to other domains where a computer system needs to model the relation between different classes of users and different classes of information. In cases where the user population is homogeneous or where the information is homogeneous, then a double-stereotype system would not be applicable.

The methodology used to deduce the user's stereotype in KNOME transfers in two ways to other domains. First, the method used to infer individual facts about what the user knows is applicable to any natural language system that needs to infer facts about the user's knowledge based on what the user says. On the other hand, the methodology used to combine such evidence is highly dependent on a double-stereotype system. It can, however, be used in any double-stereotype system where the system can deduce particular relations between the user's stereotype and the modeled information.

6.2. Current Status

KNOME is currently implemented in FRANZ LISP and Common LISP in a version of KODIAK [WILE87], which is a network style knowledge representation language featuring objects and relations, multiple inheritance, and other epistemological tools.

6.3. Problems

Currently, KNOME does not address the problem of whether the user knows something after UC has informed the user. The difficulty is that the user may not understand the system's presentation, in which case the user modeling system should not infer that the user knows the new information. A drawback of systems limited to natural language interactions is that these systems cannot watch the user's reactions to deduce whether or not the user is having trouble understanding the system's presentation. Human consultants are able to switch explanation strategies midway through a presentation when they notice their clients' facial expressions showing confusion. A computer consultant would have to wait for the user to

respond with language in order to get feedback. In such cases, the best that a user modeling system can do may be to assume that the user has understood (perhaps with a lower certainty factor), until the modeler gets evidence to the contrary. Then, if the user does not ask for a clarification in the next exchange, the modeler can increase the certainty of its belief that the user has indeed understood.

Another deficiency in KNOME is that it does not contain very many specific rules for inferring what specific facts a user might know based on other specific facts that the user knows. For example, if a user knows the rwho command, then it is very likely that the user will also know the who command. KNOME has avoided these types of explicit rules in favor of the more efficient, though less specific mechanism of double- stereotypes. The double-stereotypes allow KNOME to combine a small number of explicit facts to first infer the user's level of expertise and from this, the user's knowledge of large sets of other facts. Although this powerful general mechanism works well, there are still cases when explicit rules will give more specific information about what a user knows with a higher degree of certainty. The GUMS general user modeling shell [FINI*] makes good use of these explicit rules.

A possible deficiency with KNOME's method of combining evidence is that KNOME marks the user as belonging to a particular stereotype after only a short time (typically 3 interchanges). After this, KNOME does not alter its conception of the user's level of expertise. This is reasonable in the UC domain, because UC sessions are typically short. However, this would not be acceptable for longer term user models. If a system needs to keep track of a user's level of expertise over periods long enough that the user's expertise may increase appreciably, then the system needs to be able to change its judgment of the user's level of expertise. This process is easier than one might suppose, since a user's level of expertise tends to grow monotonically (it seems reasonable to ignore the phenomenon of forgetting).

Currently, KNOME only has a single range of user expertise and a single range of knowledge difficulty. This is sufficient for UC, since UC only covers commands related to the UNIX file system. In order to extend UC to cover other areas of UNIX such as using the vi or emacs editors, KNOME would need additional ranges of user expertise levels and information difficulty levels. Problems which need to be addressed with multiple ranges include how the levels in one range might relate to the levels of other ranges. For example, a system may be able to predict that an expert in using vi is likely to be an expert in using the UNIX file system. On the other hand, there may be no such relations between familiarity with emacs and familiarity with the UNIX file system, since the emacs editor is common to many other operating systems.

Another area which KNOME does not address completely is how to model users of other operating systems. Such users may be able to transfer some amounts of expertise to UNIX. These users can benefit from analogies between UNIX and the operating system that they know. On the other hand, such users sometimes incorrectly apply analogies and often mix up commands. Adding additional stereotypes for such users would enhance KNOME's ability to provide proper analogies and to detect improper analogies or command usage. [MACM83] has shown that one user modeling system can adequately model different operating systems (UNIX,

TOPS-20, and VM). However, no one has investigated how knowledge of the user's expertise in one domain transfers to another domain (correctly or incorrectly) or how to exploit such knowledge (e.g. to provide analogies).

Other areas not addressed by KNOME include applying the techniques demonstrated in KNOME for other purposes such as selection of terminology in generation (cf. [LEHM*]). The approach used in KNOME would also be useful in the selection of different explanation strategies as in the TAILOR system [PARI*].

7. References

[ALLE80], [ANDJ85b], [BARR77], [BROW78], [BURT76], [CARB83], [CARB*], [CARR77], [CHIN83], [CHIN86], [CHIN87], [DAVI77], [FINI*], [HOEP84b], [JOHN84], [JOSH84a], [JOSH84b], [KAPS83], [KASS*], [KAYD85], [KOBS85b], [KOBS85d], [KOBS86a], [LEHM*], [LITM84], [MACM83], [MARB87], [MARB88], [MARJ85], [MATT86], [MAYS80b], [MCCO85], [MCCO85a], [MCCO*], [MORI85b], [MORI*], [NESS87], [PARI*], [QUIL*], [REIS85], [REIT78], [RICH79b], [RICH83], [RICH*], [ROSC77], [ROSC78], [ROSC83], [SEAR69], [SLEE81], [SLEE82c], [SMID83], [STEV79], [WAHL*], [WEBB83], [WILE84], [WILE86], [WILE87], [ZADE65], [ZADE82]

5 Detecting and Responding to Plan-Oriented Misconceptions [1]

Alexander Quilici

Abstract

This chapter[2] discusses the problem of detecting and responding to plan-oriented misconceptions in advice-seeking dialogs, concentrating on the problems of novice computer users. A cooperative response is one which not only corrects the user's mistaken belief but also addresses the missing or mistaken user beliefs that led to it. Providing a cooperative response is presented as a process of the advisor trying to explain why he doesn't hold the user's belief and presenting this explanation to the user. An explanation is shown to correspond to a set of advisor beliefs that together contradict the user's belief, and searching for an explanation to determine whether various abstract configurations of advisor beliefs hold. The search for an explanation makes use of both domain-specific advisor beliefs and a taxonomy of domain-independent explanations for potential user misconceptions. A taxonomy for misconceptions involving plan applicability conditions, preconditions, and effects is presented, along with a method of organizing the necessary advisor planning knowledge.

1. Introduction

A novice computer user having trouble performing a task often describes his problem to an expert, who is then expected to explain its cause and provide its solution. Consider, for example, the following advice-seeking dialog between a novice UNIX user and his more experienced counterpart.

> **User:** I tried to remove a file with the "rm" command. But the file was not removed and the error message was permission denied. I checked and I own the file. What's wrong?

---

[1] This chapter is an extended and revised version of [QUIL88].

[2] The work reported here was supported in part by a grant from the Lockheed Software Technology Center (Austin, Texas). Special thanks go to Mike Gasser, John Reeves, and Ron Sumida for fighting their way through several earlier incoherent and uninteresting versions of this chapter. Comments by the editors of this book and a collection of anonymous reviewers have also greatly improved its organization and content.

Advisor: To remove a file, you need to be able to write into the directory containing it. You do not need to own the file.

User: How do I make that directory writeable?

Advisor: To make a directory writeable, use the "chmod +w" command.

User: Wait a second while I try it. No, "chmod +w" prints the error message "permission denied".

Advisor: To make a directory writeable, you need to own it.

User: So to remove a file, I have to own the directory that contains it.

Advisor: No, to remove a file, you need to have write permission on the directory that contains it. You do not need to own the directory that contains it. You need to own that directory when you do not already have write permission on it.

User: So how do I remove the file?

Advisor: Send mail to whoever has write permission on the directory, asking him to remove the file for you.

The advisor in such a dialog must be able to recognize missing or mistaken user beliefs about plan applicability conditions, preconditions, and effects. The advisor above recognizes two user misconceptions: an initial incorrect belief that owning a file is a precondition to removing it, and a subsequent incorrect belief that the precondition is owning the directory in which it resides. This advisor also notices several gaps in the user's knowledge: The user has no plan for making a directory writeable, does not know why the advisor's plan for doing so failed, and has no plan for removing a file that isn't in a writeable directory.

The advisor must also be able to respond to both missing and mistaken user beliefs. A cooperative advisor response to a *missing* user belief simply provides that belief. This advisor provides the missing beliefs that the plan for making a directory writeable is to use the "chmod +w" command, that owning a directory is a precondition to making it writeable, and that the plan for removing a file when all else fails is to send mail requesting its removal to some person who can write into the directory in which the file resides.

For a *mistaken* user belief, however, the advisor must not only point out that the belief is incorrect and provide a correction, but must also address the missing or mistaken beliefs that are the source of this misconception. Above, this is done by pointing out that the actual precondition is being able to write into the directory that contains it, and by explaining that owning that directory is necessary only if it is not already writeable.

This chapter examines the problem of detecting and responding to mistaken user beliefs. We consider misconceptions involving plan applicability conditions (whether a particular plan should be used to achieve a goal), enablements (whether a particular state must exist before a plan can achieve a goal), and effects (whether a state will exist as a result of a plan's execution). This problem can be broken into several subproblems:

1. Mapping the user's natural language problem description into a set of user beliefs.
2. Determining which of these beliefs are incorrect.

3. Inferring the missing or mistaken user beliefs that might have led to these incorrect beliefs.
4. Selecting the advisor beliefs to present to the user as a conversationally-cooperative response.
5. Mapping these advisor beliefs into a natural language response.

This chapter provides a computational model of (2), (3), and (4). Its input is an initial user model composed of a set of potentially incorrect user beliefs. Its output is a set of advisor beliefs that constitute a cooperative response to any mistaken user beliefs. The advisor selects these beliefs by trying to explain why he doesn't hold the user's mistaken beliefs.

The rest of the chapter presents this explanation-based approach to misconception detection and response. We begin with an overview of our approach and of our representation for plan-oriented user and advisor beliefs. We then examine the process by which the necessary explanations are found and the knowledge needed to find these explanations. As part of this discussion, we provide a taxonomy of explanations for the types of plan-oriented beliefs we consider, and show how the advisor's planning knowledge can be organized to facilitate the process of explanation finding. Finally, we compare our approach with other work in detecting and correcting user misconceptions, and discuss its implementation and limitations.

2. An Explanation-Based Approach

How can an advisor decide whether a particular user belief is mistaken and then understand how the user came to believe it? And furthermore, how can the advisor determine the contents of a cooperative response to the user's misconception?

We suggest an explanation-based approach to accomplish these tasks. To understand why the user holds a mistaken belief, the advisor tries to *explain* why he does *not* hold this user belief. The explanation consists of a set of advisor beliefs that together contradict the user's incorrect belief. These beliefs constitute the advisor's response. The underlying assumption is that the user would not hold his mistaken belief if he knew the reasons why the advisor does not hold it.

We illustrate our approach by showing how the advisor arrives at the response found in this exchange from our introductory dialog.

User: So to remove a file, I have to own the directory that contains it.

Advisor: No, to remove a file, you need to have write permission on the directory that contains it. You do not need to own the directory that contains it. You need to own that directory when you do not already have write permission on it.

The user's possibly incorrect belief is that a file cannot be removed without owning the directory in which it resides.

The advisor tries to explain why he does not hold this user belief. The explanation he finds consists of two related beliefs. The first is his belief that a file cannot be removed without write permission on the directory containing it. The other is

his belief that owning the directory is necessary only if one does not already have write permission on it. Together these beliefs contradict the user's belief that owning the directory is always necessary to remove a file. The contradiction is that the advisor believes that the user's claimed precondition for removing a file is merely a precondition for achieving one of its preconditions.

Once the advisor finds this explanation, he presents it to the user. This response corrects the user's misconception by pointing out that the user's claimed precondition for removing a file is incorrect, by providing the actual precondition for removing a file, and by providing the relationship between the user's presumed precondition and the actual precondition. The user's being unaware of this relationship led to the user's misconception.

In effect, our approach has the advisor infer a set of beliefs that contradict the user's erroneous beliefs. Our assumption is that the user's misconception resulted because he was unaware of these advisor beliefs.

Our approach derives from work in explanation-based story understanding [SCHR86, DYER83, WILE83, SCHR77]. The basic idea is that to understand a particular input, such as a person's action, we have to explain why it has occurred. One way to explain a person's action is to relate it to known goals the person is trying to achieve. Suppose, for example, that a story understander reads that a hungry character bought a restaurant guidebook [WILE83]. One explanation for this action is that hungry people want to eat, to eat you have to be near food, and to be near food you have know where it is and then go there. The guidebook says where the food is. This explanation can be constructed either by using rules to build a reasoning chain or by applying preexisting schemas that capture the relationship.

Our task is to try to understand why the user holds a particular belief, a task similar to that faced by explanation-based story understanders. Since we have a similar task, we use a similar approach.

3. Representing User and Advisor Beliefs

The mistaken user beliefs that we consider involve plan applicability conditions, enablements, and effects. In this section we describe how we represent these beliefs. In essence, we utilize existing frameworks for representing planning knowledge, except that we are careful to distinguish between user and advisor beliefs.

Traditional planning systems [SACE74, FIKE71] represent an agent's planning knowledge as a database of operators associated with applicability conditions, preconditions, and effects. Since these systems have only one agent, the planner, the entries in the database are implicitly assumed to represent that agent's beliefs. However, because user misconceptions occur when the user's planning knowledge differs from the advisor's, systems that deal with user misconceptions must explicitly distinguish between advisor beliefs about what the user knows and advisor beliefs about what the advisor knows.

Our representation for beliefs [ABEL79, ABEL73] is similar to that of existing systems that keep track of the possibly contradictory knowledge of multiple

participants [ALVA86, POLL86b, FLOW82]. A **belief** relation represents an advisor's belief that an actor maintains that a particular plan applicability condition, precondition, or effect holds. The actor is either the user or the advisor.

belief(user, R)	Advisor believes that user maintains *R*
belief(advisor, R)	Advisor believes that advisor maintains *R*

In this chapter we do not discuss beliefs involving other relationships, such as a belief that an object has a particular property. In addition, for readability we do not use the **belief** predicate here but instead precede a list of planning relationships with either "the user believes" or "the advisor believes".

3.1. Representing Planning Relationships

The planning relation can be one of the relations between actions and states shown below. Here *A* denotes an action, which is either a primitive operator whose execution results in a set of state changes, or a plan, which is a sequence of these operators. *S*, *S1*, and *S2* denote states, which are descriptions of properties of objects.

causes(A,S)	Executing A has S an effect
!causes(A,S)	Executing A does not have effect S
enables(S1,A,S2)	S1 is necessary for A to have S2 as an effect
!enables(S1,A,S2)	S1 is unnecessary for A to have S2 as an effect
applies(A,S)	A is a correct or normal plan for achieving goal state S
!applies(A,S)	A is not a plan for achieving S
precludes(S1,S2)	S1 and S2 cannot exist simultaneously
!precludes(S1,S2)	S1 and S2 can exist simultaneously
goal(A,S)	Actor A wants to achieve S

These relationships are derived from existing representations. SPIRIT's [POLL86b] representation for planning knowledge uses **gen** to represent a state resulting in an action and **cgen** to represent a state resulting in an action only if some other state exists. **causes** and **enables** are identical in semantics to **gen** and **cgen**. **applies**, which has no analog in SPIRIT, is similar to the **intends** relation in BORIS [DYER83]. The difference between **causes** and **applies** is in whether the action is *intended* to cause the state that results from its execution to exist. **causes** represents cause-effect relations which are nonintentional – the effect is not necessarily a user goal – while **applies** represents a cause-effect relation between an action (sequence) or plan which is intended to achieve a desired state (a goal). An action **causes** a state whenever the state results from its execution. But an action **applies** to a state only when an actor believes the action causes the desired state to occur.

To see why this distinction is necessary, consider two actions that can be used by a user who wants to remove one of his files: typing "rm" followed by the file's name, and typing "rm" followed by an asterisk. Both have removing the file as one of their effects, but the latter also causes all other files to be removed as well, an

effect that is *not* the user's *goal*. Only typing "rm file" **applies** to removing a file, although both actions have an effect that **causes** the file to be removed.

To further illustrate the semantics of these relations, we use them to represent the first exchange in our example dialog.

User: I tried to remove a file with the "rm" command. But the file was not removed and the error message was permission denied. I checked and I own the file. What's wrong?

Advisor: To remove a file, you need to be able to write into the directory containing it. You do not need to own the file.

Three of the user's beliefs in this exchange are: (1) the "rm" command is used when one wants to remove a file, (2) one has to own a file to remove it, and (3) an error message resulted when the plan was executed. In terms of these planning relations, the user's beliefs are:

applies(using "rm file", the file's removal)
enables(owning the file, using "rm file", the file's removal)
causes(using "rm" on the user's file, an error message)

The advisor holds several similar beliefs, except that he believes that to remove a file it is necessary to have write permission on the directory containing it. In terms of the planning relationships, the advisor's beliefs are:

applies(using "rm file", the file's removal)
enables(directory write permission, using "rm", the file's removal)
causes(using "rm" on the user's file, an error message)

(The chapter is not concerned with representing notions such as "the file's removal" or "write permission on the directory containing the file". The details of the representation for such things may be found in [QUIL85].)

The user and advisor in this exchange share one belief that we have not yet represented. This belief is that using "rm" did not cause the user's file to be removed. To represent beliefs that a state does not result from an action, that a plan is not applicable to a goal, or that a state is not an enablement condition of an action having another state as a result, we use **!causes**, **!applies**, and **!enables**, respectively. The belief above is represented with **!causes**, a belief that "mkdir" is not used to remove a file is represented with **!applies**, and a belief that "rm" does not require owning the directory containing the file is represented with **!enables**.

!causes(using "rm" on the user's file, the file's removal)
!applies(using "mkdir file", the file's removal)
!enables(owning the directory, using "rm", the file's removal)

Examining user-advisor exchanges makes it is apparent that we must also be able to represent the notion that a state's existence caused a planning failure. Consider the following exchange.

User: I accidentally hit the up arrow key and it deleted 20 unanswered mail messages. How can I get them back?

Advisor: Hitting the up arrow does not delete your messages, but does result in your being disconnected from the etherplexer. You could not access your mail messages because they were moved to "mbox". The mail program requires that your mail messages be in "mailbox".

Here the advisor believes that the user's mail messages are inaccessible because they are not in the location the mail program expects them to be. The belief that the mail program requires the mail messages to be in the file "mailbox" can be represented using **enables**. The advisor's belief that the mail messages being in the file "mbox" prevents the mail program from accessing is represented with **precludes**, which captures the notion that two states are mutually exclusive.

enables(messages in "mailbox", use "mail", display messages)
precludes(messages in "mbox", messages in "mailbox")

precludes and **!precludes** relations between states can be inferred using rules such as "an object cannot be in two places at once".

One other relation we find useful is **goal**, which represents a belief that an actor wants to achieve a particular state. In the example above, the advisor believes that one goal of the user is accessing his mail messages.

goal (user, access user's mail messages)

Most user-modeling systems use a similar relation to explicitly represent that a state is a user's goal.

3.2. Summary of the Representation

The main focus of our work is in trying to detect and respond to user misconceptions. To do so, it is necessary to have some representation for user and advisor planning knowledge. Our representation is based on that used by traditional planning systems. The most important difference is that we take care to distinguish between things the advisor believes and things the advisor thinks the user believes. We also distinguish between actions that are intended to achieve a state and actions that happen to have a particular state as one of their effects. And we find it necessary to represent beliefs that two states cannot exist at the same time and that achieving a particular state is a goal of the user.

4. Explanation-Based Misconception Recognition and Response

Our approach to recognizing and responding to a potentially incorrect user belief revolves around the advisor trying to explain why he does not hold a particular user belief. But what exactly is an explanation? And what knowledge does the advisor need to find one? And, finally, how is an explanation found?

4.1. Explanations as Sets of Beliefs

An explanation is a set of advisor beliefs that justify his not sharing a user's belief. To find an explanation the advisor tries to find a set of advisor beliefs that contradict the user's belief.

For instance, the user in our earlier example incorrectly believes that owning a directory is a precondition for removing a file.

enables(own directory, use "rm", the file's removal)

The advisor holds two beliefs that together contradict this user belief. The first is the advisor's belief that the actual precondition is write permission on the directory containing the file. The other is his belief that owning a directory is a precondition to obtaining write permission on the directory.

enables (writeable directory, use "rm", the file's removal)
enables(own directory, use "chmod", obtain writeable directory)

Together these beliefs let the advisor infer that owning a directory is not a precondition of removing a file, the negation of the user's belief.

!enables(own directory, use "rm", the file's removal)

Thus, these beliefs constitute an explanation for the user's mistake: the user does not realize that owning a directory is merely a precondition to obtaining write permission on the directory, which is the actual precondition to removing the file.

4.2. Required Advisor Knowledge

Two types of knowledge are needed to locate these explanatory advisor beliefs: (1) a set of domain-specific beliefs, (2) a set of abstract explanation patterns that let the advisor recognize contradictory beliefs.

4.2.1. Domain-Specific Beliefs

The first type of required knowledge is a set of domain-specific beliefs about plan applicability conditions, preconditions, and effects. Examples of these include beliefs that "rm" is used to remove a file, and that it is necessary to have write permission on the directory containing the file. Without these types of beliefs it would be impossible for the advisor to correct user misconceptions about the preconditions for removing a file. This category of knowledge also includes beliefs such as a belief that "rm" is not used to remove a directory. These negated beliefs – **!applies**, **!enables**, **!causes**, and so on – are especially useful in detecting misconceptions. An advisor with the explicit belief that "rm" is not applicable to removing a directory, can easily detect that a user belief that "rm" is applicable to removing a directory is incorrect (although he may still have to do some work to understand why the user holds this belief).

These domain-specific beliefs are assumed to derive from past advisor experiences or past advisor interactions with users. An advisor who successfully uses "rm"

to remove a file will believe that using "rm" is applicable to the goal of removing a file. An advisor who uses "rm" to try to remove a directory and has it fail will believe that "rm" is not applicable to removing a directory. The negated beliefs correspond to the bug lists kept by many tutoring and planning systems [ANDJ85b, BURT82, STEV82, BROW78].

4.2.2. Potential Explanations

The other type of knowledge is a taxonomy of potential explanations. Each type of planning relation – **applies**, **enables**, and **effects** – is associated with a set of potential explanations that allow the advisor to recognize when his beliefs contradict the user's. Each potential explanation is an abstract pattern of planning relationships. To find a set of contradictory beliefs that constitute an explanation for why the advisor does not hold a particular user belief, the advisor classifies the belief and then tries to verify one of the potential explanations associated with that class of belief. The idea is that to verify a potential explanation, the advisor tries to prove, either by memory search or by deductive reasoning, that each of these planning relationships hold.

5. A Taxonomy of Potential Explanations

The potential explanations for why the advisor *does not* hold an instance of a plan-oriented belief are shown below. Each of these potential explanations suggests that to find a belief that contradicts the user's, the advisor must try to verify some small set of beliefs. This section presents a taxonomy of these explanations for each of the three types of beliefs.

5.1. Explanations for Incorrect Applies

There are four ways to recognize that the advisor does not hold a user belief that a plan is applicable to a goal; that is, a belief of the form **applies**(*Ap*, *Sp*). These are shown in Figure 1.

The first, *Achieves Other Goal*, applies to one of our earlier examples. The explanation is that the user's plan has no effect that achieves his goal, that his plan actually achieves some other goal, and that some other plan achieves his goal.

User: I tried to display my file with the "ls" command but it just printed the file's name.

Advisor: The "ls" command is not used to display the contents of files, the "more" command is. "ls" is used to list the names of your files.

The user's incorrect belief is that using "ls" displays a file. This belief is contradicted by the advisor's beliefs that using "ls" does not display the contents of files, that "ls" is applicable to listing the names of files, and that "more" can be used to display a file's contents.

Class of mistake	Potential explanation	English description
Plan Achieves Other Goal	!causes(Ap,Sp)	No effect achieves goal
	applies(Ap,S)	Plan applies to other goal
	causes(A, Sp)	Some other plan achieves goal
Plan Missing Needed Effect	!causes(Ap,Sp)	No effect achieves goal
	applies(A, Sp)	Other plan achieves goal
	causes(A,S)	That plan has an effect the
	!causes(Ap,S)	user's plan does not have
Unachievable Plan Enablement	enables(S,Ap,Sp)	Some state enables Plan
	!causes(Ao,S)	No action achieves this state
Plan Thwarts User Goal	applies(A, Sp)	Other plan achieves goal
	causes(Ap,S)	User plan has effect that
	precludes(S, Sp)	thwarts the user's goal
	goal(user, Sp)	Other plan does
	!causes(A,S)	not have that affect

Figure 1. Potential explanations for incorrect applies

The second explanation, *Plan Missing Needed Effect*, is that the user's plan P1 does not have one of the effects that the plan P2 (that achieves his goal) has.

User: I tried to remove my directory and I got an error message "directory not empty". But "ls" didn't list any files.

Advisor: Use "ls -a" to list all of your files. "ls" cannot be used to list all of your files because "ls" does not list those files whose names begin with a period.

The user's mistaken belief is that "ls" lists all file names. This belief is contradicted by the advisor's belief that "ls" does not have an effect that causes it to list files whose names begin with a period, an effect that the plan that achieves the goal (ls -a) has.

The third explanation, *Unachievable Plan Enablement*, is that there is no plan that achieves one of the user's plan's enablements.

User: So to read Margot's mail, all I have to do is "more ~flowers/mail".

Advisor: No, only "flowers" can read her mail.

The user mistakenly believes that his plan of using "more" to examine Margot's mail file will allow him to read her mail. This belief is contradicted by the advisor's beliefs that one of the enablements of reading a user's mail is that you have to be that user, and that no plan achieves this enablement.

The final explanation, *Plan Thwarts User Goal*, suggests that the user is unaware that another plan achieves the user's goal without an additional effect that the user's plan has.

User: To list files whose names begin with a number, I pipe "ls" to "grep [0-9]".

Advisor: Use "ls [0-9]*" instead. It is more efficient.

The user's mistaken belief is that piping "ls" to "grep" is the most appropriate plan for listing files whose names begin with a digit. This belief is contradicted by the advisor belief that the plan of using "ls [0-9]*" not only achieves the user's goal but also does not thwart his other goal of using his time efficiently.

5.2. Explanations for an Incorrect Enables

Just as there are several different explanations for why the advisor does not share a user belief that a plan is applicable to a goal, there are also several different explanations for why the advisor does not share a user belief that a state is a precondition to a plan achieving a goal; that is, a user belief of the form **enables**(*Se*, *Ap*, *Sp*). These explanations are shown in Figure 2.

Class of mistake	Potential explanation	English description
Enablement For Subgoal	enables(S,Ap,Sp) enables(Se,A,S)	State that is actual enablement Is enabled by user's enablement
Not Always Enablement	enables(S,Ap,Sp) causes(A,Sp) !enables(Se,A,Sp)	Other state is necessary Plan achieves user's goal without claimed enablement
Enablement Too Specific	causes(A,Se) causes(A,S) enables(S,Ap,Sp) causes(Ao,S) !causes(Ao,Se)	User enablement results from action that achieves real enablement Other action achieves real enablement without user's enablement as effect

Figure 2. Potential explanations for incorrect enablement

The first, *Enablement For Subgoal*, explains the user's mistake in our introductory exchange. The explanation is that the user is unaware that his precondition is not a precondition of the goal itself, but of one of its preconditions.

User: So to remove a file, I have to own the directory that contains it.

Advisor: No, to remove a file, you need to have write permission on the directory that contains it. You do not need to own the directory that contains it. You need to own that directory when you do not already have write permission on it.

The user's belief is contradicted by the advisor's beliefs that owning a directory is a precondition for achieving write permission on it, and that having write permission is a precondition for removing a file.

The second explanation, *Not Always Enablement*, is that some plan achieves the user's goal without the user's claimed precondition.

User: So I can only edit files when I'm on a smart terminal?

Advisor: Only if you edit with "re". "vi" works fine on a dumb terminal.

The user's incorrect belief is that it is necessary to have a smart terminal to edit a file. This belief is contradicted by the advisor's belief that plans exist that allow him to edit but do not require a smart terminal.

The last explanation, *Enablement Too Specific*, is that the user's precondition is less general than the actual precondition for achieving his goal.

User: So I have to remove a file to create a file?

Advisor: You do not have to remove a file to create a file. You must have enough free space. Removing a file is only one way to obtain it. You could also ask the system administrator for more space.

The user mistakenly believes that it is necessary to remove an existing file before a new file can be created. This belief is contradicted by the advisor's belief that the precondition is sufficient space for the new file, which can be achieved either by executing a plan for removing a file or by executing the plan of requesting more space.

5.3. Explanations for Incorrect Causes

One final class of user misconception is an incorrect belief that a particular state results from a plan's execution; that is, a user belief of the form **causes**(*Ap*, *Sp*). The explanations for this class are shown in Figure 3.

Class of mistake	Potential explanation	English description
Effect Requires Enablement Unfulfilled	enables(S,Ap,Sp) !causes(A,S)	State required for Action and no way to achieve State
Effect Inferred From Other Effect	causes(Ap,So) precludes(So, S) precludes(Sp, S)	Action causes other effect That effect precludes a state that is precluded by user's effect
Effect From Another Plan	causes(Ap,So) causes(A, Sp)	Action used to cause other effect Some other action caused it

Figure 3. Potential explanations for incorrect effect

The first, *Effect Requires Unfulfilled Enablement*, suggests that the user is unaware that a particular state is required for the plan to have the claimed effect.

User: I was cleaning out my account when I accidentally deleted all the command files by typing "rm".

Advisor: You can't delete the command files with "rm" unless you are the system administrator.

The user incorrectly believes that typing "rm" removed several system files. The advisor's contradictory belief is that it is necessary for the user to be the system administrator for this effect to occur.

The second, *Effect Inferred From Other Effect*, accounts for another one of the earlier examples. It suggests that the user has inferred an incorrect plan effect from an actual plan effect.

User: I accidentally hit the up arrow key and it deleted 20 unanswered mail messages. How can I get them back?

Advisor: Hitting the up arrow does not delete your messages, but does result in your being disconnected from the etherplexer. You could not access your mail messages because they were moved to "mbox". The mail program requires that your mail messages be in "mailbox".

The user incorrectly believes that one effect of hitting uparrow was that his mail messages were deleted and are now inaccessible. This contradicts the advisor's belief that one effect of hitting uparrow is that files are moved to a different location, which makes them seem inaccessible.

The third, *Effect From Another Plan*, accounts for the following example. The explanation is that the user is unaware that the user's action actually has a different effect.

User: I deleted a file by typing "remove".

Advisor: No, typing "rm" deleted it. Typing "remove" deletes a mail message from the mail program.

The user's mistaken belief is that typing "remove" deletes a file. This contradicts the advisor's belief that typing "remove" actually throws away old mail messages.

6. A Detailed Process Model

When presented with a potentially incorrect user belief, the advisor must answer two questions: Is the belief mistaken? And, if so, why does the user believe it? To do so, the advisor first tries to determine whether he holds the user's belief. If he does not, the advisor assumes the user's belief is mistaken and tries to find an explanation for why the user holds this belief. The taxonomy guides this search for an explanation.

6.1. Finding an Explanation

The advisor finds an explanation by hypothesizing and trying to confirm one of the potential explanations associated with the user's type of belief. The advisor

1. Classifies the user's belief according to its type: **applies**, **enables**, or **effects**.
2. Selects one of the potential explanations associated with that class of belief.
3. Instantiates this particular potential explanation with information from the user's belief.
4. Tries to verify each of the planning relationships within the potential explanation. If all can be verified, this potential explanation is the desired explanation.
5. Repeats the process until one of the potential explanations associated with this belief's type is verified or all potential explanations have been tried and have failed.

The result of this process is that the advisor has verified that he holds a particular set of beliefs that contradict the user's belief. These beliefs constitute the desired explanation.

6.2. Verifying an Advisor Belief

The advisor's verifying whether or not he holds beliefs about particular planning relationships constitutes an important part of finding an explanation. To verify a belief, the advisor first searches his memory for the desired piece of planning knowledge. If it isn't found, the advisor then tries to deduce that the relationship holds. Once the advisor has proved that the planning relationship holds, either by search or by reasoning, that piece of knowledge is noted to be an advisor belief.

Consider, for example, the process of verifying that the advisor holds a belief that owning a directory is not a precondition of removing a file. If this fact is already known from past experience, the advisor will recognize it during memory search. If not, the advisor can try to deduce it. One rule that applies here says that "if a state S is not one of the known states that are preconditions to an action A for achieving a goal state, then assume that *S* is not a precondition." Here this means that if owning a directory is not among the known preconditions for removing a file, assume it is not a precondition for removing a file. In effect, this is the closed world assumption: If the advisor cannot show that a belief holds, assume its negation.

6.3. An Example

This section shows how the advisor processes the user belief that owning a directory is a precondition of removing a file.

enables(user own directory, use "rm", the file's removal)

The advisor first determines whether he holds this belief. He verifies that he does not by noting that owning a directory is not a known precondition of removing a file and applying the closed-world assumption. The advisor then tries to

explain why he does not hold this user belief. He notices that this belief can be classified as a belief that some state *Sp* (owning the directory) is a precondition to achieving some other state *Sg* (removing a file). The taxonomy provides several potential explanations for this type of belief. One potential explanation is that *Sp* is actually a precondition of achieving a state *S*, which is a precondition to achieving *Sg*. In this case, instantiating *Sp* and *Sg* leads the advisor to try to verify that he holds these two beliefs:

enables(*S*, use "rm file", the file's removal)
enables(owning a directory, *A*, *S*)

These beliefs are verified when the advisor finds that having write permission on a directory is a precondition to removing a file, and that owning a directory is a precondition to obtaining write permission on the directory (the advisor verifies these beliefs either by finding the belief relationships in memory or by deducing them). These matching beliefs contradict the user's belief, confirming it as a misconception and providing a possible explanation for why the user holds it. Finally, the advisor presents the resulting beliefs to the user.

6.4. The Point of Potential Explanations

Having a taxonomy of potential explanations lessens the amount of reasoning the advisor must do to detect and respond to user misconceptions.

To see why, consider an advisor trying to understand how the user arrived at the mistaken belief that a precondition of removing a file is owning the directory containing it. The advisor is trying to find some connection between the user's enablement and removing a file. The potential explanations suggest how to find specific, likely-to-be-useful connections. For example, the potential explanation *Enablement for Subgoal* suggests examining whether achieving any of the preconditions of removing a file requires owning a directory.

Without a set of potential explanations it becomes necessary to reason from a set of rules that describe likely differences between user and advisor beliefs. One rule might be that a user may incorrectly attribute an enablement of one action to another action. Another rule might be that a user may incorrectly attribute the result of one action to another action. From a set of such rules the advisor must somehow deduce the cause of the user's mistake. The potential explanations turn the problem of finding an explanation into one of guided memory search rather than reasoning from first principles.

7. Accessing Advisor Planning Knowledge

The advisor verifies potential explanations by searching his planning knowledge for the appropriate belief relationships. The standard way of organizing planning knowledge is a database in which each plan is associated with applicability conditions, enablements, and effects. Using our representation for planning knowledge,

this corresponds to having each plan associated with a set of **applies**, **enables**, and **causes** beliefs. For example, some beliefs about "rm" are that "rm" is applicable to removing a file; that two enablements of using "rm" to remove a file are having write permission on the directory containing the file, and making sure that the file is not a currently executing program; and that one effect of using "rm" to remove a file is that the file is removed. The advisor has similar sets of beliefs about other plans. As demonstrated earlier, however, these beliefs – which essentially capture the advisor's knowledge about planning successes – are not sufficient. The advisor also needs knowledge about planning failures.

7.1. Organizing Knowledge About Planning Failures

The advisor has two types of knowledge about planning failures associated with each plan. The first type involves negated plan-oriented beliefs and the explanations for why these beliefs are held. These beliefs involve goals to which a plan does not apply (**!applies**), states that do not enable a plan's achieving a goal (**!enables**), and the states that do not result from an action's execution (**!causes**). This means that along with the above knowledge about "rm", the advisor has:

1. *Beliefs about attempts to use "rm" to achieve other goals, and explanations for why those attempts failed.* The advisor may believe that "rm" is not an applicable plan for removing a directory. The advisor's explanation for this belief is that he believes "rm" has no effect that removes a directory and is instead used to remove files, and that another command, "rmdir" is used to remove a directory (*Plan Achieves Different Goal*).
2. *Beliefs about states that are not enablements of "rm", and explanations for why those states are not necessary.* The advisor may believe that a file that is being displayed can be removed. The explanation for this belief is that the advisor believes that he has previously removed a displayed file and that the enablement is instead simply that the file cannot be an executing program (*Not Always Enablement*).
3. *Beliefs about states that do not result from "rm", and explanations for why those states do not occur.* The advisor may believe that "rm" does not query the user before removing a file. The explanation for this belief is that "rm" is known to produce no output, and that "rm -i" queries the user before removing the file (*Effect From Another Plan*).

These beliefs and explanations are stored along with the rest of the advisor's knowledge of each plan. Accessing these stored beliefs allows the advisor to immediately recognize when a user's belief is contradicted without doing any reasoning. The explanation patterns are used only if a canned explanation cannot be found.

The other type of knowledge associated with each plan describes the results of failing to hold correct beliefs when the plan was used. These include beliefs about what happens when a plan is applied to an incorrect goal, or is applied without various plan enablements being achieved, or produces unexpected effects. This information is maintained as lists of effect/failed-enablement, effect/failed-application pairs, and effect/related-effect pairs. Some of these belief pairs for "rm" include

the belief that a "permission denied" message results from failing to have write permission on the directory containing the file; that a "file is a directory" message results when "rm" is incorrectly applied to the goal of removing a directory; and that a "permission denied" message means that the file was not removed. This organization means that an advisor given an unusual plan effect can retrieve the failed enablement or incorrect plan application that led to it, as well as any other effects that are likely to have occurred with it.

7.2. Retrieving Advisor Planning Knowledge

Although the advisor possesses a great many beliefs, only a few apply to the potential explanation for any particular incorrect user belief. "rm", for example, has many enablements: the user must have write permission on the directory containing the file, the file being removed cannot be a program in use, the file system containing the file must be mounted, and so on. But if the advisor knows the user's use of "rm" resulted in a permission denied message, only the first enabling condition is relevant, and the other enablements should be ignored. On the other hand, if the advisor knows the result was the message "text file busy", only the second condition is relevant. The point is that somehow the advisor must be able to determine which of these beliefs are relevant to his search for an explanation. Ideally, when the advisor verifies beliefs that are part of any potential explanation, he first examines the likely-to-be-relevant beliefs, and then his more general database of planning knowledge.

To find a set of relevant advisor beliefs, the advisor examines each of the beliefs in the user model. If a user belief corresponds to one of the plan's standard advisor beliefs (that is, beliefs about the plan's usual applicability conditions, enablements, or effects), the corresponding advisor belief is added to the set. The user belief above that "rm" is used to remove a file means that the similar advisor belief is relevant. Otherwise, the advisor uses the user belief as an index into the plan's failure-oriented advisor beliefs, and adds any indexed beliefs to the set. The user belief above that "rm" printed a permission denied message indexes the advisor beliefs that having write permission on the directory containing the file is an enablement to removing a file, which implies that these beliefs are relevant.

7.3. An Example

Consider the user model we developed earlier for the first exchange in our example dialog.

applies(use-rm, remove a file)
!causes(use-rm, file removed)
causes(use-rm, permission denied)
enables(own file, use-rm, remove a file)

The advisor constructs a set of relevant advisor beliefs by using each of these beliefs as indices into his knowledge base. The user's belief that "rm" is used to

remove a file is a standard advisor belief, and is therefore automatically added to the set. The user's belief that "rm" does not cause a file to be removed is not a standard belief, and is used to try to index other advisor beliefs. But it does not index any beliefs. The user's belief that "rm" causes a permission denied belief is also not a standard advisor belief and it, too, is used to try to index other advisor beliefs. It succeeds, indexing a pair of beliefs: that having write permission enables removing the file (effect/failed-enablement), and that when "rm" prints permission denied, the file is not removed (effect/related-effect). Finally, the last user belief, that owning a file enables removing a file, is used as an index but retrieves no beliefs. The result is that the following advisor beliefs are marked relevant.

applies(use-rm, remove file)
enables(write directory, use-rm, write permission on a directory)
!causes(use-rm, file removal)
causes(use-rm, permission denied)

This means that when the advisor later tries to verify various potential explanations, write permission will show up as the relevant enablement.

7.4. Related Work in Memory Organization

Unlike traditional planning systems, our advisor uses knowledge of past planning failures. Our organization of this knowledge is based on earlier work in organizing past experiences and indexing them appropriately for efficient search and retrieval [KOLO86, KOLO84, SCHR82]. These systems organized knowledge of a class of episodes around the differences between each episode and the class prototype. For example, specific diplomatic trips were indexed around general knowledge of diplomatic trips by differences in location, participants, purpose, and so on. A particular diplomatic experience was accessed by providing its location, participants, purpose, or some combination of these features.

Our approach is similar. A plan's normal behavior corresponds to the prototypic episode, and its different plan failures correspond to the other episodes. Specifying some information about the failed episode (such as its results) allows the advisor to retrieve its cause (the particular failed enablement or bad plan application).

8. Related Work

Two approaches have been used to detect and correct misconceptions. The first approach is used by many intelligent tutoring systems. These systems [ANDJ85b, BROW78, BURT82, STEV82] locate mistaken beliefs in a database of error-explanation pairs and provide the associated explanation. A basic problem with this approach is that, because there is no information about the underlying causes of the errors, these systems can handle only those misconceptions known in advance.

The other approach avoids the difficulty inherent in enumerating all possible misconceptions within a domain by using strategies that address an entire class

of misconceptions. The user's misconception is classified according to the abstract reasoning error likely to have led to it. This approach shares many features with recognizing abstract thematic situations (such as irony) in narratives, where such situations are defined in terms of abstract planning errors made by the narrative characters [DYER88, DOLA86, DYER83]. Once an appropriate strategy is found, it can be used to generate advice (in narratives, this advice may be in the form of adages). In advisory systems, this approach has been applied to both object- and plan-oriented misconceptions.

8.1. Object-Oriented Misconceptions

ROMPER [MCCO*] corrects user misconceptions dealing with whether an object is an instance of a particular class of objects or possesses a particular property.

User: I thought whales were fish.

ROMPER: No, they are mammals. You may have thought they were fish because they are fin-bearing and live in the water. However, they are mammals since, while fish have gills, whales breathe through lungs and feed their young with milk.

ROMPER classifies a user's misconception as either a misclassification or misattribution and then selects one of several strategies associated with each class of misconception to generate a response. Each strategy addresses a different type of reasoning error, and is selected based on ROMPER's own beliefs about objects and its model of the user's relevant beliefs. One such strategy is useful when the advisor believes that *X isa Z*, the user mistakenly believes that *X isa Y*, and the advisor believes that *X* and *Y* share certain attributes. The strategy suggests presenting these shared attributes as a possible reason for the misclassification, and pointing out the unshared attributes that lead the advisor to believe that *X isa Z*.

Despite dealing with a very different class of misconceptions, ROMPER's approach is similar to ours.

8.2. Plan-Oriented Misconceptions

Two efforts have examined detecting and responding to plan-oriented misconceptions.

Joshi, Webber, and Weishedel [JOSH84a] suggest using a strategy-based approach to provide cooperative responses to problematic planning requests. They consider "How do I do *X*?" questions in which *X* can be inferred to be a subgoal of a more important goal *Y*.

User: How can I drop cs577?

System: It is too late in the quarter to drop it. But you can avoid failing by taking an incomplete and finishing your work next quarter.

They provide several strategies, listed below, for selecting the contents of a reasonable response, with strategy selection based on the advisor's beliefs about which plans achieve a particular goal and the achievability of their preconditions.

	Situation	Response
(1)	Unachievable Precondition E of X	Provide E
	(a) Plan P achieves Y	Provide P
	(b) No plan to achieve Y	Point this out
(2)	X doesn't help achieve Y	Point this out
	(a) Plan P achieves Y	Provide P
	(b) No plan to achieve Y	Point this out
(3)	Plan P better way to achieve Y	Provide P
(4)	X only way to achieve Y	Point this out
(5)	Plan P involving uncontrollable event E achieves Y	Provide P

One such strategy, which is useful when the advisor believes that *X* cannot be achieved because of an impossible-to-achieve precondition, is to point out the troublesome precondition and suggest an alternate plan that achieves *Y*.

Our work differs from theirs in several respects. The main difference is that they focus on correcting the user's misconception instead of trying to explain why it occurred. Only one strategy above is concerned with providing an explanation that addresses the source of a user misconception (in this case, an inappropriate plan). The other strategies describe situations in which achieving X is inappropriate and an alternate plan for Y exists and should be presented to the user as a correction. In addition, they do not consider responding to incorrect beliefs about plan preconditions or effects.

The other effort, SPIRIT [POLL86a, POLL86b] tries to detect the inappropriate plans underlying queries made by users of a computer mail program and the mistaken user beliefs underlying those plans.

User: I want to prevent Tom from reading my file. How can I set the permissions on it to faculty-read only?

System: You can make the file readable by faculty only using "set permission". However, Tom can still read it because he's the system administrator.

User misconceptions about the applicability and executability of plans are detected by reasoning about the likely differences between the advisor's beliefs and the user's, with various rules used to infer these differences. One such rule, used to detect the source of the misconception above, states that an advisor who believes that an act has a particular result under certain conditions can infer that the user has a similar belief missing one of the required conditions.

SPIRIT has a task similar to ours but takes a very different approach, trying to determine the cause of the user's error through reasoning from first principles rather than memory search. In addition, SPIRIT cannot detect or respond to mistakes involving plan applicability conditions or preconditions. Finally, SPIRIT does not specify how knowledge of the cause of the user's mistaken belief affects the information to be included in a cooperative response, something that naturally is beyond our model.

8.3. UNIX Advisors

Finally, there are two other related research efforts, UC [WILE86, WILE84] and SC [KEMK86], that address providing advice to novice UNIX users. Neither system, however, detects or responds to misconceptions. Instead, both are concerned with tailoring a response to a question to reflect the user's level of expertise. UC's user modeling component, KNOME [CHIN*], analyzes a user's questions to determine which stereotypical class the user belongs to and then uses this information to provide more details and possibly more examples to less experienced users.

Novice: What does the "rwho" command do?

UC: Rwho lists all users on the network, their tty, and their login and idle times.

Expert: What does the "rwho" command do?

UC: Rwho is like who, except rwho lists all users on the network.

SC's user modeling component, SCUM [NESS87], takes an approach similar to UC's, also using stereotypical information. These approaches are complementary to ours.

9. Implementation Details

The theory discussed in this chapter is implemented in AQUA, a computer program under development at UCLA. The latest version of AQUA is implemented in Prolog and runs on Apollo workstations. Given a set of user beliefs involving plan applicability conditions, preconditions, or effects, AQUA determines which of these user beliefs are incorrect and what missing or mistaken user beliefs are likely to have led to them, and then produces a set of advisor beliefs that capture the content of the advisor's response. AQUA's domain of expertise is in the basic plans used to manipulate and access files, directories, and electronic mail. It has been used to detect and respond to at least two different incorrect user beliefs in each class of misconception that we have identified.

10. Limitations and Future Work

Our approach to determining why a user has a particular incorrect belief has been to let potential explanations direct the search for the advisor beliefs that serve as an appropriate explanation. Our focus has been on discovering and representing these explanations. The limitations of our approach arise in areas we have ignored, each of which is an interesting area of research.

10.1. Building the User Model

Most of our work has involved automatically inferring the sources of mistaken user beliefs, given an initial set of user beliefs. Unfortunately, users rarely explicitly state their beliefs, leaving the advisor the difficult task of inferring them. Consider our introductory exchange.

User: I tried to remove a file with the "rm" command. But the file was not removed and the error message was permission denied. I checked and I own the file. What's wrong?

Advisor: To remove a file, you need to be able to write into the directory containing it. You do not need to own the file.

Here the advisor must infer the user's beliefs that (1) using "rm" is applicable to removing a file, that (2) using "rm" did not cause the file's removal, that (3) using "rm" resulted in an error message, and that (4) owning a file is a precondition to removing it. How can these beliefs be inferred, assuming that their components (plans, goals, and state descriptions) have been recognized?

Our approach to recognizing these components has been to apply a set of belief inference rules to each mentioned or inferred plan, goal, and state. Each of these rules describes a situation to check for, and suggests a user belief that may be inferred in that situation. For example, one such rule is "if the user executes a plan, and has achieving a particular state as his goal, assume the user believes that plan applies to that state." This rule helps infer the first user belief from the user's executing "rm" to try to remove a file. The second belief is inferred from the rule that "if an utterance describes the nonexistence of a state that is a believed result of an action, assume that the user believes that the action did not cause the state." A similar rule helps infer the third belief.

Inferring the final belief, that owning a file is a precondition to its removal, is a difficult task. Because there are a potentially-infinite number of incorrect user beliefs about the preconditions of removing a file, the advisor cannot simply match owning a file against a list of incorrect preconditions. Because the user may have been discussing other plans and other goals the advisor cannot simply assume that any utterance after a plan's failure refers to its preconditions. Instead, the advisor must infer this user belief from the knowledge that the user did some sort of verify-action, the knowledge that one plan for dealing with a plan failure is to try to verify that the enablements of the plan have been achieved, and the knowledge that both owning the file and having write permission are different instantiations of having sufficient permission. The rule used to infer this belief is "if the user tries to verify a state S, and both S and a known precondition are instances of a more general state, assume that the user believes that S is a precondition of the user's goal."

These rules are useful and provide a feel for how different user beliefs can be recognized – making more plausible our assumption that an initial user model exists – but they fall far short of being a serious process model for belief recognition. Creating such a model is one of our most pressing research concerns. Recognizing these components of these beliefs (the user's plans and goals) is also an active area of research [CARB*, KAUT86, GOOM86, WILE86].

10.2. Other Classes of Misconceptions

We are currently studying how well the classes of misconceptions described here account for responses to misconceptions in domains other than the problems of novice computer users, such as the domain of simple day-to-day planning. In addition, we are examining other classes of planning misconceptions. For example, to respond to an incorrect user belief such as "rm" cannot be used to remove a file, the advisor needs potential explanations for why an action does not apply to a particular goal state.

We do not yet know whether our approach is suitable for generating responses to misconceptions that are not directly related to plan-goal interactions, such as mistakes in referring to an object. Consider the following exchange:

User: Diana is up but I cannot access my file.
Advisor: Your files are on Rhea, not Diana. They moved your files yesterday because your file system was full.

Here the user's problem is that he is incorrectly using "Diana" to refer to the machine his files are on. We are examining whether our approach is extendable to respond to these types of user misconceptions.

10.3. Response Generation

The response we provide is a set of advisor beliefs that is as complete as possible. We make no attempt to use knowledge about other user beliefs to modify our response to provide only the most relevant beliefs. However, if the advisor can infer that a user knows that his plan has failed (perhaps because of the error message a command produces), he need not inform the user that his plan is incorrect. One straightforward way to extend our model is to have the advisor filter out those beliefs he can infer the user has.

The advisor should use information about the user to tailor the response based on the user's level of expertise. Recall the following exchange.

User: I tried to remove my directory and I got an error message "directory not empty". But "ls" didn't list any files.
Advisor: Use "ls -a" to list all of your files. "ls" cannot be used to list all of your files because "ls" does not list those files whose names begin with a period.

An advisor who knows the user is a novice might want to augment his response with an explanation that "-a" is a command option and that command options cause changes in the normal behavior of commands. Several researchers are working on tailoring the response to the user based on knowledge about the user's expertise [CHIN*, PARI*].

10.4. Acquiring and Using Planning Knowledge

Currently, to verify a potential explanation, the advisor searches his memory for relevant beliefs. While we have specified the types of planning knowledge the advisor needs and suggested how this knowledge should be organized to facilitate rapid retrieval of relevant beliefs, we have not shown how this information can be automatically acquired and appropriately organized. Similarly, we have assumed the advisor possesses an abstract taxonomy of potential explanations for planning failures, but have not specified how this taxonomy is acquired. Initial work in recognizing and indexing abstract configurations of planning relations is discussed in [DOLA86, DOLA85].

In addition, the advisor may be required to verify a belief that he does not already possess somewhere in memory. Suppose the advisor is trying to verify that write permission is needed to remove a file. The advisor may be able to deduce this from his beliefs that write permission is needed to make changes on objects and that removing a file involves making a change to a directory. This requires more powerful reasoning capabilities than simple memory search.

11. Conclusions

We have presented an explanation-based approach to the problem of recognizing and responding to user misconceptions. The advisor determines the source of an incorrect user belief by finding an explanation for why he does not hold the user's belief. The explanation is found by hypothesizing and trying to verify a small set of potential explanations associated with each type of user belief. In essence, the model uses information about likely sources of different classes of user misconceptions to recognize user mistakes and infer their underlying causes.

This approach is attractive for several reasons. First, because it has information about classes of abstract misconceptions, it can handle misconceptions of which it has no prior knowledge, as long as they fall into one of these classes. Its small set of potential explanations account for a large number of specific user mistakes. Second, the model circumvents the need for general deductive reasoning by using knowledge of the types of misconceptions users are likely to make. However, the model can also use its knowledge of specific misconceptions and their sources to rapidly produce responses to misconceptions it has seen before, as do tutoring systems. In addition, the system can easily be augmented to use general reasoning when it is confronted with a misconception that cannot be explained by any of its potential or canned explanations, as does SPIRIT. Finally, and perhaps most importantly, our approach leads to responses similar to (and sometimes more informative than) those of the many UNIX advisors we have observed.

12. References

[ABEL73], [ABEL79], [ALVA86], [ANDJ85b], [BROW78], [BURT82], [CARB*], [CHIN*], [DOLA85], [DOLA86], [DYER83], [DYER88], [FIKE71], [FLOW82], [GOOM86], [JOSH84a], [KAUT86], [KEMK86], [KOLO84], [KOLO86], [MCCO*], [NESS87], [PARI*], [POLL86a], [POLL86b], [QUIL85], [QUIL88], [SACE74], [SCHR77], [SCHR82], [SCHR86], [STEV82], [WILE83], [WILE84], [WILE86]

6 Plan Recognition and Its Use in Understanding Dialog [1]

Sandra Carberry

Abstract [2]

Analysis of naturally occurring dialog suggests that human information-providers often use knowledge acquired from the dialog to understand subsequent utterances and provide cooperative, helpful responses. One of the most important ways in which an information-seeking dialog can be assimilated is by inferring the task-related plan motivating an information-seeker's queries. This chapter presents two process models. The first dynamically constructs a model of an information-seeker's plans and goals from an ongoing dialog, and the second reasons on this component of a user model to understand one class of problematic utterance. The chapter further addresses the problem of detecting and recovering from discrepancies between the system's model of the user and the actual plan under construction by the user. It proposes a four-phase approach to detecting and recovering from disparate models, and argues for an enriched user model that differentiates among its components on the basis of the support it accords each component as a correct and intended part of the user's plan.

1. Introduction

Analysis of naturally occurring dialog suggests that when two individuals participate in an information-seeking dialog, the information-provider often uses the information exchanged during the dialog and her knowledge of the domain to hypothesize a model of the speaker, and that this model is adjusted and expanded as the dialog progresses. There are many ways in which an information-seeker can be modeled, but one of the most significant of these is the inference of the underlying task-related plan motivating the information-seeker's queries. It appears that human information-providers often attempt to infer this plan and use their acquired knowledge to interpret subsequent utterances and provide useful responses.

---

[1] This chapter is a revised and extended version of [CARB88]

[2] Some of this work was partially supported by a grant from the National Science Foundation, IST-8311400, and a subcontract from Bolt Beranek and Newman Inc. of a grant from the National Science Foundation, IST-8419162.

We are interested in a class of information-seeking dialogs in which the information-seeker is attempting to construct a plan for a task that will be executed after the dialog terminates. These dialogs are representative of a large percentage of interactions with database management systems, decision support systems, and expert systems. Typical tasks include expanding a company's product line, purchasing a home, or pursuing a degree at a university. One aspect of our research has been to develop a strategy for dynamically constructing a model of an information-seeker's underlying task-related plan from an ongoing dialog and for tracking his focus of attention in this plan. Since the system's beliefs about the user's plans and goals provide a context for understanding subsequent dialog, we will refer to this part of a user model as a *context model.*

In an information-seeking dialog, the information-provider is engaged in helping the information-seeker construct a plan for his task. The information-seeker must communicate whatever information he believes necessary for the information-provider to fulfill her helping role. But naturally occurring dialog is both imperfect and incomplete. A speaker may inadvertently use incorrect terminology in constructing the language representation of his intended query, or he may short-cut its complete specification. Our analysis of naturally occurring dialog indicates that human information-providers often use knowledge acquired from the dialog to understand many of an information-seeker's faulty utterances and allow the dialog to continue without interruption. We believe that a natural language system should operate in a similar manner since, if its communication is to be regarded as natural, the system must be able to handle the full spectrum of utterances that humans understand with relative ease.

Two kinds of utterances that fall in this category are those that are imperfect and those that are incomplete. Part of this research has been concerned with investigating how a user model can aid in understanding these. We have developed a method, based on Grice's theory of meaning and maxim of relevance, for handling pragmatically ill-formed utterances. It uses the task-related plan inferred for an information-seeker to suggest variants of an ill-formed utterance that might represent his intentions or at least satisfy his perceived needs. We have also developed a method for understanding intersentential elliptical utterances that occur during the course of an information-seeking dialog. Our strategy uses discourse expectations and our model of the speaker's plan to identify the discourse goal that he is pursuing via an elliptical fragment and to interpret his elliptical utterance relative to his task-related plan [CARB85].

Section 2 of this chapter briefly reviews related work in user modeling, and Section 3 presents our strategy for dynamically inferring a model of the user's plan from an ongoing dialog. Section 4 describes our mechanism for reasoning on this user model to understand utterances that are problematic for current natural language systems. These ideas have been implemented and tested in the IREPS system (Intelligent REsPonse System). This is an ongoing research effort whose objective is a robust natural language interface to information systems. The component that infers the user's underlying task-related plan is called TRACK. To move from one domain to another, only the corpus of domain-dependent plans and goals must be reconstructed; the heuristics and processing strategies are completely transportable.

Section 5 describes our current research on relaxing restrictive assumptions present in previous work on plan recognition and developing a more robust plan inference framework. We present a four-phase approach to handling the problem of possible disparity between the system's context model and the actual plan under construction by the user, and argue for an enriched model that differentiates among its components according to the support it accords each component as a correct and intended part of the user's plan. Section 6 relates our research on plan recognition to other work on user modeling.

Throughout this chapter, the information-seeker and information-provider will be referred to as IS and IP respectively.

2. Plan Recognition in Dialog Systems

Early work in dialog understanding concentrated on apprentice-expert dialogs, during which an expert guided an apprentice in performing a task. Grosz [GROS77a] formulated heuristics for recognizing shifts in focus of attention in the task structure and presented a strategy that used the knowledge currently focused on by the dialog participants to identify the referents of definite noun phrases appearing in an utterance. Robinson [ROBI80, ROBI81] extended Grosz's work and developed a process model for determining the referents of verb phrases, such as variants of *do* in the utterance

"I've done it."

However, in apprentice-expert dialogs, the overall task that the apprentice is attempting to perform is known at the outset of the dialog, and the ordering of actions and subtasks in the plan being executed strongly influences the dialog between expert and apprentice. This differs from the kind of information-seeking dialog that we are investigating, in which the information-seeker is attempting to construct a plan for a task that will be executed after the dialog is complete. In such dialogs, the information-seeker's utterances are not tightly constrained by the order in which actions in the task will eventually be executed, as in apprentice-expert dialogs.

Allen [ALLE80, PERR80] inferred the goal underlying a speaker's utterance in the context of an information agent in a train setting. This inferred goal was used to account for extra helpful information included in the agent's response, and the inference path connecting the speaker's utterance and the inferred goal was used to interpret indirect speech acts. However, Allen's domain was very restricted; the only domain goals were *meeting a train* and *boarding a train*, each of which could be accomplished by a few primitive steps, and his system was primarily concerned with utterances that might occur at the outset of a dialog. In more complex domains, the information-seeker's complete plan will consist of a hierarchy of subplans and subgoals which accomplish his overall goal. Such a complete plan is not immediately evident from a single utterance, and individual utterances must be related to one another to build the user's plan as the dialog progresses.

Sidner [SIDN83b, SIDN85] and Litman [LITM86] developed enhanced models of plan inference. However, both were concerned with dialogs that were initiated

in order to begin or continue execution of an underlying task (display of structures on a graphics terminal and meeting/boarding a train), and the dialogs were therefore constrained by the order in which individual actions in the task must be executed. In addition, Sidner investigated how discourse markers aid in recognizing the speaker's intent, and Litman studied a meta-plan framework and its use in handling clarification subdialogs.

Pollack [POLL86a] has recently proposed that plans be viewed as mental phenomena. She contends that, in order to comprehend an utterance and relate it to the user's plan, the system must reason about the configuration of beliefs and intentions that it should ascribe to the user. This will be discussed further in Section 5.

3. Inferring and Modeling the Task-Related Plan

In order to reason about what the user wants to accomplish, the system must have knowledge about the goals that a user might pursue in a domain and plans for accomplishing those goals. We view a plan as the means by which an agent can accomplish a non-primitive task-related goal. Using an extension of the STRIPS formalism [FIKE71, ALLE80], we represent a plan as a structure containing applicability conditions, preconditions, a plan-body, and effects. Applicability conditions and preconditions both represent constraints that must be satisfied before a plan can be executed. However, an agent can plan to satisfy preconditions, whereas it is anomalous to plan to satisfy applicability conditions; the latter determine whether it is reasonable to even consider a particular plan for achieving a desired goal. The plan-body contains a set of subgoals and the effects represent the results of successfully executing the plan. Arguments in plans are either constants, represented as uppercase strings, or typed variables, represented as lowercase strings preceded by the character "_" and followed by the characters ":&" and an uppercase string giving the variable's type. Figure 1 presents a sample plan used by TRACK. These plans are hierarchical, since many of the subgoals in a plan are non-primitive and therefore have associated plans which may be substituted for them. Thus a plan can be expanded to any desired degree of detail.

```
Learn-Material(_agent:&PERSON, _sect:&SECTIONS, _syl:&SYLLABI)
    Plan-Body:
        Learn-From-Person(_agent:&PERSON, _sect:&SECTIONS, _fac:&FACULTY)
            where Teaches(_fac:&FACULTY, _sect:&SECTIONS)
        Learn-From-Text(_agent:&PERSON, _txt:&TEXTS)
            where Uses(_sect:&SECTIONS, _txt:&TEXTS)
    Effects:
        Learn-Material(_agent:&PERSON, _sect:&SECTIONS, _syl:&SYLLABI)
```

Figure 1. Plan for learning course material

At the outset of an information-seeking dialog, the system has little knowledge about the information-seeker's (IS) purpose in requesting information. In most cases, a complete plan for IS cannot be constructed during the first part of a dialog. Instead, potential goals must be inferred from individual utterances and integrated into the overall plan structure, thereby incrementally expanding and instantiating the plan as the dialog progresses.

Oftentimes there will be many domain goals that a single utterance might address. For example, if a student wants to know what time Dr. Smith arrives in the morning, he may want to either visit Dr. Smith or call him in his office. Furthermore, even if we can identify a single domain goal addressed by an utterance, there may be many ways in which that goal could be incorporated into an overall plan. For example, if a student asks whether Political Science 210 is offered in the spring, we might infer that the student wants to take Political Science 210. But how should this goal be built into the student's overall plan? Perhaps he is considering taking Political Science 210 in order to satisfy a breadth requirement or perhaps he wants to earn a minor. So the issue that must be addressed in dynamically inferring IS's underlying task-related plan from an ongoing dialog is the following: *how can we identify which of many candidate goals is the actual goal which IS is addressing with a particular utterance, and how can we determine where this particular goal fits into IS's overall plan?*

Two factors appear to provide a basis for a solution. The first is the cooperative nature of the dialogs we are studying. The dialog participants are expected to maintain a meaningful exchange, during which the relationship of individual utterances is explicitly or implicitly conveyed. The second factor is the organized nature of naturally occurring information-seeking dialogs, as exhibited in dialog transcripts. These two factors produce a structure in information-seeking dialogs. As a result, we can formulate focusing heuristics that specify how individual utterances should be related to the existing dialog context, as represented by the plan inferred for IS and his current focus of attention in that plan.

Thus our approach is the following:

1. Hypothesize from an individual utterance a set of domain-dependent candidate focused goals that may represent the information-seeker's focus of attention in the task.
2. Use focusing heuristics to select the candidate focused goal most appropriate to the existing dialog context and incorporate it into the model of the information-seeker's plan.

3.1. Hypothesizing Candidate Focused Goals And Plans

The first stage of processing analyzes an utterance without considering the preceding dialog. Plan-identification heuristics are used to hypothesize a set of domain-dependent goals and associated plans that might represent that aspect of the task on which IS's attention is currently focused. These heuristics are extensions of inference rules proposed by Allen [ALLE80]. For example, if IS wants to know the values of an argument that cause a proposition to be true, then that proposition or its

negation may be relevant to the plan that IS is considering. Therefore any goals whose plans might prompt such a request become *candidate focused goals* and their associated plans become *candidate focused plans*. Thus, if IS asks,

"Who is teaching Section 10 of French 112 in the fall?"

then IS wants to know the values of the argument _fac:&FACULTY such that the proposition

Teaches(_fac:&FACULTY, FRENCH112-10-FALL)

is true. The plan for learning the material of a section of a course (Figure 1) contains the proposition

Teaches(_fac:&FACULTY, _sect:&SECTIONS)

as a constraint on a subgoal in its body. Substituting FRENCH112-10-FALL for _sect:&SECTIONS produces the proposition

Teaches(_fac:&FACULTY, FRENCH112-10-FALL)

addressed by IS's utterance. Making this substitution throughout the plan in Figure 1 produces a plan for learning the material of Section 10 of French 112 in the fall. Therefore the goal

Learn-Material(IS, FRENCH112-10-FALL, _syl:&SYLLABI)

becomes a candidate focused goal, and the plan produced by substituting FRENCH-112-10-FALL for _sect:&SECTIONS in Figure 1 becomes a candidate focused plan. The goal

Learn-From-Person(IS, FRENCH112-10-FALL, _fac:&FACULTY)
where
Teaches(_fac:&FACULTY, FRENCH112-10-FALL)

is the most recently considered subgoal in this candidate focused plan; thus it provides the greatest expectations for future utterances.

3.2. Context Processing

The second stage relates an utterance to the context established by the preceding dialog. We use a tree structure called a *context model* to represent the task-related plan inferred for IS from the preceding dialog. Each node in this tree represents a goal that the system believes IS is considering achieving and, except for the root, is a descendant of a higher-level goal whose associated plan contains the subgoal represented by the child node. One node in the tree is marked as the current focus of attention and indicates that aspect of the task on which IS's attention is currently centered. The path from the root of the context model to the current focus of attention is called the *active path* and represents the global context, or sequence of progressively lower-level goals that led to the subgoal under consideration by IS.

Initially, there is no existing context; each candidate focused goal and its plan become the root of a context model and are marked as current focused goals and

plans. If there is only one context model and its root goal appears as part of only one domain-dependent plan, then we have further knowledge about what IS wants to do and can add this higher-level plan as the new root of the context model, with the old root as its child. We continue expanding the tree upward until more than one higher-level plan is possible. Thus only that part of the user's task-related plan that the system believes IS intended it to recognize is built into the user model. Section 5 discusses more robust user modeling, in which default inference rules are used to expand the system's model of the user's plan, and addresses the problem of detecting and recovering from errors that might be introduced into the model.

As each new utterance occurs, it must be related to the established context. A set of focusing heuristics are used to determine the most likely relationship between one of the hypothesized candidate focused plans and the context model, and to expand the context model to include it. Grosz [GROS77a] introduced the concept of focusing in her work on identifying the referents of definite noun phrases in apprentice-expert dialogs. She noted that the focus of the discourse followed the plan for performing the apprentice's task. Our information-seeking dialogs differ from apprentice-expert dialogs in that the dialog is not constrained by the order of execution of the actions in the overall plan. However, we do find structure in the dialogs we are studying, and it is the basis for our focusing heuristics. This structure appears to be caused by two factors.

The first is the organized nature of naturally occurring information-seeking dialogs. Dialog transcripts indicate that humans generally ask all their questions that are relevant to a plan for one subgoal before moving on to ask questions about a plan for another subgoal of the overall task. One possible explanation for this behavior is that it may require less mental effort than switching back and forth among partially constructed plans for different subgoals.

The second factor producing structure in our dialogs is their cooperative nature. Since the dialogs are cooperative and both participants must be focused on the same subset of knowledge in order to understand one another [GROS77a], we expect IS to shift topic slowly between consecutive utterances, and to adhere to the focusing constraints espoused by McKeown [MCKE85a]. McKeown expanded on focus rules proposed by Sidner [SIDN81a] to explain how speakers should organize their utterances when faced with a choice of topic. In particular, McKeown claims that a speaker should move to a recently introduced topic if he has something further to say about it; otherwise he will have to reintroduce the topic at a later time. Similarly, the speaker should choose to finish discussion of the current topic before switching back to a previous one.

Our focusing heuristics rely on these expectations about possible shifts in focus of attention in IS's underlying task-related plan to identify which candidate focused plan is most relevant to the established dialog context and to determine how it fits into the context model. The following ordered list gives the focusing heuristics' preferences on the relationship between a candidate focused plan and the context model. Each relationship is illustrated under the assumption that the tree shown in Figure 2a is the context model immediately preceding the utterance, with node C (marked by an asterisk) representing the current focused goal/plan and node G representing the most recently considered subgoal in the current focused plan.

1. The candidate focused plan is part of the expansion of a plan for the most recently considered subgoal in the current focused plan; for example, if Figure 2b is an expansion of the context model shown in Figure 2a, then node C1 might represent such a candidate focused plan.
2. The candidate focused plan is part of an expansion of the current focused plan; for example, if Figure 2b is an expansion of the context model shown in Figure 2a, then node C2 might represent such a candidate focused plan (where C2 is part of a plan for the goal at node F, which is in turn part of a plan for the goal at node C).
3. The candidate focused plan is part of the expansion of a plan for a goal along the active path, with preference given to goals that are closest to the current focused goal on the active path. For example, if Figure 2b is an expansion of the context model shown in Figure 2a, then nodes C3 and C4 would both be part of the expansion of a plan for a goal along the active path; but if nodes C3 and C4 both represent candidate focused plans, then node C3 would be preferred, since it appears in an expansion of the plan associated with node B, which is closer to the current focused goal (represented by node C) than is node A.
4. The candidate focused plan is a plan whose expansion contains the goal associated with the root of the context model; Figure 2c illustrates such a relationship, where node C5 represents a candidate focused plan.
5. The candidate focused plan is part of the expansion of a higher-level plan, and this expansion also contains the goal associated with the root of the context model; Figure 2d illustrates such a relationship, where node C6 represents a candidate focused plan.

In applying each rule, we use a breadth-first expansion of plans, so that the resulting shift in focus of attention will be as small as possible. For example, if nodes C7 and C8 in Figure 2e both represented candidate focused goals/plans, the second rule in the above list would prefer C7 to C8, since C7 is closer to the existing focus of attention in the dialog.

3.3. An Example

To illustrate this plan inference process, let us consider a dialog segment containing four utterances by IS. Suppose IS begins with the statement:

IS: "I want to major in computer science."

Since IS states that he wants to achieve a goal, majoring in CS, TRACK's plan identification heuristics hypothesize

Satisfy-Major(IS, BA, CS)
and
Satisfy-Major(IS, BS, CS)

as candidate focused goals and their associated plans as candidate focused plans. Since there is no way of choosing between these, two context models would be built,

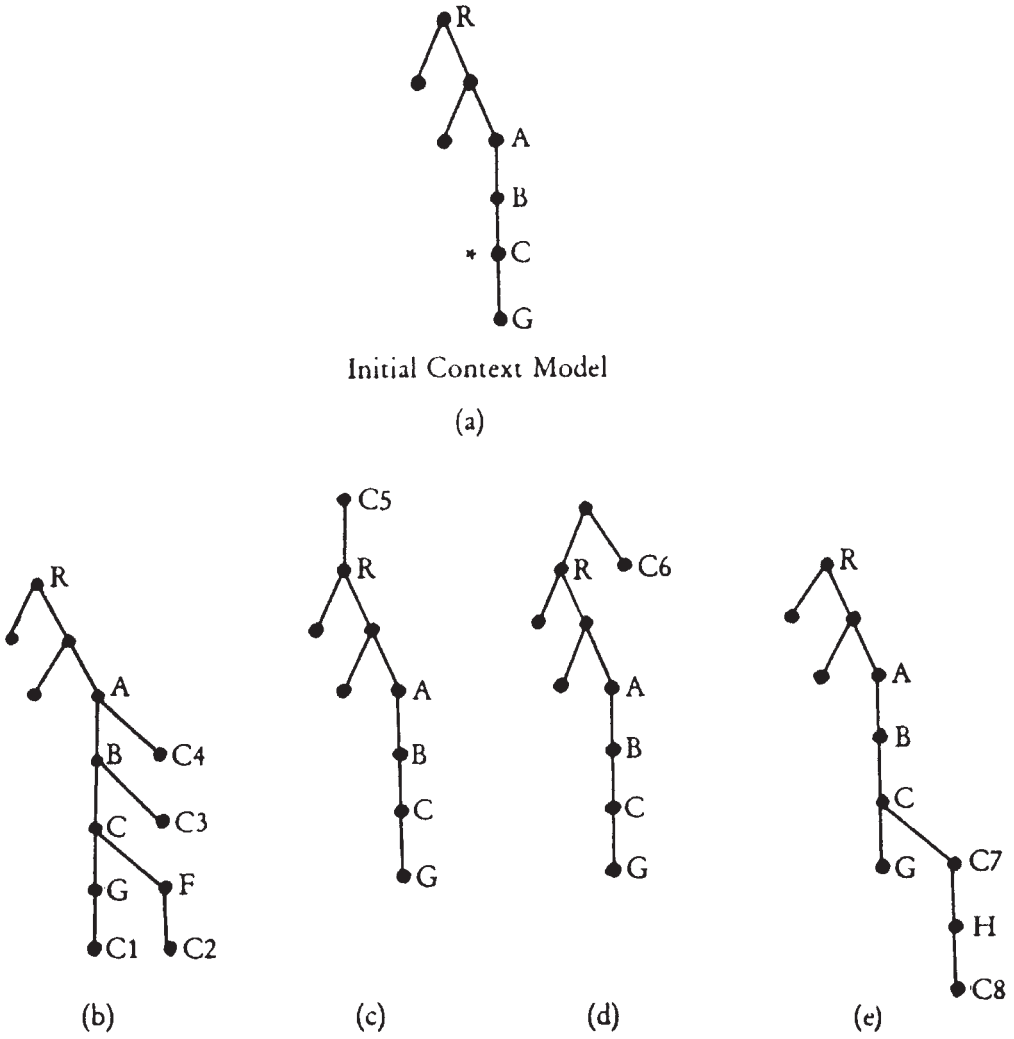


Figure 2. Examples of relationships between candidate plans and context model

each with one of the candidate focused goal/plan pairs as its root (Figure 3). The resulting current focused plan in each context model is preceded by an asterisk.

* Satisfy-Major(IS, BA, CS)	* Satisfy-Major(IS, BS, CS)

Figure 3. Two context models

Suppose that IS's next utterance is the query

"What are the prerequisites for taking CS180?"

Since IS wants to know the preconditions for the plan associated with the goal of taking CS180 (the introductory course for majors and minors in computer science), TRACK hypothesizes

Earn-Credit(IS, CS180, _ss:&SEMESTERS, _cr1:&CREDITS)

and its associated plan as the candidate focused goal/plan pair. The focusing heuristics must determine how this relates to the preceding dialog, as represented by the context model. The strongest expectation is that IS is continuing with some aspect of the current focused plan; since the preceding utterance did not address any particular goal in this plan, there is no *most recently considered* subgoal. Since taking CS180 appears in an expansion of the plans for majoring in computer science, TRACK expands the context models as shown in Figure 4 and marks the plan for earning credit in CS180 as the new current focus of attention.

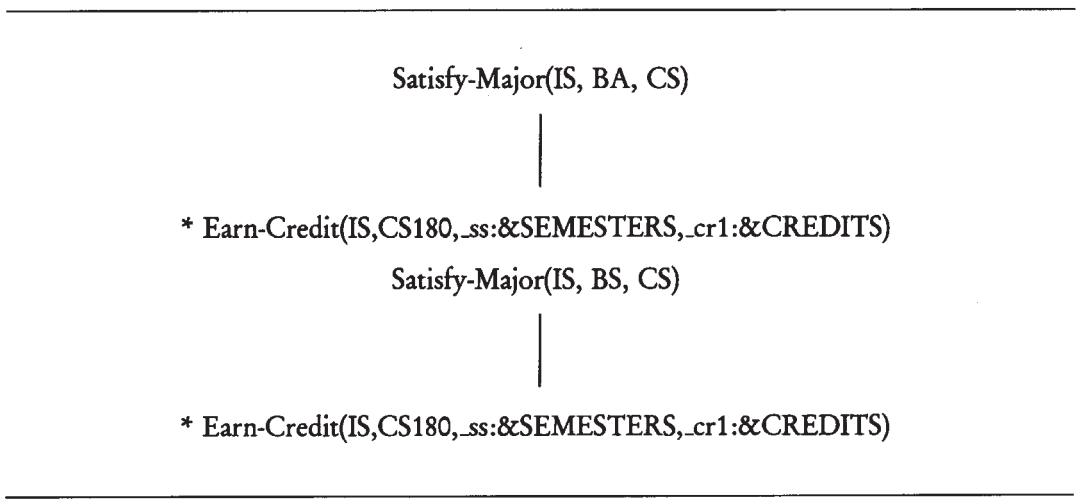


Figure 4. Context models after two utterances

Suppose that IS's next query is

IS: "What courses must I take in order to satisfy the foreign language requirement?"

Since IS is asking about the argument (courses) of a subgoal (taking courses) that is part of a plan for achieving a second goal, TRACK hypothesizes the second goal and its associated plan

Satisfy-Language-Req(IS)

as the candidate focused goal/plan pair. The focusing heuristics must now determine how this relates to the preceding dialog, as represented in the context model. The strongest expectation is that IS will continue with some aspect of the current focused plan. However, the candidate focused plan does not appear in an expansion

of the current focused plan, indicating that IS has shifted focus to another aspect of the overall task. In fact, none of the first four focusing heuristics find a relationship between the candidate focused plan and the context model. However, the last focusing heuristic finds that there is a goal,

Obtain-Degree(IS, BA)

whose associated plan can be expanded to include both the candidate focused plan and the context model whose root is Satisfy-Major(IS, BA, CS), indicating that IS has shifted his attention to another subtask (satisfying the foreign language requirement) of a higher-level plan (obtaining a Bachelor of Arts degree) of which the old current focused plan (obtaining a computer science major) is also a part. Therefore, this goal becomes the root of a new context model, as shown in Figure 5; Satisfy-Language-Req(IS) is marked as the new current focus of attention, as indicated by the asterisk preceding it. The other previous context model, whose root was Satisfy-Major(IS, BS, CS), is discarded, indicating that the second query has led us to deduce that IS wants to pursue a Bachelor of Arts degree. Note that our plan inference process makes what Pollack [POLL86a] terms the *appropriate query assumption* – namely, that IS does not ask queries that are inappropriate to his intended goal. This aspect of our plan inference process will be discussed further in Section 5, where we pursue a more robust plan recognition paradigm.

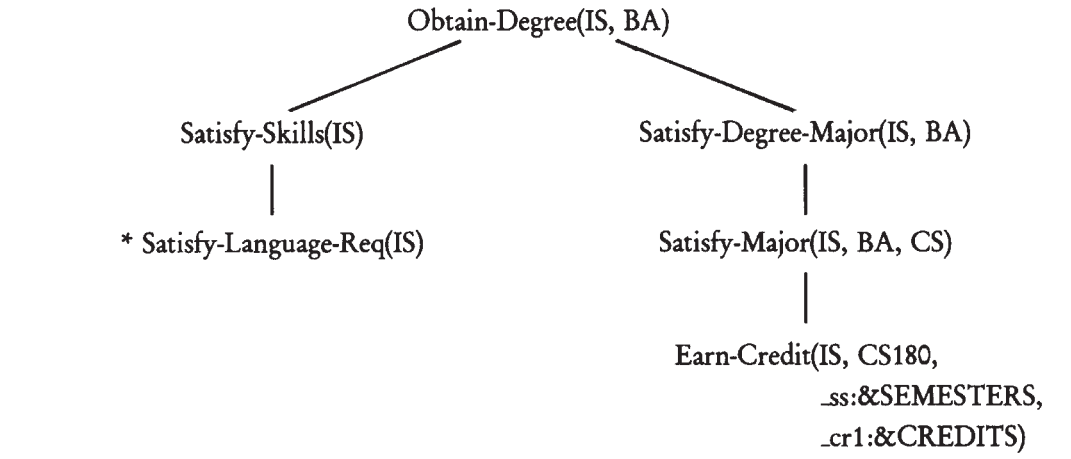


Figure 5. Context model after three utterances

Suppose that IS's next query is

IS: "Who is teaching Section 10 of French 112 in the fall?"

As described earlier, since IS is asking about the teacher of a particular section of a course, he may be considering the subgoal of learning from that teacher; this sub-

goal appears in a plan for learning the material of a course, and therefore TRACK hypothesizes the plan associated with the goal

Learn-Material(IS, FRENCH112-10-FALL, _syl:&SYLLABI)

as one of the candidate focused plans. The focusing heuristics find that this candidate focused plan appears in an expansion of the most recently considered subgoal (taking courses) in the current focused plan, and therefore it is selected as the new focus of attention and the context model is expanded to include it (Figure 6).

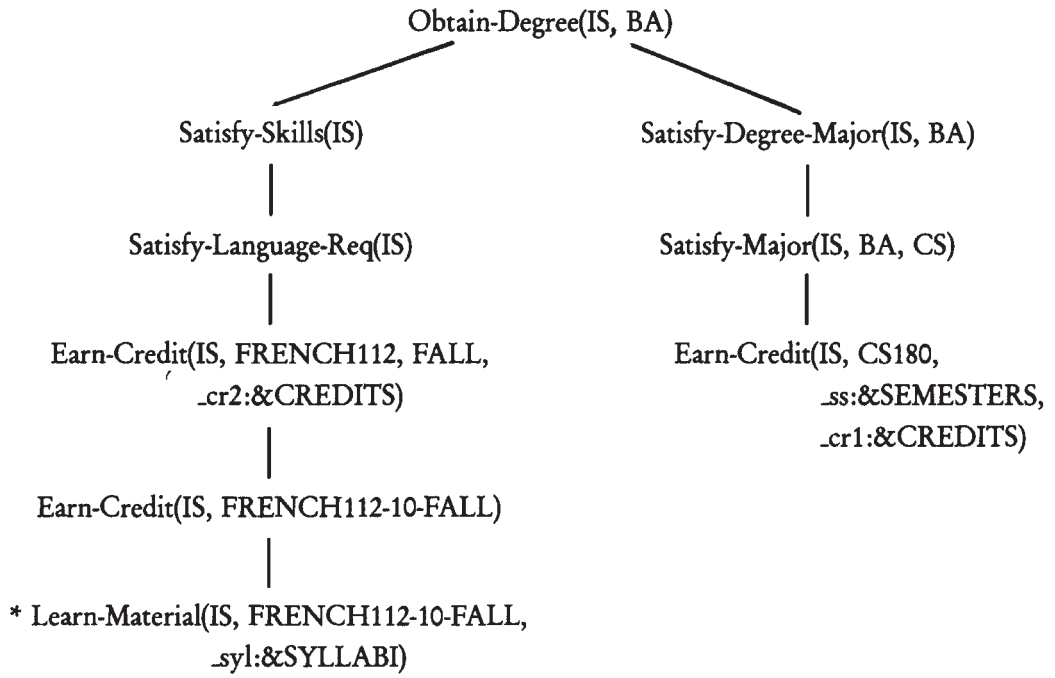


Figure 6. Context model after four utterances

In this manner, our plan inference process dynamically infers from an ongoing dialog the underlying task-related plan motivating an information-seeker's queries and tracks his focus of attention in this plan structure.

4. Application of User Models

The context model is one component of a comprehensive user model, representing the system's acquired beliefs about the plan an information-seeker is trying to construct. The possible expansions of this plan provide expectations about information that IS might want, and these expectations can often be used to repair and

disambiguate IS's subsequent utterances. We have developed strategies that use our context model to handle two forms of problematic input: pragmatically ill-formed utterances and intersentential ellipsis. This section describes our approach to the first of these; our framework for handling ellipsis is described in [CARB85].

4.1. Pragmatic Ill-Formedness

Every listener, whether human or machine, has a model of the world that is used to interpret utterances. An utterance can be syntactically and semantically well-formed, but violate the structural properties of the listener's world model. We shall say that such an utterance is *pragmatically ill-formed.*

Consider, for example, the query

IS: "What is the area of the Special Cargo Hold on the Europa?"

whose semantic representation will contain the proposition

Area(EUROPA-SPECIAL-CARGO-HOLD, _areaval:&SQ-FT)

A query similar to this appears in a dialog transcript of an information-seeker attempting to load cargo onto ships using the REL natural language interface [THOM80]. The system was unable to understand the query since the query erroneously presumed that these storage locations had an *area* attribute in the associated database. If a human information-provider had a similar problem in understanding the utterance, or considered the meaning of *area* ambiguous, he might be able to use the context established by the preceding dialog to identify what the information-seeker really wanted to know. For example, if IS's goal is to load cargo of the appropriate type into the various cargo holds, then he probably wants to know the *remaining capacity* of the Special Cargo Hold. On the other hand, if his goal is to assign ships to routes in order to handle the expected cargo shipping requirements, then IS probably wants to know the *total capacity* of the Special Cargo Hold; similarly, if his goal is to assign workers to fill the storage holds, with one worker assigned to handle all cargo holds located in the same section of the ship, then IS probably wants to know the *location* of the Special Cargo Hold.

Another example of a pragmatically ill-formed query illustrates the *missing joins* problem.

IS: "Who is teaching Section 10 of French 112 in the fall?"

IP: "Dr. Walker."

IS: "When's Mitchel meet?"

A semantic representation of this query contains the proposition

Meeting-Time(MITCHEL, _tme:&MEETING-TIMES)

Suppose that in the system's world model, faculty teach sections of courses, chair committees, and present colloquia, and each of these may have a scheduled meeting time, but there is no direct relationship between faculty and times. Then the above

query will appear pragmatically ill-formed. Although this utterance might be an abbreviated version of any of the queries

"When does the section of French 112 taught by Dr. Mitchel meet?"
"When does the committee chaired by Dr. Mitchel meet?"
"When does the colloquium given by Dr. Mitchel meet?"

a human information-provider would be likely to recognize from the above dialog that IS wants the meeting time of the section of French 112 taught by Dr. Mitchel, and respond accordingly.

4.2. Understanding Pragmatically Ill-Formed Queries

Our analysis of naturally occurring dialog suggests that human listeners often reason on their model of the speaker to remedy many of his faulty utterances and allow the dialog to continue without interruption. We contend that a natural language system should operate in a similar manner since, if its communication is to be regarded as natural, the system must be able to handle the full spectrum of utterances that humans understand with relative ease. A number of researchers have investigated the problem of handling pragmatically ill-formed queries [SOWA76, CHAN78, MAYS80b, KAPS79], but their strategies were deficient in that they considered the queries in isolation, without using a model of the preceding dialog to address the speaker's intentions.

Grice's theory of meaning [GRIC69, GRIC57] and maxim of relation [GRIC75] suggest that the listener's beliefs about what the speaker is trying to do should be used to recognize the intent behind an ill-formed query. According to Grice's theory, a listener should believe that the speaker thinks the listener can infer the intended meaning of an utterance – otherwise the speaker would not have made it. So given a pragmatically ill-formed query, a cooperative listener should attempt to deduce these intentions. Grice's maxim of relation suggests that the speaker's utterance is relevant to the existing dialog context, so the listener should use this context to attempt to deduce the speaker's intended meaning and enable the dialog to continue without interruption.

Our strategy is based on this theory of meaning and intention. It uses the context model to suggest substitutions for the erroneous proposition appearing in the semantic representation of IS's pragmatically ill-formed query, thereby producing semantic representations for one or more revised queries, all of which are relevant to what IS is trying to accomplish. If more than one revised query is proposed, then it must be determined whether any of these is significantly more likely than the others to represent the speaker's intentions or satisfy his perceived needs. Two criteria appear appropriate for comparing suggested revised queries. The first is the relevance of the revised query to the current focus of attention in the dialog. Since we have contended that some shifts in focus of attention in the plan structure are more likely than others, it is reasonable to hypothesize that the more expected the shift in focus of attention that would result from a revised query, the more likely is that query to represent the speaker's intentions. The second criteria for comparing

suggested revised queries is the similarity of a revised query to the speaker's actual utterance. The more the speaker's utterance is altered, the less likely it is to represent what he meant to say.

Therefore our pragmatic ill-formedness processor contains a suggestion mechanism and a selection mechanism. The suggestion mechanism proposes revised queries, all of which are relevant to IS's underlying task-related plan, and the selection mechanism uses the criteria of relevance and semantic similarity to select, from among multiple suggestions, the revised query deemed most likely to represent the speaker's intentions or satisfy his perceived needs.

4.3. Repair Strategy

Our suggestion mechanism uses two sets of substitution heuristics, one for making simple substitutions of a property, relation, function, or object class for that used by the speaker, and a second set for expanding relational paths to handle the missing joins problem. As an example of a simple substitution, suppose the dialog preceding the query

"What is the area of the Special Cargo Hold on the Europa?"

indicates that IS's current focused goal within his overall plan is to load cargo of the appropriate type into the various cargo holds. A subgoal in the plan associated with this goal would be

Load-Type-Cargo(IS, _item:&CARGO, _storearea:&STORAGE-AREA)
where
Is-Type(_item:&CARGO, _cartype:&CARGO-TYPE)
Cargo-Type(_storearea:&STORAGE-AREA, _cartype:&CARGO-TYPE)

and a plan for this subgoal would contain the precondition

Greater(_remcap:&CUBIC-FT, _itemsize:&CUBIC-FT)
where
Volume(_item:&CARGO, _itemsize:&CUBIC-FT)
Remaining-Capacity(_storearea:&STORAGE-AREA, _remcap:&CUBIC-FT)

specifying that the storage area must have room for the cargo item. Our property substitution heuristic would examine this plan and suggest substituting either of the propositions

Cargo-Type(EUROPA-SPECIAL-CARGO-HOLD, _cartype:&CARGO-TYPE)
and
Remaining-Capacity(EUROPA-SPECIAL-CARGO-HOLD,
_remcap:&CUBIC-FT)

for the erroneous proposition

Area(EUROPA-SPECIAL-CARGO-HOLD, _areaval:&SQ-FT)

appearing in the semantic representation of IS's query, producing suggested revised semantic representations equivalent to the two queries

IS: "What is the cargo type of the Special Cargo Hold on the Europa?"

IS: "What is the remaining capacity of the Special Cargo Hold on the Europa?"

More formally, this heuristic is represented by the following rule:

> If IS's proposition erroneously presumes that a member Obj of CLASS1 has a property Att1, then replace property Att1 with property Att2 if the following conditions hold:
>
> 1. a proposition specifying property Att2 on a member Obj2 of CLASS1 appears in an expansion of IS's context model.
> 2. Obj1 and Obj2 unify (Either Obj1 in the semantic representation of IS's utterance or Obj2 in the plan proposition refers to a general member of CLASS1, or both refer to the same specific member of CLASS1).

In the context of our student advisory dialog, suppose a student wants to pursue an independent study project; such projects can be directed by full-time faculty but not by faculty who are *extension* or *on sabbatical*. The student might erroneously follow the utterance

"I want to take an independent study project."

with the pragmatically ill-formed query

"What is the classification of Dr. Smith?"

In a university world model, only students have a classification attribute; this attribute can have values such as Arts&Science-1989, Engineering-1988, and Business-1990. Faculty have attributes such as rank, status, age, and salary. Pursuing an independent study project under the direction of Dr. Smith has the precondition that Dr. Smith's status be *full-time* or *part-time*. Our substitution mechanism would analyze the plan for taking an independent study course, and the property substitution rule would suggest substituting the proposition

Status(DR.SMITH, _statusval:&STATUSVALUES)

for the erroneous proposition

Classification(DR.SMITH, _classval:&CLASSVALUES)

appearing in the semantic representation of the student's query, resulting in a suggested revised semantic representation equivalent to the query

"What is the status of Dr. Smith?"

As an example of the second set of heuristics, the path expansion heuristics, consider again the query

"When's Mitchel meet?"

following the dialog that produced the context model shown in Figure 6. As mentioned earlier, the semantic representation of this query contains the erroneous proposition

Meeting-Time(MITCHEL, _tme:&MEETING-TIMES)

indicating a direct relationship between faculty and times. Our path expansion heuristics will analyze and expand the context model shown in Figure 6 and note that a plan for the goal

Earn-Credit(IS, FRENCH112, FALL, _cr2:&CREDITS)

can include a path containing the sequence of goals

Earn-Credit-Section(IS, _sec1:&SECTIONS)
where
Is-Section-Of(_sec1:&SECTIONS, FRENCH112)
Section-Offered(_sec1:&SECTIONS, FALL)

Learn-Material(IS, _sec1:&SECTIONS, _syl1:&SYLLABI)
where
Is-Syllabus-Of(_sec1:&SECTIONS, _syl1:&SYLLABI)

Learn-From-Person(IS,_sec1:&SECTIONS,_fac:&FACULTY)
where
Teaches(_fac:&FACULTY, _sec1:&SECTIONS)

Attend-Class(IS,_plc:&MEETINGPLCS,_day:&MEETINGDAYS,
_tme:&MEETINGTIMES)
where
Is-Meeting-Place(_sec1:&SECTIONS, _plc:&MEETINGPLCS)
Is-Meeting-Day(_sec1:&SECTIONS, _day:&MEETINGDAYS)
Is-Meeting-Time(_sec1:&SECTIONS, _tme:&MEETINGTIMES)

One path expansion heuristic notes that the propositions

Teaches(_fac:&FACULTY, _sec1:&SECTIONS)
Is-Meeting-Time(_sec1:&SECTIONS, _tme:&MEETINGTIMES)

both appear on this path in the expanded plan, and suggests substituting the conjunction of the propositions

Teaches(MITCHEL, _sec1:&SECTIONS)
Is-Meeting-Time(_sec1:&SECTIONS, _tme:&MEETINGTIMES)

for the erroneous proposition appearing in the semantic representation of IS's query, resulting in a suggested revised semantic representation equivalent to the English query

"When do sections taught by Mitchel meet?"

Although we have only illustrated substituting a conjunction of two propositions for the erroneous proposition in the user's query, the path expansion heuristics can propose expansions of any length.

Other heuristics and other parts of a plan will suggest other substitutions in addition to the ones shown in our examples. The important point is that the revised queries resulting from these suggestions are all relevant to the plan that IS is constructing. As mentioned earlier, relevance to the current focus of attention and similarity to the speaker's actual utterance are used to select from among multiple suggestions. We use focusing heuristics, similar to those used for constructing the context model, to measure relevance of a revised query to the current focus of attention in the dialog, and generalization hierarchies for properties, relations, functions, and object classes to measure the semantic similarity of a substituted term and the term that it replaces. In the example,

"What is the area of the Special Cargo Hold on the Europa?"

both suggested revised queries have approximately the same relevance to the current dialog but, of the two properties *cargo type* and *remaining capacity*, *remaining capacity* is much closer semantically to the property *area* used by the speaker. Therefore our selection mechanism chooses

"What is the remaining capacity of the Special Cargo Hold on the Europa?"

as the most appropriate query representing IS's needs.

This approach is superior to previous strategies because it uses a model of the speaker to identify and address his perceived intentions and needs in making an utterance. As such, it not only reasons on the user model to suggest possible interpretations relevant to the user's goals and plans, but it also limits consideration to those interpretations that are reasonable given the established dialog context.

5. Improving Plan Recognition

Our research has shown how an information-seeker's underlying task-related plan can be dynamically inferred from an ongoing dialog, and how the resulting context model can be used to achieve better communication. However, research on plan recognition has made four critical assumptions which enable the system to avoid errors in forming its beliefs about the user:

1. IS has no misconceptions about the task domain.[3]

[3] Allen's and Pollack's models are exceptions, as described later.

2. IS's queries always address aspects of the task within the system's limited knowledge. These systems maintain the *closed world assumption* [REIT78].
3. IS's statements and queries are correct and not misleading.
4. The system's inference mechanisms do not introduce errors into the context model.

These assumptions represent unrealistic constraints on real-world dialogs and must be removed. The first assumption, called the *valid plan assumption* by Pollack [POLL87], limits the kinds of beliefs IS can already have about the domain – namely, it says that IS's knowledge may be incomplete but not erroneous. But IS is interacting with the system because IS does not know enough about the domain to construct his task-related plan by himself. Therefore, since IS is not an expert in the area, it is to be expected that some of his beliefs about the domain may be false, contradicting the first assumption. An implication of the valid plan assumption is what Pollack terms the *appropriate query assumption* – namely, that IS knows enough about how to solve his problem that he always asks relevant questions.

The second assumption limits the questions IS can ask to those which the system can answer. But even an expert system has limited domain knowledge. Furthermore, in a rapidly changing world, knowledgeable users may have more accurate information about some aspects of the domain than does the system. For example, a student advisory system may not be altered immediately upon changing the teacher of a course. A cooperative system should recognize its limited knowledge and reason with it to provide whatever pertinent, helpful information it can.

The third assumption restricts IS to utterances that are clear, precise, and accurate. For example, it eliminates the possibility that IS might say he is a junior, when in fact he is 3 credits short of junior standing, thereby leading the system to erroneously infer that IS is eligible for certain programs or awards. But human information-seekers are often imprecise, especially when they are not aware that small perturbations in the data can be significant.

The fourth assumption says that the system never makes an error in inferring IS's plan. But even in the simplest cases, the system must hypothesize how individual utterances relate to one another. Such decisions select from among multiple possibilities and are a potential source of error.

Pollack [POLL87] argues against plan inference systems making the first two assumptions, because they prevent the system from inferring plans which the user believes he can pursue but which are novel (to the system) or invalid. However, there is another implication of relaxing the appropriate query assumption that is not considered by Pollack: IS may ask an irrelevant question that seems perfectly reasonable to the system, thereby leading the system to develop incorrect beliefs about IS's objectives. Consider, for example, a student advisory system. If only BA degrees have a foreign language requirement, the query

"What courses must I take to satisfy the foreign language requirement in French?"

may lead the system to infer that IS is pursuing a Bachelor of Arts degree. If only BS degrees require a senior project, then a subsequent query such as

"How many credits of senior project are required?"

is problematic. Either the second query is inappropriate to IS's goal of obtaining a Bachelor of Arts degree [POLL87], or the system's context model does not accurately reflect what IS wants to do. Note that in both cases the user has a misconception; but in the latter case, the misconception went undetected and was allowed to introduce errors into the system's context model.

Traditional natural language plan inference systems also make the third and fourth assumptions, which together with the first two, guarantee that the underlying plan inferred by the system and the task-related plan under construction by IS are never at variance with one another. If we want systems capable of understanding and appropriately responding to naturally occurring dialog, natural language interfaces must be able to deal with situations where those assumptions are not true.

Grosz [GROS81] claimed that miscommunication can occur if both dialog participants are not focused on the same subset of knowledge. Joshi [JOSH82] contended that successful communication requires that the mutual beliefs of the dialog participants be consistent. Extending this to inferred plans, we claim that a successful cooperative dialog requires that the system's beliefs about IS's plan be consistent with what IS is actually considering doing. But clearly it is unrealistic to expect that the system's model will always be correct, given the different knowledge bases of the two participants and the imperfections of communication via dialog.

Thus we need a repair mechanism that attempts to detect inconsistencies in the models and repair them whenever possible. This view is supported by the work of Pollack, Hirschberg and Webber [POLL82b]. They suggested that expert-novice dialogs could be viewed as a negotiation process, during which not only an acceptable solution is negotiated but also understanding of the terminology and the beliefs of the participants. The context model is one component of the system's beliefs, as is its belief that this model accurately reflects the plan under construction by IS.

In the remainder of this chapter, we outline a four-phase approach to the problem of handling plan disparity. This approach was motivated by an analysis of naturally occurring dialog and is an extension of the hypothesize and revise paradigm proposed by Schmidt, Sridharan, and Goodson [SCHM78] for inferring a user's goal by observing his non-communicative actions. This approach requires a richer context model than maintained by current systems, and we argue that the support that the system accords a component as a correct part of the model should affect its hypothesis about the source of error once plan disparity is suggested.

5.1. Related Work

Several research efforts have addressed problems related to plan disparity. Kaplan [KAPS79], McCoy [MCCO*], and Quilici [QUIL*] investigated misconceptions about domain knowledge and proposed responses intended to remove the misconceptions. However, such misconceptions may not be exhibited when they first influence the information-seeker's plan construction; in such cases, disparate plans may result, and correction will entail both a response correcting the misconception and further processing to bring the system's context model and the plan under construction by the information-seeker back into alignment.

Allen's [ALLE80] plan inference system did not eliminate invalid plans, but weighted them less favorably than valid plans. However, his model did not consider how potential user misconceptions might affect the partial plan inferred by the system.

Pollack [POLL86a] studied removal of the *appropriate query assumption* of previous planning systems. She proposed a richer model of planning that regarded plans as mental phenomena and explicitly reasoned about the information-seeker's possible beliefs and intentions. She addressed the problem of queries that indicated the information-seeker's plan was inappropriate to his overall goal, and attempted to isolate the erroneous beliefs that led to the inappropriate query. However, queries deemed inappropriate by the system may signal phenomena other than that the query is inappropriate to what the user really wants to do. For example, the information-seeker may have shifted focus to another aspect of the overall task without successfully conveying this to the system, the system's context model may have been in error prior to the query, or, as noted by Pollack [POLL87], the information-seeker may be addressing aspects of the task outside the system's limited knowledge.

Pollack was concerned with issues that arise when the information-seeker's plan is incorrect due to a misconception exhibited by the current query. She assumed that, immediately prior to the user making the problematic query, the system's partial model of the user's plan was correct. We argue that since the system's inference mechanisms are not infallible and communication itself is imperfect, the system must contend with the possibility that its inferred model does not accurately reflect the user's plan. Previous research has failed to address this problem.

Schmidt, Sridharan, and Goodson [SCHM78] proposed a hypothesize-and-revise paradigm for inferring a user's goal by observing his non-communicative actions. They formulated a set of revision critics for altering a plan upon observing actions that conflict with expectations, but failed to provide any principles or mechanism for selecting the appropriate revision. Although the model of plan recognition for an office environment formulated by Carver, Lesser, and McCue [CARV84] attempted to repair the inferred plan when actions inconsistent with it were observed, it did not reason about where the plan might be wrong but merely backtracked to select another interpretation.

5.2. An Approach to Robust Plan Recognition

Our goal is to have the system's context model reflect the plan under construction by the information-seeker. But errors may enter the system's context model if the user's statements are inaccurate or misleading, if the system uses default inference rules, or if the system uses focusing heuristics to relate new utterances to the existing dialog context. In addition, models that are less specific than intended by the information-seeker may result if the user fails to adequately communicate his intentions (or the system fails to recognize these intentions). Furthermore, the system's limited domain knowledge may hamper its recognition and modeling of a valid plan being constructed by the user. As mentioned earlier, Pollack [POLL86a] has addressed the problem of recognizing and reasoning about invalid plans; our

concern here is with modeling the user's plan when his planning is correct, but for some reason the system's model fails to reflect that plan.

Rarely will an information-seeker be aware that the system has constructed an incorrect model of what he wants to accomplish, and even less often will he know exactly where the model is wrong. Instead, the information-seeker may express surprise with the system's response or in some way convey that his needs are not being satisfied. The system must then determine the cause of this dissatisfaction.

Our research indicates that detecting and recovering from disparate plan structures requires a four-phase approach:

1. Detecting clues to disparity between the system's model of the user and the user's actual goals and plans.
2. Reasoning on the system's model and the system's domain knowledge to hypothesize the source of these disparities.
3. Responding to the user and negotiating with him to isolate the errors.
4. Appropriately repairing the user model.

This approach is motivated by the way humans appear to respond in similar situations, as determined through an analysis of dialog transcripts. Therefore a natural language system that pursues this strategy will be viewed as acting naturally by human users.

The next sections discuss each of these phases. The dialogs shown in Figures 7 and 8, which we will refer to as the *IRA example* and the *Kermit example* respectively, will be used to motivate and illustrate the strategy.

---

(1) **IS:** "I'm a retired government employee but I'm still working. I'd like to start out an IRA for myself and my wife — she doesn't work."

(2) **IP:** "Did you work outside of the government last year?"

(3) **IS:** "Yes I did."

(4) **IP:** "There's no reason why you shouldn't have an IRA for last year."

(5) **IS:** "I thought they just started this year."

(6) **IP:** "Oh no. IRA's were available as long as you are not a participant in an existing pension."

(7) **IS:** "Well, I do work for a company that has a pension."

(8) **IP:** "Ahh. Then you're not eligible for 81."

Figure 7. Individual retirement account (IRA) dialog

(1) **IS:** "I wish I could transfer files between the Vax and my PC."

(2) **IP:** "Kermit lets you do that."

(3) **IS:** "How do I get Kermit?"

(4) **IP:** "The computing center will give you a copy if you bring them a floppy disk."

(5) **IS:** "How late is the University Bookstore open?"

Figure 8. File Transfer Via Kermit Dialog

The first dialog[4] was transcribed from a radio talk show on investments[5], and the second is similar to a dialog that took place between two people in a university. Utterances are numbered for later reference.

5.2.1. Detection

As claimed earlier, since the system is presumed to be a cooperative dialog participant, it must be on the lookout for plan disparity. We have identified four sources of clues to the existence of such disparity:

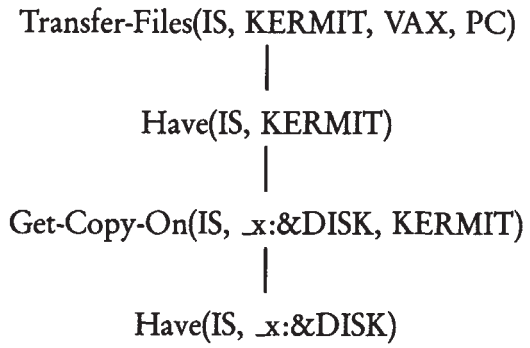
1. The discourse goals of IS, such as expressing surprise or confusion.
2. Conflicts between inferences drawn from IS's current utterance and the existing context model.
3. Relevance of IS's current utterance to the context model.
4. Abrupt shift in focus of attention in the context model.

IS can express surprise or confusion about the system's response, thereby cuing the possibility of plan disparity. For example, in the IRA dialog, plan disparity is suggested when IS, in utterance (5), expresses confusion at IP's previous response. On the other hand, IS's query may contradict or appear irrelevant to what the system believes is IS's overall task, leading the system to suspect that its context model may not reflect IS's plan. Or IS's query may require so sharp an unsignaled shift in focus as to suggest possible problems in the system's model of what IS wants to accomplish; the strongest expectations are for speakers to address aspects of the task closely related to the current focus of attention [SIDN81a, MCKE85a, CARB83].

The Kermit dialog illustrates a case in which plan disparity is suggested by an abrupt shift in focus of attention. Upon completion of utterance (4), the system's model of IS's plan might be represented as

[4] Minor alterations have been made to the dialog to remove restarts and extraneous phrasing.

[5] These transcripts were provided by the Department of Computer and Information Science of the University of Pennsylvania.



Suppose that the system does not know how to obtain floppy disks. Then from its limited knowledge, IS's next query,

"How late is the University Bookstore open?"

will not appear to address an aspect of the plan inferred for IS, or any expansion of it. The system could just respond by

1. answering the direct question, if possible, ignoring its ramifications.
2. responding *I don't know*, if the direct answer is not available.

However, cooperative information-providers are expected to try to understand the import of a query and provide as helpful a response as they can.

Grice's maxim of relation [GRIC75] suggests that IS believes the query is relevant to the overall dialog. Several possibilities exist. IS may be shifting focus to some aspect of a higher-level task that includes transferring files as a subaction. One such higher-level task might be to compose a document using the Scribe text formatting system, and the aspect queried by the new utterance might be the purchase of a Scribe manual from the university bookstore. In this case, the subtask of the overall task represented by the existing context model might be the transfer of files containing the document so that they can be modified using a PC editor.

On the other hand, focusing heuristics and the absence of discourse markers [SIDN83b] suggests that the new query is most likely to be relevant to the current focus of attention. So the system should begin trying to determine how IS's utterance might relate to the currently focused subtask in the context model, and consider the possibility that the context model might contain errors or that IS's domain knowledge might exceed the system's or be erroneous.

5.2.2. Hypothesis Formation

We have found that the knowledge acquired from the dialog and how it was used to construct the context model are important factors in hypothesizing the cause of disparity between the system's context model and the actual plan under construction by the information-seeker. Natural language systems must employ various techniques such as focusing heuristics and default rules for understanding and relating dialog in order to do the kind of inferencing exhibited in dialog transcripts and provide the most helpful responses. But confidence in individual components of the resultant context model appears to be important in hypothesizing errors. We contend that the system's context model should be enriched, so that its representation of the plan

inferred for the user differentiates among its components according to the support that the system accords each component as a correct and intended part of that plan. The system can then reason on this enriched context model to hypothesize the most likely sources of suspected disparities.

We currently see four kinds of support for plan components:

1. Whether the system believes that IS intended to communicate the component.
2. Whether the system has inferred the component on the basis of its own domain knowledge, which the system cannot be certain IS is aware of.
3. The kinds of rules used to add each component to the model (for example, default rules that select one component from among several possibilities, or heuristics that suggest a shift in focus of attention), and the evidence for applying the rules.
4. Whether the system's domain knowledge supports, contradicts, or is neutral regarding inclusion of the component as part of a correct overall plan.

The first three are important factors in formulating a hypothesis regarding the source of disparity between the system's model and IS's plan. If the system believes that IS intends the system to recognize from his utterance that G is a component of his plan, then the system can add G to its context model and have the greatest faith that it really is a component of IS's plan. Therefore such components are unlikely sources of disparity between the system's context model and IS's plan.

Components that the system adds to the context model on the basis of its domain knowledge will be strongly believed by the system to be part of IS's plan, but not as much as if the system believes IS intended to communicate them. These components resemble *keyhole recognition* rather than *intended recognition* [COHE81]. Keyhole recognition is the inference of an agent's goals and plans by unobtrusively observing him, as if through a keyhole; in such cases, the agent has no intent to communicate his goals to the observer. Intended recognition is the inference of those goals and plans that an agent *intends* to convey. Intended recognition is essential in communicative situations [COHE81], since the listener must identify the intended meaning of a speaker's utterance; but it appears that in user modeling and response generation, keyhole recognition should also play a significant role. Our analysis of naturally occurring dialog suggests that humans often use keyhole recognition in order to expand their beliefs about what the information-seeker is trying to do and how he should go about doing it, beyond what the information-seeker explicitly intended to convey. However, since the information-seeker may not have intended to communicate these goals and plans, they are more likely sources of error than components which IS intended that the information-provider recognize.

Consider, for example, a student advisement system. If CS180 is an introductory course restricted to majors in computer science and electrical engineering, then the system might infer from the utterance

"Can you tell me what time CS180 meets?"

not only that IS wants to know the meeting time for CS180, but also that IS is a computer science or electrical engineering major. However, IS may *intend* the system to recognize the first goal, but it is questionable whether he actually *intends* the system to recognize that he is pursuing a major in computer science or electrical

engineering. This latter inference is based on the system's beliefs about who can take CS180 – knowledge that IS may not have. Therefore, since IS may not have intended to communicate these components, they are more likely sources of error than components which he intended the system to recognize.

The particular rules used to add a component to the context model should affect the system's faith in that component as part of IS's overall plan. Consider again the IRA example in Figure 7. Utterance (4) can be explained by postulating that IP has applied the default inferences that since IS is retired but still working, he is only working part-time, is therefore not covered by a pension program, and is therefore eligible for an IRA during the year in question (at that time, rules on IRA's were different). IS's next utterance expresses confusion at IP's response, thereby cuing the possibility of plan disparity. In utterance (6), IP identifies one of these components as part of IP's context model, and attempts to justify its inclusion.

Analysis of naturally occurring dialogs such as that in Figure 7 suggest that natural language systems should use mechanisms such as default inference rules and focusing heuristics to expand the context model and provide a more detailed and tractable arena in which to understand and respond to subsequent utterances. However, these rules select from among multiple possibilities and therefore produce components that are more likely sources of error than components added as a result of intended recognition or inferences made on the basis of the system's domain knowledge.

Whether the system's domain knowledge supports, contradicts, or is neutral regarding inclusion of a component into the context model is important in deciding whether the user's query might address an aspect of the task outside the system's limited domain knowledge. Consider again the Kermit dialog shown in Figure 8. If the system can creatively construct a plan that may reasonably be ascribed to IS, is relevant to the current focus of attention, and which the system's knowledge supports or views neutrally (for example, floppy disks are an educational item and the bookstore sells many other items of an educational nature, so perhaps one can obtain a floppy disk by purchasing it from the bookstore), then the system can hypothesize that the cause of the plan disparity may be that IS has more extensive domain knowledge.

5.2.3. Response Generation and Negotiation

Webber [WEBB86] distinguishes between answers and responses. She defines an answer as the production of the information or execution of the action requested by the speaker, but a response as

"the respondent's complete informative and performative reaction to the question which can include ... additional information provided or actions performed that are salient to this substitute for an answer."

Our analysis of naturally occurring dialog indicates that humans *respond*, rather than *answer*, once disparate models are detected. This third phase often entails additional actions, including a negotiation dialog to ascertain the cause of the discrepancy and enable the models to be *squared away* [JOSH82] so that they are once again in alignment. Otherwise, the participants might lack confidence that they understand the other's utterances or that their own responses are being understood. In

the ensuing negotiation dialog, each participant may attempt to identify errors by making relevant assertions, asking questions, or justifying suspect portions of his own model.

The IRA example illustrates a negotiation dialog. In utterance (6), IP identifies a suspect component of her context model (that IRA's were available to IS in 1981) and justifies it. IS's next utterance informs IP that the assumption that led IP to infer that IS was eligible for an IRA is incorrect. IP then notifies IS that IP recognizes the error and that her context model has been repaired. The information-seeking dialog then resumes.

Negotiation is also appropriate when a query addresses an aspect of the task outside the system's limited domain knowledge. The system should attempt to understand IS's utterance in terms of the system's limited knowledge and provide any pertinent helpful information supporting or contradicting the user's plan. However, the system must inform IS of its limited knowledge in order to avoid misleading him by appearing to implicitly support his task-related plan. Thus the system's response takes the form of negotiation with the user as to whether the plan being constructed by the user is likely to work.

Consider again the Kermit dialog presented in Figure 8, and suppose that the system does not know how to obtain floppy disks but has reasoned that IS may be thinking about trying to purchase one from the university bookstore. The system can now respond to IS. This reply should of course contain a direct answer to IS's posited question. But this alone is insufficient. In a cooperative information-seeking dialog, IS expects the information-provider to assimilate the dialog and relate utterances to IS's inferred underlying task in order to provide the most helpful information. If the system limits itself to a direct response, IS may infer that the system has related IS's current utterance to his task-related plan and that the system's knowledge supports it – that is, that the system also believes IS can purchase a floppy disk at the bookstore. Joshi's revised maxim of quality [JOSH82] asserts that the system's response must block false inferences. In addition, as a helpful participant, the system should include whatever evidence it has for or against the plan component proposed by IS. An appropriate response would be:

> "The University Bookstore is open until 4:30PM. But I don't know whether it sells floppy disks. However, it does sell many other items of an educational nature, so it is perhaps a good place to try."

5.2.4. Model Repair

Once the cause of model disparity is identified, both participants must adjust their models to remove the disparities. If the system's model is in error, the system must alter its context model, as determined through the negotiation dialog. The system may communicate to IS the changes that are being made, so that IS can assure himself that the models now agree. For example, utterance (8) in the IRA dialog conveys that IP has modified her beliefs to include the correction provided by IS in utterance (7) and that, as a result, IP deduces that IS is not eligible for an IRA in 1981. On the other hand, if IS's model is in error, the system may inform IS of

any information necessary for IS to construct an appropriate plan and achieve his goals.

However, if the disparity was caused by IS's queries addressing an aspect of the task beyond the system's limited domain knowledge, then the system's model should be expanded to reflect what it believes to be IS's plan structure, noting however that its own knowledge offers only limited support for these components. This is the fourth kind of support that an enriched context model should include: differentiation among components according to whether the system's domain knowledge supports, contradicts, or is neutral regarding inclusion of the component as part of a correct overall plan. For example, upon determining that IS's domain knowledge may exceed the system's in the Kermit dialog, the system should expand its existing model to incorporate the acquired knowledge about how IS believes floppy disks can be obtained. The plan components creatively constructed can be added to the system's model, but as components proposed by IS and not fully supported by the system's knowledge. In this manner, the system can assimilate new utterances that exceed its limited domain knowledge and develop an expanded context model which serves as *knowledge* that can be referred back to in the ensuing dialog. This will enable the system to understand (within its limited knowledge) how IS plans to accomplish his objectives, and permits it to use this knowledge to understand subsequent utterances and provide helpful information.

5.3. Future Directions

We have outlined a four-phase approach to handling disparate plan structures, motivated by an analysis of naturally occurring dialog. We have further argued for an enriched context model that differentiates among its components according to the support it accords each component as a correct and intended part of the user's plan. Our current research is focusing on representing the knowledge that should affect the system's support for the components of an inferred plan, and on formulating heuristics for hypothesizing the source of error in the system's context model. This work should produce a more robust and natural user modeling framework.

6. Constructing and Exploiting Other Components of a User Model

There are many aspects of user modeling besides plan recognition that can aid in understanding and generating dialog. Lehman and Carbonell [LEHM*] have designed a framework that enables a natural language system to dynamically adjust to a user's idiosyncratic use of language. Their strategy extends the system's grammar to produce a customized parser that can be viewed as a model of how the user speaks.

Several researchers have exploited models of the user's knowledge and expertise. Chin designed a user modeling component called KNOME [CHIN*] that constructed a model of a user's knowledge about, and expertise in using, the UNIX

operating system. KNOME attempts to stereotype the user based on the user's queries to UC, a natural language help system for UNIX. This acquired knowledge about the user's expertise is then used in a variety of ways, including detecting user misconceptions and generating useful responses.

Paris was also concerned with exploiting a model of the user's domain knowledge. Her research indicated that texts directed toward naive users typically provide a different kind of information than those directed toward expert users. Her system, TAILOR [PARI*], used its beliefs about the user's domain knowledge to generate appropriate descriptions of physical objects.

Both McCoy and Quilici have addressed the problem of responding to misconceptions that a user exhibits about the domain. McCoy [MCCO*] was concerned with misconceptions about objects and their properties, and how the user's perspective on the domain should influence the corrective response. Quilici [QUIL*] was concerned with identifying the source of user difficulty in using the UNIX operating system and generating responses that corrected relevant user misconceptions.

Cohen and Jones [COHE*] have formulated a framework for reasoning on a user model in a system for educational diagnosis, in order to generate responses appropriate to the background and goals of individual users. Morik [MORI85b] has employed a user model to address user preferences in the context of a real estate system.

These research efforts can benefit from the work described in this chapter, and similarly our work can make use of many of the results of research in other areas of user modeling. For example, knowledge acquired from dynamically modeling the user's plans and goals may be useful in understanding ill-formed utterances [RAMS87] and could be incorporated into an adaptive parser. McCoy assumes that the system is given the user's perspective on the domain, and plan recognition can provide much of that perspective. On the other hand, if the system determines that a pragmatically ill-formed query is caused by a serious user misconception, the strategies proposed by McCoy and Quilici can be used to generate an appropriate corrective response. In addition, the system's beliefs about the user's expertise and knowledge of the domain will be important in hypothesizing whether a pragmatically ill-formed query is caused by a misconception or is simply the result of careless use of language.

7. Conclusions and Current Research

A cooperative natural language system must attempt to infer the underlying task-related plan motivating the information-seeker's queries and use this plan to provide cooperative, helpful responses. The system's model of this plan, which we call a *context model*, is one component of a user model. We have presented a strategy for dynamically inferring the context model from an ongoing dialog, and have shown how this model can be used to handle one class of problematic utterances – those that violate the pragmatic rules of the underlying world model.

However, the assumptions underlying current plan inference systems are unrealistic and must be removed. We have argued that an appropriate approach to handling

the resultant problem of disparity between the system's context model and an actual valid plan under construction by the information-seeker is an extension of the hypothesize and revise paradigm, consists of four phases, and requires an enriched context model that differentiates among its components according to the support it accords each component as a correct and intended part of the information-seeker's plan. Since this approach is motivated by the way humans appear to respond in similar situations, a natural language system that pursues this strategy will be viewed as acting naturally by human users.

8. References

[ALLE80], [CARB83], [CARB85], [CARB88], [CARV84], [CHAN78], [CHIN*], [COHE81], [COHE*], [FIKE71], [GRIC57], [GRIC69], [GRIC75], [GROS77a], [GROS81], [JOSH82], [KAPS79], [LEHM*], [LITM86], [MAYS80b], [MCCO*], [MCKE85a], [MORI85b], [PARI*], [PERR80], [POLL86a], [POLL87], [POLL82b], [QUIL*], [RAMS87], [REIT78], [ROBI80], [ROBI81], [SCHM78], [SIDN81a], [SIDN83b], [SIDN85], [SOWA76], [THOM80], [WEBB86]

7 Learning the User's Language: A Step Towards Automated Creation of User Models

Jill Fain Lehman and Jaime G. Carbonell

Abstract

No matter how comprehensive the coverage of a natural language interface, users will invariably provide unparsable input: idiosyncratic phrases, linguistically deviant utterances, or sentences simply beyond the linguistic sophistication of the interface. Although individual users differ significantly from each other in their preferred linguistic expression, they show surprising consistency in multiple interactions over time. Addressing both observations, a general framework is developed for automated adaptation to the preferred and idiosyncratic linguistic behavior of individual users. In essence, a method for flexible recovery and completion of partial parses is developed, with subsequent individualized extensions to a base grammar. Thus, the system adapts to the user by growing its grammar dynamically to acquire her preferred forms of expression and recognize them directly in future interactions.

1. Introduction: Adaptable Interfaces

To be effective, user interfaces require some measure of user training, whether by explicit instruction or implicitly through a gradual familiarization process. Thus, developing interfaces for novice users provides a great challenge in that the design must minimize the need for training and familiarization time. Natural language interfaces traditionally strive to bridge the user-system gulf by exploiting the user's existing familiarity with his or her own natural language.[1] However, even natural language interfaces fall short of immediate utility by naive users; no existing interface can deal with all the complexities of full natural language, including the full range of potentially relevant vocabulary, all non-standard turns of phrase, extra-grammatical utterances [CARN83d], abbreviations and other elided expressions. Thus, a user must learn the extent and limitations of the natural language interface, mostly by trial and error.

---

[1] Natural language exhibits many advantages (e.g., expressivity, human comprehensibility) and disadvantages (e.g., verbosity, complexity of machine interpretation) over other interface modalities such as menu selection, formal command interfaces, and graphic-oriented techniques. Whereas detailed comparative analyses are beyond the scope of this chapter, we note that the user modeling and adaptation techniques proposed here may well be adaptable to multi-media interfaces, speech, and any other free-form communication medium.

There are three possible methods of compensating for the fact that present interfaces do not achieve close to full coverage of the English language or the domain of discourse:

- **The user beware** – The simplest approach is to face the fact that many utterances will not be interpreted and expect the user to tolerate such a mismatch and learn the sublanguage of the system. One might hope that such adaptation could be rapid, but the empirical evidence discussed later in this chapter and elsewhere [CARN83a] suggests otherwise; frustration and disenchantment are the observed reactions.
- **The ultimate language interpreter** – Whereas the long-term goal of natural language processing is to interpret the full range of syntactic and semantic constructions possible in natural language while accommodating a measure of grammatical deviance [CARN83d, FAIN85], the state of the art does not approach such total performance. Moreover, as lexicons, grammars and semantic knowledge bases increase in size, and as more grammatically deviant constructions are searched upon failing an exact parse, the response time of any comprehensive system must increase considerably. Near real-time response, however, is necessary in a habitable interface. Also problematic is the reality that with increasing grammar size and increasing ability to recognize deviant forms, the number of spurious interpretations that the user would never consider sensible rises dramatically.
- **The user-adaptable interface** – Instead of asking the user to adapt, or asking the developer to build the ultimate system with complete coverage, a radically different approach is to build a system capable of adapting to the linguistic usage patterns of the user. In this approach we start with a kernel grammar (including semantics and lexicon) that is extended dynamically when the user provides uninterpretable input. Such extensions occur both by a combination of selective relaxation and augmentation of the kernel grammar and by focused clarificational dialog. Extensions to the grammar are stored in an individualized user model[2]; progressively fewer extensions are required as the system's linguistic coverage extends to that of the individual user (rather than to the union of all possible users, as in the previous alternative). This alternative is clearly superior for users who interact frequently enough with a system for it to create a rich linguistic user model.

The remainder of this chapter discusses in some detail the adaptation model in which a kernel grammar is gradually extended to match the linguistic usage patterns of each individual user. In addition, empirical evidence from *hidden-operator* experiments is presented that suggests:

- User performance, measured in speed and accuracy of task completion, increases significantly as the system adapts to the linguistic usage patterns of individual

[2] Thus, the grammar serves as a user model in a sense distinct from that of Schuster [SCHU85]. Her system, VP^2, used a static grammar in each of a native and second language to help detect and explain errors in second language use.

users. The same performance advantages accrue in a non-adaptive version of the interface only when the user's natural linguistic style conforms closely to the chosen kernel, and thus little adaptation by the user is required.

- The size of each resulting extended and user-customized grammar is much smaller than an equivalent complete grammar, i.e. a grammar that could tolerate all expected grammatically deviant forms. Moreover, the user customizations do not overlap greatly, showing diversity of usage across users, but consistency within users. This is exactly the empirical profile that makes our user customization process a tractable solution to the demands of the task.

2. Foundations: Least-Deviant-First Parsing and MULTIPAR

Our approach to adaptive parsing is based in part on ideas first explored in the flexible parser MULTIPAR [FAIN85] [MINT85]. MULTIPAR was designed to allow for the particular types of extragrammatical behavior frequently displayed by natural language interface users, for example, simple spelling errors, some word-order inversions, missing case markers, missing determiners, and semantic inconsistencies.

Like most natural language parsers, the system's performance can be described as search through the virtual space of possible parses for a given input. However, since ungrammatical constructions were permitted, the system could not limit search by rejecting a partial parse when a syntactic or semantic constraint was violated. Instead, each type of *deviation* from grammaticality had an associated recovery strategy that compensated for it (e.g. by inserting a case marker where one was missing) so that the parse could continue. Since use of a recovery strategy incurred some loss of confidence in the resulting meaning structure, deviant partial parses were inserted into a queue and explored in a *least-deviant-first* manner. More serious deviations from a grammatical parse were examined only if no parse was found with less serious deviations. Grammatical parses, of course, were attempted first.

In the next sections we examine the design of MULTIPAR more closely and discuss the mechanisms by which least-deviant-first exploration of the search space was accomplished.

2.1. Organization of Knowledge in MULTIPAR

In anticipation of our discussion of knowledge structures for adaptive parsing, it is useful to distinguish between semantic and syntactic knowledge in MULTIPAR. Semantic knowledge is completely contained in the set of *entity definitions* that define the actions and objects in the domain. An entity [HAYE84] is a generalized caseframe; a simplified example, taken from MULTIPAR's operating system domain, is given below. The entity shown encapsulates the system's knowledge about moving a file or directory from one place in a file structure to another. Note that entity definitions may be embedded according to the semantic composition princi-

ple, i.e. values of slots in entities may be terminal symbols or they may be other entities satisfying a set of local semantic constraints and global semantic predicates.

```
(EntityName   MoveCommand
      SemanticCases (
            Object (FileObjDesc[3] or DirectoryObjDesc)
            Source (DirectoryObjDesc or LogicalDeviceObjDesc)
            Destination (FileObjDesc or DirectoryObjDesc)
            Location (DirectoryObjDesc or LogicalDeviceObjDesc))
      Constraints (
            (Required Object Destination)
            (Destination FileObjDesc ⟩ Object FileObjDesc)
            (Object DirectoryObjDesc ⟩ Destination DirectoryObjDesc)
            (Object DirectoryObjDesc ⟩ Source LogicalDeviceObjDesc)
            (Object DirectoryObjDesc ⟩ Location LogicalDeviceObjDesc))
      SurfaceForms (
            (SFName ImperativeCaseframe-Canonical
                  Head ⟨movehead⟩
                  DirectObject Object
                  Cases (
                        (Preposition ⟨sourcepreps⟩
                         Bind Source)
                        (Preposition ⟨destpreps⟩
                         Bind Destination)
                        (Preposition ⟨locpreps⟩
                         Bind Location))))
      InstanceTemplate (
            Action 'MOVE
            Deviations Deviations
            Source (
                  IsA (IsA in Object)
                  Name (Name in Object)
                  Extension (Extension in Object)
                  Directory (Directory in Object or
                              Directory in Source or
                              Directory in Location)
                  LogicalDevice (LogicalDevice in Object or
                                 LogicalDevice in Source or
                                 LogicalDevice in Location)
                  ObjDesc (Description in Object)
                  SourceDesc (Description in Source)
                  LocDesc (Description in Location))
                  Destination (
                        ...)))
```

[3] FileObjDesc means file object description; other abbreviations are similar.

There are two sources of semantic information in the entity definition:

- **SemanticCases** define the basic structure of the entity in terms of the other entities that are its components. In the example, the entity MoveCommand has an object (the file or directory to be moved), a source (the directory or logical device to move it from), a destination (the file or directory to move it to), and a location (the directory or logical device in the context of which the move takes place). The semantic cases do not tell MULTIPAR how to recognize the entity in an utterance; rather they denote what other entities may or must be referenced in the input as part of recognizing the entity defined.
- **Constraints** specify, for any instance of an entity, both the predicates on individual cases and the relations that are required to hold between cases. The first constraint of MoveCommand indicates that the Object and Destination cases are required to be present for any instance of this action. The second constraint means that if the Destination case is filled by a file, then the Object case must also be filled by a file. The remaining constraints have analogous interpretations. MULTIPAR enforces constraints of both types insofar as it is able. In other words, it prefers parses in which the constraints are satisfied, but will accept (as a grammatical deviation) utterances in which the constraints are violated.

The location of syntactic information in MULTIPAR is more diffuse. Specification of the acceptable syntax can be found partly in the remaining fields of the entity definition. In particular:

- **SurfaceForms** act as a bridge between domain semantics and syntax; the parsing strategies they point to (through the SFName attribute) recognize the syntactic forms that indicate a reference to this action or object. In general, an entity may have more than one SurfaceForm, each of which corresponds to a different form of syntactic expression for the same underlying semantic case. There is, however, a 1-to-1 correspondence between SFName and parsing strategy in the implementation.[4]

  The remaining attributes (Head, DirectObject, etc.) are specific to the SurfaceForm and are best thought of as the values provided to parameters in the parsing strategy. Symbols like ⟨movehead⟩ are non-terminals in a grammar used by the DYPAR pattern-matcher [CARN81, BOGG85] and expand in the course of computation (e.g. ⟨movehead⟩ → move | transfer). SurfaceForm slot fillers that are not surrounded by "⟨⟩" are SemanticCase names. When a case name is present, the strategies appropriate to parsing entities of that type are invoked subject to the Constraints. Consider the DirectObject case in the SurfaceForm shown. The strategy that uses this form will be passed an instance of the MoveCommand entity definition and an input segment to work on. When the strategy reaches the point at which it expects the direct object, it retrieves the SemanticCase name from the SurfaceForm. Since the Object case can be filled by either a FileObjDesc or DirectoryObjDesc, the strategy will try to parse the current subsegment as each (unless the Destination has already been filled with a FileObjDesc as per

[4] Parsing strategies are simply functions written in Commonlisp.

the first constraint). To do this it calls those strategies corresponding to the SurfaceForms in the entity definitions of FileObjDesc and DirectoryObjDesc respectively. The result of the call may be an InstanceTemplate (see below) that is then bound to the case name.

- The **InstanceTemplate** binds syntax to meaning representation. The slot fillers associated with each attribute act primarily as a map between the values bound to the SemanticCases during the parse and a canonical meaning structure for the utterance. In this way, "move [c410jf90] foo to my directory" produces the same meaning structure as "move foo from [c410jf90] to my directory," even though in the former the directory name would be found when parsing the Object as a FileObjDesc and in the latter it would be found when parsing the Source as a DirectoryObjDesc.

While the syntactic information in the SurfaceForms and InstanceTemplate is fairly explicit, there is a great deal of syntactic knowledge left implicit in MULTIPAR's parsing and recovery strategies. It is the parsing strategy that knows how to organize the information given in the SurfaceForm. The parsing strategy dictates, for instance, the order in which the cases may appear in the input. The parsing strategy also contains the recovery strategy whose purpose is to compensate for the user's deviation from a particular, expected grammatical construction. The recovery for a violated expectation is placed at the point of the expectation.

The final piece of implicit linguistic knowledge we consider is the tolerance level associated with a recovery action. Tolerance levels, or *flexibility increments*, will be discussed more fully in the next section. Here we note only that they encode statically the relative severity of each of the specific types of deviation permitted by MULTIPAR.

2.2. The Control Structure

If the least-deviant parse is to be found efficiently, the exploration of alternative paths in the search space must be controlled carefully. For example, spelling correction should be tried before hypothesizing a missing word, hypothesizing one word before hypothesizing five. Moreover, a strategy failing to find a correct parse for some utterance does not mean that the recovery actions should be invoked immediately. It may be that another strategy will find a grammatical and semantically consistent parse; if so, that strategy should be tried before any recovery actions are attempted.

To control the exploration of the search space, competing alternatives within a parsing strategy are explicitly specified using SPLIT statements. When a SPLIT statement is encountered, the computation divides into separate branches; each branch has a *flexibility increment* indicating the additional degree of ungrammaticality tolerated by the associated continuation. The values of the increments form a partial ordering on the severity of the deviations for which MULTIPAR can compensate. A partial parse is generated along each branch, and each computation may proceed from the SPLIT statement independently of the others.

```
(Split (+0 continuationA)
       (+1 continuationB)
       (+3 continuationC)
              ...)
```

Thus, a SPLIT statement such as the one above produces a three-way branch in the search tree. ContinuationA does not result in a gain in flexibility of the associated parse; this path would be followed unless the particular grammatical expectations it represents are not satisfied by the utterance. Continuations B and C, if they are ever pursued, will result in gains of one and three respectively; each represents the violation of a grammatical expectation but the recovery action in continuationC creates more uncertainty about the validity of the resulting meaning structure.[5] All branchpoints in the parse deposit their untried combinations in the same global agenda, indexed by their global *flexibility level* (explained below). The structure of a global agenda looks like:

```
[Agenda: (0 ContinuationA ContinuationD ...)
         (1 ContinuationB ...)
         (3 ContinuationC ContinuationE ...)
         (5 ContinuationF ...)
                  ...]
```

MULTIPAR starts with an agenda containing one or more top-level strategies at the 0-flexibility level, and a global flexibility level set to 0 (i.e. full grammaticality). As each strategy is applied it can post additional continuations in the agenda at level 0 or higher. Continuations are posted when SPLIT statements are encountered, and the flexibility levels at which they are posted correspond to the sum of the global flexibility level and the flexibility increment of the continuation. When a continuation completes execution it is removed from the agenda.

MULTIPAR pursues all the continuations at the 0-flexibility level. Ideally, exactly one of these yields a complete and consistent parse, indicating that the input was grammatical and unambiguous. If more that one parse is generated, we have true ambiguity, which must be resolved by external means. If no parse is generated at the 0-flexibility level then the input was ungrammatical and MULTIPAR goes on to the next-higher flexibility level in the agenda, selecting this one as the new global flexibility level. The process is repeated at this flexibility level, perhaps resulting in more continuations being posted on the agenda. If no parse is found, the global flexibility level is increased again. In this fashion, the search for a possible parse continues, seeking the least deviant interpretations first. If the global flexibility level reaches some preset threshold, too much uncertainty has accrued and MULTIPAR abandons its parsing attempts.

---

[5] The non-determinism provided by the SPLIT statement is also exploited for handling normal ambiguities. When a strategy finds that a parse is ambiguous, a separate branch is created for each alternative parse. +0 flexibility increments are used for each locally-ambiguous but semantically-correct continuation.

3. CHAMP: Design for an Adaptive Parser

In MULTIPAR the loci at which expectations could go unmet were predetermined at fixed junctures in the grammar; these fixed locations corresponded to only a subset of the possible recovery points required by an input. In some sense, when we built MULTIPAR we simply extended our notion of what we would consider part of the grammar to include those classes of extragrammatical constructions most often produced by interface users. Our goal in building CHAMP,[6] an adaptive parser, is to create a system that acquires an idiosyncratic grammar with minimal ambiguity and maximal coverage via experience with a user. In essence, we need to remember the recovery actions performed in the presence of an unfamiliar utterance, and then augment the grammar for that particular user with a representation that can parse the same structure directly. In this way a kernel grammar can be extended to handle new forms, or variants of existing forms, whether they be true grammatical deviations, or grammatically valid extensions to a kernel whose coverage was less than complete. It is clear that the model of deviance and recovery embedded in MULTIPAR is inadequate to this task, for expectations can go unmet at any point in the input, and no methods for automatically extending the grammar were provided.

There is a powerful assumption implicit in the approach to adaptation that we describe. Specifically, the method can achieve our goal only if the individual user is reasonably consistent in his or her linguistic expressions over time. If the user's grammar is not *self-bounding* then the system's idiosyncratic language model will continue to grow to intractable proportions. In Section 4 we formulate the idea of *self-bounding* linguistic behavior more precisely and examine empirical evidence in support of the phenomenon. We note here only that linguistic regularity, like other forms of stereotypy, does appear with task repetition. Thus we limit the applicability of our approach to tasks requiring fairly frequent user interaction.

In order to take advantage of regularity in language use we must generalize the key concepts of deviation and recovery, starting from the following definitions:

- **Kernel grammar**: the lexicon, caseframe concept definitions, default semantic values, and syntactic forms that are present in the system prior to interaction with any user.
- **System canonical form**: any syntactic form in the kernel grammar.
- **Deviation**: the violation of an expectation as embodied in a syntactic form. Exactly four types of deviation other than misspelling are possible: the lack of expected text (deletion error), the existence of unexpected text (insertion error), the presence of text instead of the text expected (substitution error), and the presence of expected text in an unexpected location (transposition error). Note that misspellings correspond exactly to these four categories but the constituents are letters in words, rather than constituents in sentences [DURH83].
- **Recovery action**: one of insertion, deletion, substitution, transposition or spelling correction. Each action compensates for one type of deviation. The recovery

[6] CHAMeleonic Parser: it adapts to its environment.

Form: schedule | add [a | an] meeting | appointment | ⟨meal⟩ [⟨marked-case⟩]*

0 deviations: "Schedule lunch with Bob at 12"

1 deviation: "There is a meeting at AISys at 7: p.m."

Recovery:
1. substitute "schedule" for "There is"

2 deviations: "Meeting with Philip on Thursday, June 12 10 to 11: a.m."

Recovery:
1. insert "Schedule" before "Meeting"
2. insert "from" before "10"

3 deviations: "I have to be at AISys for a meeting at 7: p.m."

Recovery:
1. substitute "schedule" for "I have to be"
2. transpose "at AISys" to post-nominal position
3. delete "for"

4 deviations: "There is a business meeeting at AISys on June 11 7: p.m.to 8: p.m."

Recovery:
1. substitute "schedule" for "There is"
2. spelling correct "meeeting"
3. insert "from" before "7:00"
4. either substitute "meeting" for "business meeting" or delete "business"

Figure 1. Sample deviant sentences and recover actions

action for a deletion error is insertion; for an insertion error, it is deletion; transposition and substitution are their own inverses. Sample sentences at various levels of deviation and the recovery actions that correct them are shown in Figure 1.[7]

- **User canonical form**: a new syntactic form created in response to the successful application of one or more recovery actions in the presence of deviant input. These forms are stored in the *user model*, and used in parsing subsequent input more rapidly and with greater confidence.
- **User model**: the union of system and user canonical forms that grows in a self-bounded manner with increasing user interaction.

Using these definitions we can describe *adaptive parsing* in the following manner. For each utterance, we first attempt a grammatical parse, i.e. we try to under-

[7] The form given at the top of Figure 1 translates to: the word *schedule* or *add* optionally followed by the word *a* or *an* followed by the word *meeting* or *appointment* or *breakfast*, *lunch* or *dinner* followed by zero or more occurrences of a marked case.

stand the utterance using the current lexicon, defaults, and system canonical and user canonical forms. In the presence of an utterance that is deviant in this context, the general recovery actions may be applied. If the recovery is successful, a new form is constructed and added to the model.

In essence, the parsing process is one in which we look for the simplest combination of deviations that "explains" the input and store some generalization of that explanation as a derived user canonical form. We constrain the search for a meaning representation both by our method of indexing forms, which proposes candidates accounting for more of the utterance first, and by ordering the subsequent search space in a least-deviant manner, as in MULTIPAR. In addition, related forms, (a system canonical form and those user canonical forms derived from it) are ordered both by recency and frequency of use over time for purposes of guiding the search.

How does this approach achieve our goal? In the initial interactions with a user, we expect the number of deviant utterances to be fairly large, although the degree of initial disparity between the kernel grammar and the user's grammar clearly depends on both the choice of system canonical forms and the user's idiosyncrasies. Those utterances that can be explained without too much uncertainty (for example, containing at most two deviations) will give rise to user canonical forms. Note that an utterance that is more than two deviations away from every form in the user model at a particular point in time may become accessible through the adaptation process. This happens if, for example, a form lying between the grammar and the desired utterance is learned first; although previously unparsable, the desired utterance may now be within two deviations of the enlarged grammar. Thus, in Figure 1, if the one and two deviation forms had already been learned, then the four deviation form would be learnable with respect to the user model at that point; only recovery actions two and four would be necessary.

By extending the grammar in response to real instances of behavior, we increase the system's linguistic coverage in useful directions while incorporating only the ambiguity intrinsic to the user's idiolect. Having found that different users exhibit different preferences and idiosyncrasies, with little overlap in their acquired user canonical forms (see Section 4.3), our method of individually tailoring extensions to the grammar into unique user models proves an appropriate solution.

If linguistic behavior is self-bounding with frequent use, then eventually the user's grammar will stabilize. During this process, the likelihood that an existing form serves as an adequate explanation of an utterance increases until most of the learning behavior of the system consists of reordering rather than increasing the search space. While a user's behavior remains constant, the system's performance should asymptotically approach the optimal performance given any stable inherent ambiguity in the user's grammar. Should linguistic behavior begin to vary again (as when a novice user becomes more expert and moves away from full sentences and toward a command language), there would follow a period of decreased performance as the system learns new forms, then a return, with stabilization, to faster response.

It is important to note that the quality and utility of the "explanation" one can derive varies with the system's coverage, which is in turn a function of the history of experiences with a user. Yet self-corrective mechanisms exist within the

adaptive parser to minimize any detrimental effect a particular history might create. Consider the movement of the user's grammar over time. If a new form is a useful generalization of a true linguistic regularity, its predictive strength will reinforce its likelihood of use. Conversely, if a derived form is a poor explanation of an utterance, or reflects only a transient idiosyncrasy in the user's grammar, then after a time it will no longer be proposed by the indexing process; in effect, it will have been forgotten.

Having discussed the process of adaptation in a general way, we turn now to the specific declarative structures and control mechanisms around which the design is organized. In Section 3.4 we will bring the declarative and the procedural together in a detailed example of the system at work.

3.1. Representing the Task Domain

We have chosen as a task domain for the adaptive parser a subset of the activities of an executive assistant: maintaining a schedule of meetings and events, and arranging airline reservations when those events require travel. The number of actions and objects relevant to the domain is fairly small. In keeping a calendar, the actions are limited to viewing some portion of the current schedule or adding, deleting, or changing the value of an entry. In addition to meetings and events, objects in the domain include the different kinds of time associated with an event (e.g. start, end, duration), and types of locations, participants, and topics. For airline reservations the list looks similar: scheduling, changing and canceling flights, arrival and departure times and locations, flight numbers and passenger names.

Although the number of actions and objects is small, the domain has the typical degree of linguistic complexity. Consider the following hypothetical utterances which show the possible initial variation between different users expressing the same request:

User 1: Schedule a meeting at 2 with the Robotics Group for an hour.
User 2: Add a Robotics Group meeting from 2 p.m. to 3 p.m.

User 1: Show me my afternoon meetings.
User 2: Display every meeting after 12:30 p.m.

User 1: Cancel the one at 3.
User 2: Delete the 3 p.m.meeting.

User 1: Reschedule the meeting at 4 to tomorrow at 10 a.m..
User 2: Change the starting time of today's 4 p.m. meeting to tomorrow at 10.

User 1: Change the location to John's office.
User 2: Change the location of the meeting at 10 a.m. tomorrow to John's office.

Even for a single user reference to the same events may change over time [RICH84]:

Day 1: Schedule a meeting at 2 with the Robotics Group for an hour.
Show me my afternoon meetings.
Cancel the one at 3.

Reschedule the meeting at 4 to tomorrow at 10 a.m..
Change the location to John's office.

Day 14: Schedule Robotics Group, @ 2, 1 hr
Show afternoon
Cancel @ 3
Reschedule @ 4 to tomorrow @ 10 a.m.
Relocate to John's office

Day 28: Sc Robotics Group 2-3
Sh p.m. (or, just, "p.m. ?")
C 3
Res 4 tomorrow 10 a.m.
Rel John's ofc

As in MULTIPAR, we represent the domain concepts using caseframes. The general form is similar; an explanation of the fields follows the example:

```
(FrameName                Add
    Isa                   action
    Forms                 addforms
    InterSlotConstraints  (endtime ) starttime)
    Cases
        (c-object
            Isa (meeting seminar class meal trip)
            History nil)
        (c-date
            Isa date
            History nil)
        (c-start
            Isa hour
            History nil)
        (c-end
            Isa hour
            History nil)
        InstanceTemplate
            (action
                Map 'add
                Defaults nil
                Constraints nil)
            (object
                Map (and    (object Map c-object.isa
                                    Defaults nil
                                    Constraints (required))
                            (ref    Map c-object.ref
                                    Defaults nil
                                    Constraints ((ref = indefdesc)))
                            (specifiers    Map c-object.specifiers
                                           Defaults nil
                                           Constraints nil)))
```

```
(day
        Map (or c-date.day c-object.day)
        Defaults ((always? *currentday*))
        Constraints (required (day )= *todaysday*))) ...
(starttime
        Map (or c-start c-object.starttime)
        Defaults ((always? *currentstart*))
        Constraints (required))
(endtime
        Map (or c-end c-object.endtime)
        Defaults ((always? *nextstart*))
        Constraints (required)))
```

Some of the slots, such as **FrameName** and **IsA**, have straightforward interpretations. Others, such as **Cases** and **InstanceTemplate**, have counterparts in the entity definition examined in Section 2.1. The remaining slots are new, however, and worthy of note.

Recall that the InstanceTemplate is a way of packaging domain information in a canonical form. As is evident from the **Map** field, that information may be provided by a variety of bindings. For this reason the **Constraints** are attached to the slots in the InstanceTemplate rather than the cases (similarly, we talk about **InterSlot-Constraints** rather than InterCaseConstraints). In addition to private constraints, a slot may have associated **Defaults**. If, for example, the user fails to specify the day on which the object should be added to the calendar, the parser will first check the Constraints field to see if the information is required, then will look for a valid default in the current context. The kernel definition of Add instructs the parser to ask the user if the last day referred to is an appropriate value. As the system gains experience with the user, the value of a default or the context under which it applies may be modified by this kind of interaction.

Also new are the closely related fields **Forms** and **History**. The filler of the Forms slot is a pointer to the list of system canonical and user canonical forms that recognize references to this concept. As we try to interpret an utterance as a reference to an Add action, we will access the definitions of Add's casefillers. But these definitions are themselves caseframes that index their own forms. Just as we encode the utility of each Add form in the ordering of the Forms list, so we encode the utility of constituent forms in a case's History list. In fact, the ordering provided by the History list may override the ordering given in the Forms list of the nested caseframe. This is desirable if, for example, the user relies primarily on the system canonical form SC-*i* to refer to a meeting, but in the context of SC-1 (an addform) always uses SC-*j* to denote the object case. Currently we use the simple *move-to-front* algorithm to balance the influences of recency and frequency on the ordering:

1. Initially the Forms list contains only the appropriate kernel forms in a random order.
2. When a user canonical form is derived it is add to the front of the list.
3. When a known form successfully parses an utterance, it is moved to the front of the list.

Thus, as some forms fall into disuse and others dominate, the order of form names is altered to take advantage of the user's emerging idiosyncrasies. In this way we can also use Forms and History to control the amount of work required to understand a novel form; since a deviant utterance may partially index many forms, we use historical information to help select a candidate form for adaptation.

3.2. Representing the Learning Domain

Since the adaptive parser is learning primarily about language in a domain rather than about the domain *per se*, it is useful to distinguish between the task domain (activities of an executive assistant) and the learning domain (language). The caseframe representation discussed in the last section is used to describe the task domain; it is comparable to MULTIPAR's entity definitions in its encoding of the semantic knowledge in the system. To make the problem of adaptation tractable, we need declarative structures that capture the learning domain as well. These structures contain knowledge about language and syntax that was left implicit in MULTIPAR.

There are two sets of structures that work together to permit adaptation. First, there are the definitions attached to words and phrases in the lexicon. Two sample definitions are shown below. In the first entry from the lexicon, "add " is seen to be an element of the class of addwd. The entry for "schedule" would also show this membership. In contrast, "June" is more than just a lexical item. It is, by itself, a legitimate instance of a monthform – a system canonical form that recognizes references to months. Thus, the definition of June contains a fully instantiated InstanceTemplate for a month concept.

```
(add
      instanceof        (wd addwd)
      def               nil)

(June
      instanceof        (forms monthforms)
      definition        (month June))
```

"Addwd" and "monthforms" tie the lexical entries into the second important knowledge structure in the learning domain: the caseframe for canonical forms. A simplified example is shown below.

```
(FormName               Act1
      Belongsto         addforms
      Type              kernel
      Parent            nil
      Children          nil
      Strategy
            step1 (find-case m-dateforms c-date)
            step2 (find-case m-hourforms c-start)
            step3 (required find-terminal addwd)
            step4 (required find-case ggforms c-object))
```

Act1 is a system canonical form that recognizes one syntactic method of referring to the Add action. Because it is a kernel form, Act1 has no **Parent**; any user canonical form derived from it is stored as one of its **Children**. The link between the Add caseframe and Act1 is through the token "addforms" which is found both in Add's Forms list and Act1's **Belongsto** slot.

The key to adaptation lies in the explicit representation of the parsing **Strategy**. The parser itself is essentially a knowledge free interpreter that executes the "find procedure" in each step. When a step fails, the parser uses the general recovery actions to add continuations to the agenda. The general form for a strategy step has four parts (see step4 above): the first position is empty or contains the word "required" indicating that the step must consume input. The second position contains the name of one of the four step types from which all strategies are constructed; in addition to find-case and find-terminal (shown above), there is find-subform (a variant of find-case) and find-primitive (a variant of find-terminal). The third position is filled by the class name of the form or word expected at this point in the utterance. Finally, the fourth position may be empty or contain the name of a case to which to bind information returned by the find procedure.

3.3. Adaptation: Detecting and Recovering from Deviation

In order to learn an idiosyncratic language, we have moved away from a grammar-specific set of deviations and recovery strategies and toward a generalized notion of deviation and recovery which corresponds to finding the simplest explanation for any violated expectation that preserves the meaning of the utterance. The process of understanding is divided into three phases. In the first phase, the utterance is segmented into known and unknown token sequences. The result of segmentation is a representation of the input that reflects both grouping of lexical items and lexical level ambiguity.

Once segmentation is complete, the system begins the bottom-up, or voting, phase of the parse. The purpose of this phase is not to assign meaning but to provide evidence to the top-down phase about which canonical forms are most likely to account for the input. The kernel is small enough that all system canonical forms could be tried at the start, but as the user model grows this approach becomes impractical. The bottom-up procedure indexes those forms that seem to be able to cover the input; as a class of forms is predicted (for example, actionforms at the start of the top-down phase), members of the form class are ordered depending upon the number of nodes in the segmentation for which the voting phase found evidence. Syntactic forms that account for little of the input must necessarily lead to violated expectations; one way in which CHAMP differs from MULTIPAR is in being able to decide not to explore a form at level 0. The output from the voting procedure also helps the top-down phase "look ahead" at critical points in the parse to see, for example, whether there is evidence for a missing pre-nominal case in the post-nominal position.

Voting is possible because of the connections that are maintained between knowledge structures. When the steps in Act1's strategy (see previous section) are loaded into the system, the form class or word class is updated. For example (Act1 3)

is created as an annotation to the class addwd, (Act1 1) for m-dateforms, etc. When a node is created during segmentation its class membership is copied from the word or phrase definition in the lexicon; since the class annotation points into the strategy structures, the lexical referents in the utterance have been tied to the basic form steps that recognize them. Recall that the forms themselves are members in a form class hierarchy (thus, Act1 isa addform and addforms are actionforms). By following these links, evidence can be propagated upward, allowing each strategy to inherit the nodes covered by its constituents.

Finally, we reach the top-down phase where parsing commences. If the sentence is non-deviant with respect to the current user model, the parsing process amounts to straightforward caseframe instantiation. To begin, CHAMP takes the most-voted-for actionform and executes each step in the form's strategy in turn. If the step type seeks to fill a case, then the evidence list is consulted for the most-voted-for form in the case's form class, and the parser is called recursively. Parsing continues in this way until all of the input has been accounted for; the result is an instantiated InstanceTemplate for the appropriate calendar action.

If the sentence cannot be understood with the current grammar and default knowledge, at some point in the parsing process a deviation is detected. Since every word is now a possible locus for deviation, every strategy step must be permitted to fail, i.e., every strategy step is implicitly surrounded by a SPLIT statement that recovers from the general deviations represented by the failure of a find procedure. This uniformity permits us to remove the recovery mechanism from the strategies and place it in the interpreter. Thus, instead of SPLIT statements, the adaptive parser relies on an agenda like that described in Section 2.2 and two new control structures: the POCS (Path Of Computation Stack) and the LSQ (Leftover Step Queue). The purpose of these structures will become clear as we explore exactly how detection and recovery are accomplished.

Of the four error types, two may occur only when a find-terminal or find-primitive procedure fails. In a substitution error, the user has replaced the expected word or phrase with a different lexical item intended to achieve the same reference. In an insertion error, the user has added extraneous text at the current point in the input (see Figure 2). At the moment at which the expectation is violated, it is impossible to tell which of these errors has occurred. Since some kind of deviation is present, we must create a continuation which recovers from each. A continuation is a "snapshot" of the current computation that will permit processing to continue. It contains the current configuration of the LSQ and POCS, the name of the current strategy, the number of the current step, the type of deviation detected, and the unused portion of the segmented input.

If we are trying to explain the deviation as a substitution error, we create a continuation noting the step in which the substitution occurs and selecting a word from the expected word class as a synonym for the unknown segment. The step is pushed onto the POCS and the unused segment is updated. If a substitution continuation is restarted the step will have appeared to succeed. If this path is ultimately successful, a new entry will be created for the lexicon indexed by the previously unknown word or phrase and defined by copying the entry of its synonym.

To create an insertion error, the snapshot notes the inserted segmentation node and the number of the failed step. In addition, the unused portion of the input

Form: schedule | add [a | an] meeting | [(marked-case)]*
Error Type: Example (deviant input italicized)
Insertion: *Please* schedule a meeting on Tuesday with John. Substitution: Schedule an *appointment* on Tuesday with John. Deletion-terminal: Add a meeting *Tuesday* with John. Deletion-case: Add a on Tuesday meeting with John. Transposition: *On Tuesday* add a meeting with John.

Figure 2. Sample sentences with single deviations

is updated. Thus, if an insertion continuation is restarted the current step will be retried at a new point in the utterance. If the explanation turns out to have been correct a new lexical entry for this phrase will be added to the lexicon, with its own word class. A user canonical form is derived from the form being executed when the deviation was detected. It is created with a find-terminal step expecting a member of that class inserted at the appropriate point. Finally, an annotation is added to the class name pointing into the new strategy; in this way the bottom-up phase will vote for the new form if the inserted segment reoccurs.

Transposition and deletion errors are possible explanations when either a find procedure seeking a terminal or a find procedure seeking to fill a case fails (again, see Figure 2). For these deviations, detection is more problematic than above, for when a strategy step fails, each of the following is a possibility:

1. The step can succeed using text further on in the input. This corresponds to the initial detection of a transposition error. The step is pushed onto the LSQ, the step counter advanced and processing continues.
2. The text satisfies a step that we had previously placed on the LSQ. For steps on the LSQ that may succeed at this point in the input, a continuation is placed on the next level of the agenda in which the the transposition is noted, and the step is removed from the LSQ. If the continuation is restarted, the transposed step is executed first. If the path eventually succeeds, a user canonical form reflecting the change in step order is derived from the form in which the transposition occurred. The form class is updated as well.
3. The step will never be satisfied. This cannot be detected until the parser finishes executing the current strategy. If all the steps in the LSQ are optional or try to fill cases for which default values exist in the user model then no deviation has occurred; the current path can continue at the current level. But for each step on the LSQ that is required and for which there is no default information, either a *case deletion* or *terminal deletion* has occurred.

   A terminal deletion causes a continuation to be posted at the next level of the agenda with the step removed from the LSQ and the deletion noted. If the continuation is restarted, the success or failure of the path relies on the remaining contents of the LSQ. If the path succeeds, the user canonical form built in this situation removes the deleted step from the parent form.

A case deletion requires interaction with the user for its resolution; it generally signals the need to acquire default information. Thus, the current step is set to correspond to the the first required case on the LSQ and a continuation noting the need for user help is posted at the next level. If this continuation is restarted the system will query the user for the missing information. The success or failure of this path relies on the success of the interaction and the remaining contents of the LSQ. If the error can be resolved in this way, a new default for the case in the current context may be added to the concept frame recognized by the incomplete form.

3.4. An Annotated Example

To make clear how the components we have described combine and interact to learn a user's style, we present the following extended example. We will rely on the definition of Add and Act1 given previously as well as the assumption of suitable definitions for constituent cases and their forms.

The kernel form Act1 is adequate to parse the imperative sentence:

"On July 10 at 7 pm schedule a meeting with John".

Consider, however, the system's behavior when a user who prefers declarative phrasing types the reasonable variant:

"On July 10 at 7 pm there will be a meeting with John".

1. The utterance is segmented:

```
(G1 G2 G3 (OR G4 G5) G6 G7 G8 G9 G10 G11 G12)

G1: ST 1 END 1 DEF (WD DTMKR))
G2: ST 2 END 2 DEF (FORM MONTHFORMS))
G3: ST 3 END 3 DEF (IS NUMBER))
G4: ST 4 END 4 DEF (WD HRMKR))          ) "at" can mark a
G5: ST 4 END 4 DEF (WD LOCMKR))         ) location or hour
G6: ST 5 END 5 DEF (IS NUMBER))
G7: ST 6 END 6 DEF (WD DAYNITEWD))
G8: ST 7 END 9 DEF (IS TOKSEQ))         ) "there will be"
G9: ST 10 END 10 DEF (WD ARTICLEWD))
G10: ST 11 END 11 DEF (WD MEETINGWD))
G11: ST 12 END 12 DEF (WD PRTMKR))
G12: ST 13 END 13 DEF (FORMS STUDENTFORMS))
```

2. The bottom-up phase creates the following evidence for actionforms. The sublists correspond to the step numbers within the strategy that may consume the given segmentation node. Since Act1 and Act2 are distinguishable only by the word class of their verbs (add vs. delete), their evidence lists are identical.

```
ACT1: ((G1 4 2) (G2 4 2) (G3 4 2 1) (G4 4 1) (G5 4)
       (G6 4 2 1) (G7 4 1) (G9 4) (G10 4) (G11 4) (G12 4))
```

```
ACT2: ((G1 4 2) (G2 4 2) (G3 4 2 1) (G4 4 1) (G5 4)
       (G6 4 2 1) (G7 4 1) (G9 4) (G10 4) (G11 4) (G12 4))
```

3. The top-down phase begins and Act1 is chosen from the list of most-voted-for action forms. The parser begins execution of Act1's strategy with step1 (find-case m-dateforms c-date). The find-case looking for a marked dateform executes successfully; nodes G1, G2, and G3 are consumed, corresponding to "On July 10." The instantiated InstanceTemplate returned by the find procedure is bound to c-date and pushed onto the POCS along with the step number.
4. Step2 (find-case m-hourforms c-start) is executed. Since Act1 does not permit the location of the object to precede the verb (as in "At CMU schedule a meeting"), the sense of "at" given by node G4 is used. Nodes G6 and G7 are also accounted for and the resulting structure and the step number are pushed onto the POCS.
5. In step3 (required find-terminal addwd), the call to the procedure find-terminal fails. The parser must now try to compensate for a violated expectation using the general recovery actions. An insertion continuation is posted to level 1 of the agenda in which the current step number remains 3 but the unused segment begins with G9. A substitution continuation in which node G8 is tied to the class addwd is also added to level 1. A final level 1 continuation explains the error as a deletion of the required verb: the current step advances to 4 but the unused segment remains the same. Anticipating a transposition error, a continuation is posted at level 0 in which step3 is on the LSQ and step4 is current. Because a deviation was detected, the current branch of the computation returns failure.
6. The parser checks level 0 of the agenda for other forms or suspended computations that will give a grammatical interpretation to the input. It finds the anticipated transposition of the verb and restarts this path. Step4 also fails to consume G8 and a number of level 2 continuations are posted. (No anticipated transposition of step4 will be posted to level 0 because it is the last step).
7. At this point the parser must try any forms covering the same number of segmentation nodes as Act1. It will find only Act2 (a deleteform) and processing identical to the above will ensue. When the level 0 transposition for Act2 fails, the level 0 bucket will be empty, the global flexibility value will be incremented, and we will move to level 1 of the agenda.
8. The insertion continuation for Act1 is restarted with the unused segment of the input beginning at G9. Step3 is retried at this position but it also fails, creating snapshots at level 2.
9. The substitution continuation for Act1 is restarted with "there will be" (node G8) considered as a synonym for "schedule". The parser successfully executes step4 (required find-case ggforms c-object) with the unused segment beginning at G9. C-object is bound to the result and the unused portion of the input becomes nil.
10. Having completed the strategy, the parser makes certain that the LSQ contains no required steps, then returns successfully. The InstanceTemplate for the add frame is used to create the specific instance of the action referred to in the

utterance. The top-level control notes that all the input has been consumed, meaning that this path has led to a consistent interpretation of the input at level 1.

11. The agenda must still be checked for competing interpretations. The deletion continuation for Act1 fails when step4 is executed at node G8. The insertion and deletion continuations for Act2 will fail as above. The substitution continuation for Act2 will fail after the object case is filled because the deletion action requires a definite article to introduce the noun phrase.
12. With level 1 exhausted, we are confident we have found the least-deviant interpretation of the input. Depending upon the degree of deviation of the utterance and the severity of the effect of performing the action, the system may check the effect of its interpretation with the user. In this example, the system would show the additional entry on July 10th and ask the user the yes/no question, "Is this what you wanted?" (If the answer was no, the system would go on to level 2).
13. When the user responds affirmatively, the parser finds the deviation notation for this path in order to update CHAMP's knowledge structures. The substitution notation indicates that node G8 should be added to the lexicon as a member of the class addwd. The new entry looks like:

```
((there will be)
      instanceof        (wd addwd)
      def               nil)
```

14. Next the history of the path is recorded: the token "Act1" is moved to the front of the add frame's form slot. Similarly, the form and history slots for the constituent concepts referred to by the utterance are updated.
15. Finally, the parser returns the instantiated InstanceTemplate to the database calendar routines which perform the update.

The next time the user tries to add an object to the calendar using the declarative "there will be" phrasing, the bottom-up phase will advance Act1 as the unique preferred form and step3 of that strategy will execute successfully at level 0.

4. Hidden Operator Experiments with Professional Secretaries

In prior discussion (Sections 1 and 3) we identified two conditions that limit the applicability of adaptation in user modeling. We review these now and add a third:

1. Within-user consistency – The tractability of the particular computational model we have presented requires that the user's grammar converge on a stable subset of natural language. We have called this asymptotic movement toward a fixed set of utterances "self-bounding behavior." If the user's past behavior is not an indicator of his future behavior, adaptation as we have described it is no longer useful because the user's grammar grows without bound.

2. Across-user variance – If all users employ the same subset of language, i.e. if there is no natural idiosyncrasy of expression, then the correct approach to interface design may not require adaptation. By "employ the same subset of language" we mean that each user employs (almost) all the forms in the subset and only forms in the subset. In this situation, a methodology such as that of Good *et al* [GOOD84] would be preferred; through a generate-and-test cycle one builds a single system incorporating the complete common subset. Note, however, that even if all users converge eventually on a consistent subset, if their paths to that point are highly idiosyncratic, adaptation may still ease the way. Our data indicate disparate linguistic subsets across users.
3. Partial user adaptability – To the degree that a user can adapt to the system's limitations, such adaptation should be exploited, allowing the user and system to "meet in the middle." However, our experiments show that most of the adaptation should be performed by the system, permitting the user to focus on her task and relieving her of the cognitive burden of learning the boundaries of the interface's sublanguage.

The behavioral assumptions that underlie our approach are apparent: we believe that with frequent interaction a user will develop stereotypical and idiosyncratic language patterns; further, a system that permits the natural expression of these patterns through adaptation will result in better task performance than a system that forces the user to adapt to it. In the series of hidden-operator experiments described in the remainder of this section, we found that in conditions of frequent use the expected behaviors were clearly evident. The underlying assumptions for automated adaptation were indeed borne out.

4.1. The Base-Adaptive Experimental Condition

In a hidden-operator experiment[8], users were told that they would be typing the input to a *natural language learning interface* that would increase its knowledge of English while helping them keep an on-line calendar for a busy professor/entrepreneur. In each of nine sessions, the user was given between ten and twelve pictorial representations of changes to be made to the calendar. Two examples of these *subtasks* are shown in Figure 3. She was told to interpret each picture and effect the change by typing her commands as if she were "speaking or writing to another person". Although no time limit was enforced, users were told that it was more important to attempt than to accomplish each subtask. They were given the general rule that three tries were sufficient effort before moving on to the next picture.

If the user's utterance was deemed either parsable or learnable (containing at most two deviations) by the experimenter, the user was told that the intended action had been carried out and the calendar and language model were updated as appropriate. If the utterance was too disparate with respect to the current model of

[8] In a hidden-operator experiment, the user believes she is interacting with a computer system, but, in reality, the feedback is produced by the experimenter at a remote terminal. In this way, alternate designs of an interface and all its capabilities can be tested prior to actual construction.

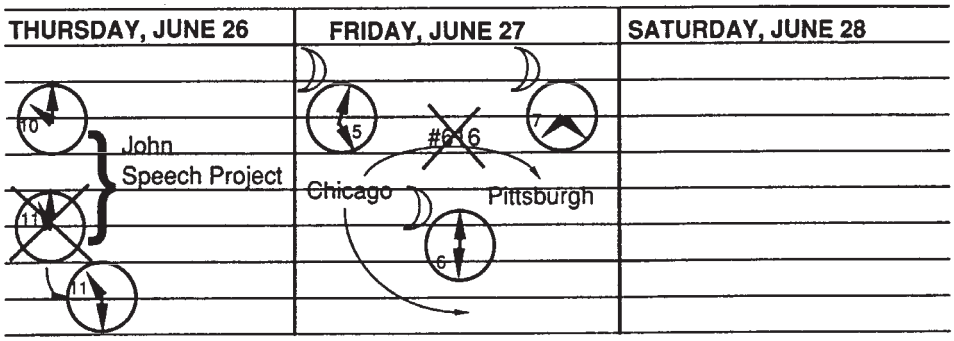


Figure 3. Sample pictorial stimuli

the user's grammar, the user was generally told which parts of the utterance were understood and asked to "please try again."

4.2. Control Conditions

One counterargument to an adaptive approach maintains that the user will naturally adapt to the system's limitations faster than the system can usefully adapt to the user's idiosyncrasies. To evaluate this claim we designed two variations on the basic experimental condition.

In the No Adaptation condition, the experiment was conducted as outlined except that users were not told the system was a learning interface and the kernel grammar was never updated. Although the system remained more permissive of extragrammaticality than the average natural language interface, the boundaries of the grammar were rigidly fixed.[9] Any increase in performance would therefore be attributable to the user's adaptability.

The Echo-Adaptive condition was included in response to arguments by Slator *et al* that users want to learn, and will learn, a mnemonic command syntax and domain-specific vocabulary [SLAT86]. When given the graphics command language equivalent to their natural language input, his users were able to gradually integrate the terser forms into their grammar. It was not known whether individuals would show the same propensity in interactions with an adaptive system. Users in the Echo-Adaptive condition were given the same instructions as those in the Base-Adaptive condition except that they were told the system would display a paraphrase of their utterance in an "internal form" that they were free to incorporate into their commands or to ignore as they saw fit.

[9] Because we felt it important to keep the criteria for parsability the same under all conditions, we extended the kernel grammar for this condition in one respect: use of articles was made optional. This is the most common ungrammatical construction displayed by interface users. If we had not extended the grammar in this way, most sentences would have involved more than two deviations and the results from this condition would have been comparable primarily to session one of the base condition.

None of the users who participated had prior experience with natural language interfaces. Each was employed as a secretary or executive assistant in either a business or university environment. Although not every user maintained a calendar for her employer, each had kept a personal calendar for at least one year.

Sessions were run at 24 hour intervals (approximately) for all users except User 1 whose sessions were run twice a day with a five hour break. The number sessions of varied in the two control conditions in accordance with the availability of the users. Table 1 summarizes the relevant user information.

Table 1. Summary of user information

User	# years other calendar	# years own calendar	Employer	Condition	# sessions
User 1	2.25	1	univ-psych	Base-Adapt	9
User 2	0	1.33	company	Base-Adapt	9
User 3	1.5	2	company	Base-Adapt	9
User 4	2	4	company	Base-Adapt	9
User 5	0	4	univ-compsci	Echo-Adapt	3
User 6	0	10	univ-compsci	No Adapt	3
User 7	0	"always"	company	Echo-Adapt	5
User 8	4.5	40	univ-english	No Adapt	5

4.3. Results and Discussion

The results for users of the adaptive system indicate a high degree of self-limiting behavior, converging upon a user-specific recurrent subset of English. With the exception of case ordering – which was strongly influenced by the stimuli – there was virtually no overlap of user canonical forms at the sentential level; non-sentential similarities were limited primarily to missing determiners and some common phrase marker substitutions. Figure 4 shows the approximate number of changes to the grammar over time for each of the four Base-Adaptive condition users, Figure 5 shows the same information for the Echo-Adaptive condition. Both figures indicate by user and session the number of new constructions per sentence (cps) which is calculated as the total number of changes to the grammar divided by the total number of parsable sentences. The number of unparsable sentences in a session is given in parentheses next to the fraction. Cross-hatched areas of Figure 4 indicate sessions following the introduction of supplementary instructions to perform the task as quickly as possible. The purpose of these instructions was to encourage the user toward a terse, possibly idiosyncratic, command language.

Figures 4 and 5 show both the importance of how the initial grammar is chosen and the difference in the rates at which language stabilization occurs. Observe that User 1's grammar requires the least learning and stabilizes quickly. She began with terse utterances and a vocabulary that was quite close to the kernel grammar chosen

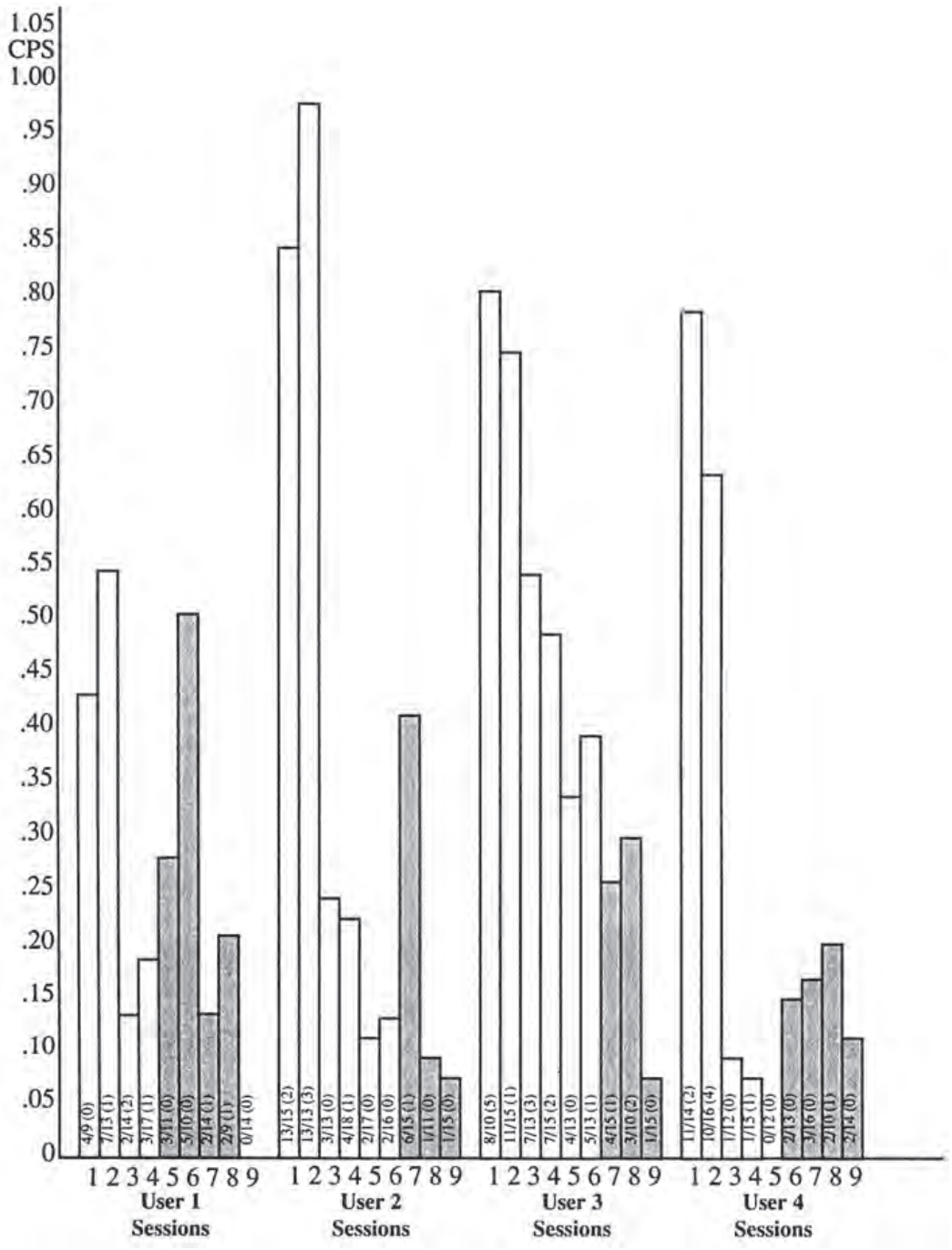


Figure 4. Self-limiting behavior in the Base-Adaptive condition as measured by the number of new constructions/number of parsable sentences (CPS)

for the system. User 2's grammar showed less initial resemblance but she employed it consistently and thus stabilized quickly as well. User 3 had a tendency to be both polite and verbose, so her utterances were more likely to contain new forms and stabilization occurred more slowly. User 4 showed the most rapid stabilization. It is interesting to note that although the new forms she introduced in sessions one

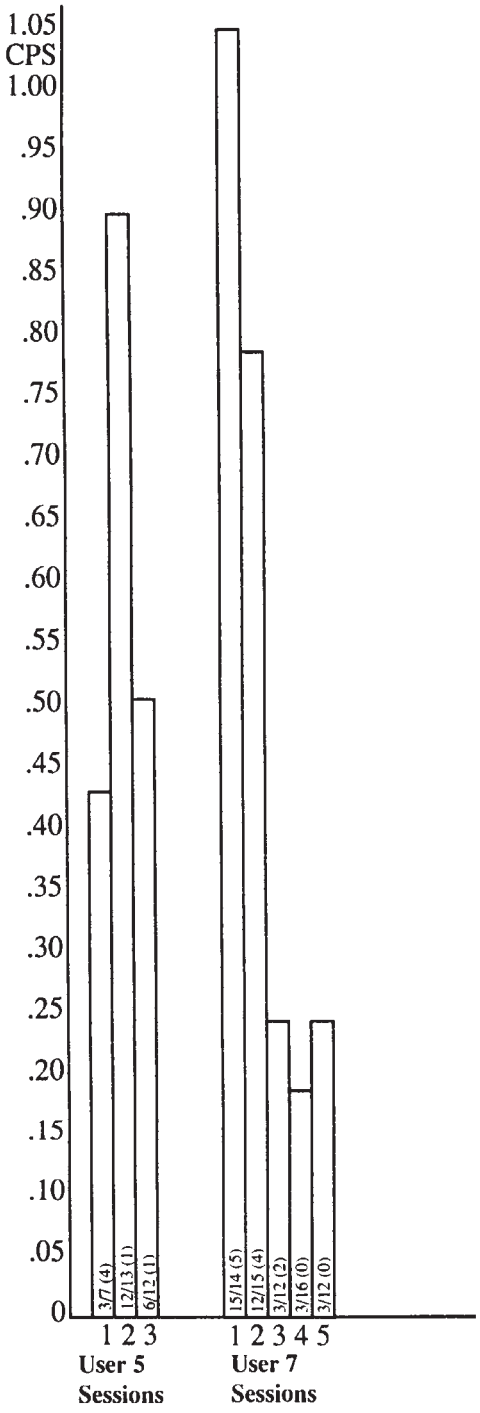


Figure 5. Self-limiting behavior in the Echo-Adaptive condition as measured by the number of new constructions/number of parsable sentences (CPS)

and two were not terse, they were quite "natural," i.e. they were used consistently and without modification in subsequent sessions.

User 5's behavior is of limited interest as she became ill after the third session and her participation was discontinued. User 7's graph shows much the same trend as that found in the Base-Adaptive condition. Neither User 5 nor User 7 showed any integration of the command language paraphrase, putting our results at odds

with Slator's general claims. Our experience did, however, replicate that of Good *et al* [GOOD84] and Cobourn [COBO86]: there is a strong assumption on the part of users that if the system can generate it, the system can parse it. Thus, although neither user in the Echo-Adaptive condition chose to employ the command language provided, many of our users displayed idiosyncrasies that were clearly based on the system's output. User 7, for example, came to construct all instances of the schedule command (and only this command) using the template: ⟨start-time⟩ – ⟨end-time⟩ ⟨date⟩ ⟨meeting or meal⟩, as in "12:00 – 1:30 June 11 lunch with Andy". Utterances thus constructed deviate from a kernel form only by the deletion of the verb and indefinite article; thus, the user canonical form is easily learned. In addition, these utterances closely conform to the template used by the experimenter to seek confirmation of an interpretation, e.g.:

```
Do you want:
      12:00-1:30          lunch, Andy
```

Table 2. Evolution of an element in User 2's grammer

	Session								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
SC: from <time> to <time>	-	2/6	-	5/9	4/4	2/3	-	6/6	1/3
UC1: from <time> till <time>	2/4	-	-	-	-	-	-	-	-
UC2a: beginning at <time> and ending at <time>	2/4	3/6	6/7	-	-	-	-	-	-
UC2b: beginning at <time> ending at <time>	-	1/6	1/7	-	-	-	-	-	-
UC3: <time> to <time>	-	-	-	4/9	-	1/3	1/1	-	2/3

To illustrate what we mean by a self-limiting grammar, the evolution of a particular referring phrase is shown in Table 2. The five forms that User 2 employs to refer to a time interval are given in the left portion of the figure. The actual use of a form in each session is shown as a fraction of the number of opportunities for its use. Initially, the system grammar contains only the system canonical form (SC). Using the general recovery actions, the experimenter equates the user canonical (UC) forms as they appear. Note the transience of UC1 and UC2 and the eventual reliance on the terse system canonical form and an even terser user derivative (UC3).

Figures 6 and 7 display the results for the No Adaptation condition. Recall that the purpose of this variation was to study the claim that people adapt well enough to obviate the need for system adaptation. We measure user adaptation in

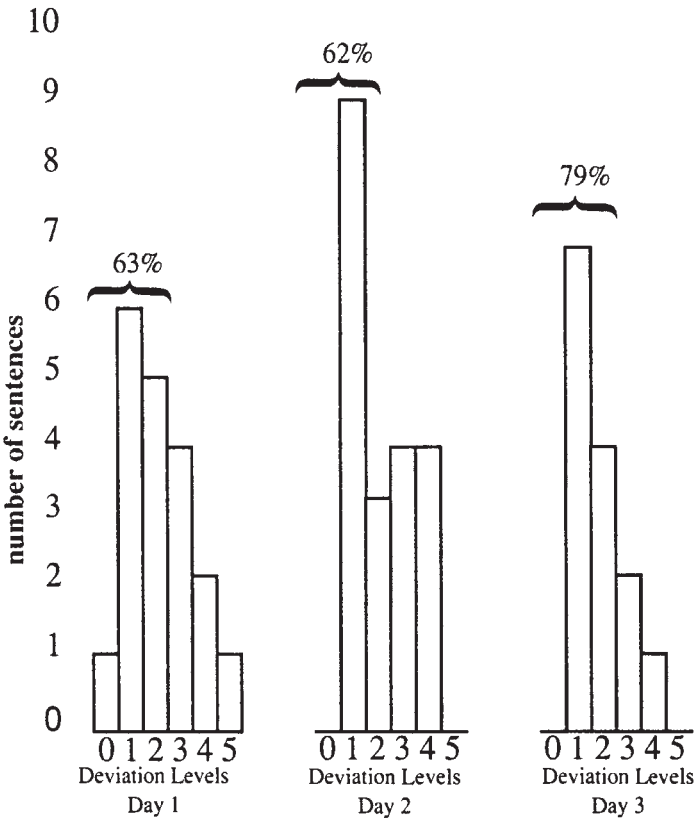


Figure 6. Number of utterances parsable at each deviance level in No Adaptation condition, User 6

this condition by categorizing each utterance by the minimum number of deviations required to parse it. If the user is adapting to the limitations of the system we should see a general increase in parsable sentences (those containing zero, one or two deviations). Further, grammatical utterances (zero deviations) should come to dominate. The latter conjecture is based on the belief that slower response time and more work are negatively reinforcing. Since the interpretation of utterances with one or two deviations took longer and had to be confirmed by the user, finding forms in the kernel grammar should have been preferred.

The graph for User 6 clearly shows the increase in parsable sentences. Her behavior is interesting in two respects. First, it was reminiscent of User 1's in that User 6's grammar was fairly close to the system kernel, especially with respect to her vocabulary, and she tended to rely on relatively terse forms. As a result, there were, in general, fewer loci for deviation. When an utterance did meet with failure, her next try almost always succeeded. Second, only one of her 54 sentences was without deviation, refuting our previous conjecture that grammatical utterances would be sought. Note, however, that deviant forms that are terse can be checked fairly quickly against the grammar so the penalty paid in time for non-grammaticality is not very great. Still, it must be admitted that User 6 did not seem to mind the extra work involved in confirming the system's interpretation of her command.

We will defer discussion of User 8 until we have examined two additional measures of performance. Table 3 shows the number of subtasks completed by each

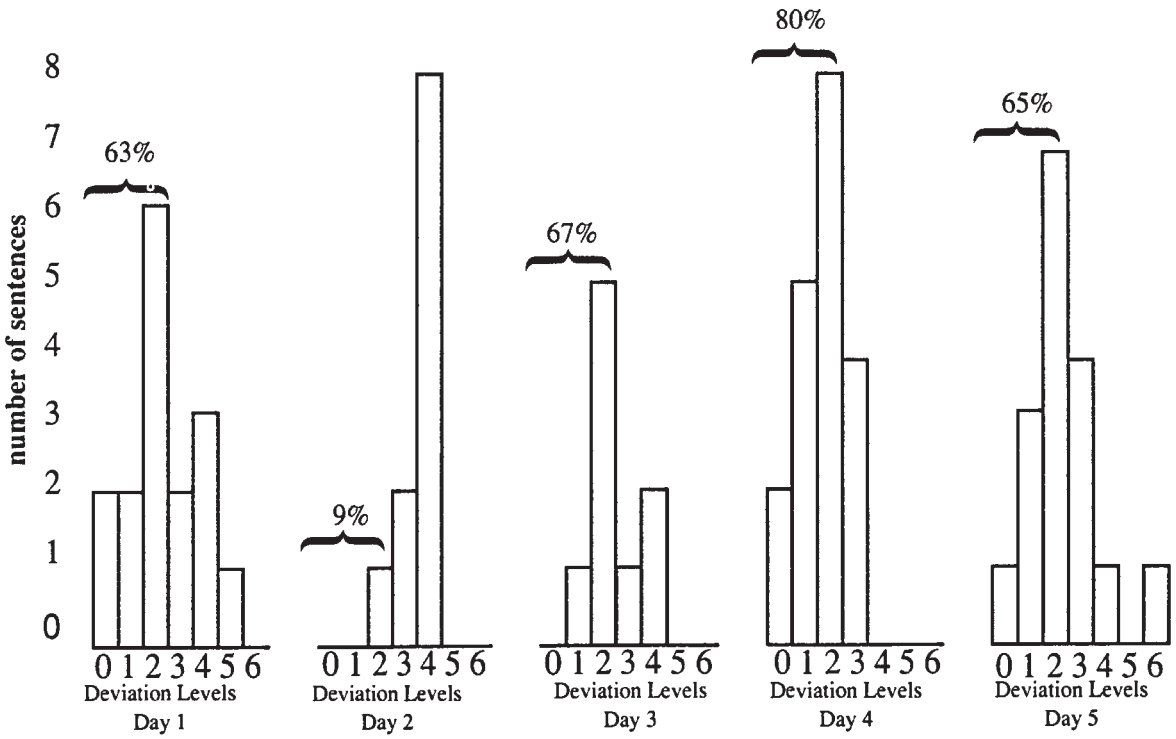


Figure 7. Number of utterances parsable at each deviance level in No Adaptation condition, User 8

user in each session. Table 4 shows the related metric of time taken per subtask. With respect to the former measure, users other than User 8 generally managed to effect the changes in each subtask they tried.[10] They rarely required more than three linguistic attempts per subtask in the initial sessions and averaged slightly more than one attempt per subtask in later sessions.

The general decrease in task time is a predictable result of self-limiting behavior. Users in the adaptive conditions accomplish the task in less time as the grammar of the simulated system comes to reflect their own and more sentences are judged parsable rather than learnable. When the supplementary instructions are given (Base-Adaptive condition only) and the users are encouraged to employ simpler, terser forms, task times increase as the new forms are learned. When these forms stabilize, response times fall to new lows since the shorter forms take less time to check against the grammar. Because she is consistent in her use of what worked in the past, and because her initial grammar conformed so closely to the kernel, User 6 shows the same general trend in time spent as those in the adaptive conditions.

User 8 shows neither consistent decline in the time taken each session nor in the number of unparsable sentences (see Figure 7). The combination of her work experience and the content of her utterances leads us to conclude that little of her

[10] There were four sessions in which a user skipped all or part of one or more subtasks. In these cases no utterance relating to the subtask appears in the log file.

Table 3. Number of completed subtasks, arranged by user and session ([#] indicates number of subtasks skipped)

		Session (total number of subtasks possible)								
		1(10)	2(12)	3(12)	4(10)	5(10)	6(10)	7(11)	8(10)	9(11)
Adaptation / Base	User 1	7[1]	11	12	10	10	8[2]	11	10	11
Adaptation / Base	User 2	10	12	12	10	10	10	11	10	11
Adaptation / Base	User 3	8	11	12	10	10	10	11	10	11
Adaptation / Base	User 4	10	12	12	10	10	10	11	9.5[.5]	10.5[.5]
Adaptation / Echo	User 5	6.5	9	12						
Adaptation / Echo	User 7	10	12	12	10	10				
No Adaptation	User 6	10	12	12						
No Adaptation	User 8	7	1	4.5	5	6				

behavior can be attributed to task misunderstanding. Put simply, she was unable or refused to adapt.

Her linguistic style, like that of User 3, can be characterized as verbose. In response to a typical utterance the system would tell her which segments it could parse and ask her to try again. She would then try typing exactly those segments just echoed, with no connecting text. When the terse form met with failure she would gravitate back to overly explanatory forms.

User 8's performance in session two was so poor (in 31 minutes she produced one parsable sentence with two deviations) and her frustration so great that she was given hints prior to beginning session three about how to use the system more effectively.[11] Although the help appeared to improve her performance in sessions three and four, in session five there is still a preponderance of unparsable forms. Contrast this with the behavior of users in the other conditions, most of whom had cut their number of unparsable utterances to zero by session five. Finally, note her low values for the number of tasks completed. These were due largely to the fact that she never managed to find a parsable form for an entire class of subtasks (those involving the "change" action). Taken as a whole, User 8's performance is a strong counterargument to the notion that everyone finds adaptation natural.

[11] Specifically, she was told to try to type simple, but fully grammatical sentences and to think of the system as someone to whom she was writing instructions. She was the only user who was given this aid.

Table 4. Number of minutes/subtasks arranged by user and session (* indicates approximate value due to damage to log files; + indicates supplementary instructions given this session)

			Session								
			1	2	3	4	5	6	7	8	9
Adaptation	Base	User 1	5.5*	3.0	2.5	2.8	1.7+	2.4	1.6*	2.1*	1.0
		User 2	5.0	3.5	2.0	3.5	2.2	2.1	2.2+	1.3	1.3
		User 3	6.3	3.2	2.7	3.3	1.7	2.3	1.6+	2.0	1.0
		User 4	3.4	3.3	1.7	2.1	1.1+	1.6	1.5	1.5	1.1
	Echo	User 5	8.5	5.3	3.3						
		User 7	4.7	3.8	2.5	3.3	1.5				
No Adaptation		User 6	4.3	3.7	2.0						
		User 8	6.3	31.0	7.1	5.0	5.3				

5. Concluding Remarks

Natural language interfaces can, in principle, be made much more effective by automated adaptation to individual users. Analysis of data from hidden-operator experiments demonstrated that there are significant differences among the active linguistic patterns generated spontaneously by different users, but that each user is fairly consistent across sessions spanning several days. These observations suggest that as an integral part of individualized user models, one should store the linguistic preferences and idiosyncrasies of each user. Such information allows for direct and accurate interpretation of the user's commands and queries in the given interface. Individualized user models, however, must be acquired automatically in order to be of any practical significance – the designer cannot hand-analyze the behavior of each individual user and hand-tailor the kernel grammar to be consistent with the observed behavior.

This chapter developed a method for automatic user adaptation starting from a kernel grammar of system canonical forms. In brief the process may be summarized as follows:

- The system begins with the kernel grammar. If the user is known to the system, and user canonical forms have been synthesized for her, these are loaded atop the kernel, forming the complete user model.

- As user inputs are processed in the interface, the system attempts to parse them using the (possibly augmented) kernel grammar and recovery strategies as needed.
  - If exact (non-deviant) parsing is possible, the user's intention is understood and no recovery or adaptation methods are required.
  - If exact parsing is not possible, recovery strategies are attempted in a least-deviant-first manner. These strategies are based on lexical spelling correction or manipulating grammatical constituents by composing the following four simple operations: insertion, deletion, substitution and transposition. If the application of a small number (typically two) of such operations results in a parse, the user is asked to confirm the system's interpretation, and the correct action is carried out.
- If a user-confirmed deviant parse occurred, the transformed grammar construct is subjected to a conservative generalization process, based on recent explanation-based learning methods [MITC86], then added to the individual user model.

Presently these methods are being developed and refined.[12] The MULTIPAR project demonstrated the feasibility of several key concepts: focused parse recovery, least-deviant-first search methods in the space of possible partial interpretations, and the parse-continuation computational architecture as an effective implementational substrate. However, MULTIPAR was unable to modify the grammar or change its behavior with experience (whether globally or tailored to an individual user). These limitations were caused in small part because the parse recovery mechanisms were encoded as arbitrary LISP functions, and in large part because MULTIPAR defined the notion of deviation in a very narrow manner. The five grammar adaptation operations discussed in this chapter provide a conceptually simpler and (allowing for composition) more powerful recovery method. Such a set of strategies performs well-defined operations on the current grammar on an "on demand" basis, i.e. when attempting to parse a grammatically deviant input. The grammar constructs that result are expressed in a declarative language to facilitate generalization and further adaptation.

Acknowledgements. This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD), ARPA order No. 4976, monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520 and in part by scholarship funding provided by the Boeing Company. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, Boeing or the US government. The authors would like to acknowledge the contributions of Steven Minton who designed and implemented MULTIPAR's control structure, the secretarial assistants from Carnegie-Mellon University, Lexeme Corporation and UNILOGIC, LTD. who participated in the experiments, and Marion Kee and Ben McLaren who helped conduct the experiments.

[12] The knowledge structures and algorithms discussed in Section 3 have been implemented in CHAMP, but the generalization process – though applied successfully in other projects at CMU – remains to be implemented and tested.

6. References

[BOGG85], [CARN81], [CARN83a], [CARN83d], [COBO86], [DURH83], [FAIN85], [GOOD84], [HAYE84], [MINT85], [MITC86], [RICH84], [SCHU85], [SLAT86]

Part III

Exploiting User Models

Introduction

In Part III, a number of possible applications of user models in dialog systems are examined in closer detail. These include the exploitation of user models for tailoring object descriptions to the user's level of expertise, correcting user misconceptions with respect to taxonomic object classification, monitoring a user's impression of objects in an evaluation-task dialog, and adapting an expert system's response behavior to the background knowledge of its users. Besides the domains for which the described systems stand as representatives, there exists a large number of other possible applications, some of which have been described in Parts II and IV. For surveys on systems not reported on in this volume, see [WAHL*, KASS88].

Cécile Paris investigates how assumptions about the user's level of expertise can be employed by a dialog system for tailoring descriptions of complex physical objects. The author compared a number of encyclopedias for children and adults and found that different types of descriptions are used in each of these two text types, which are obviously aimed at two different types of readership. For readers who can be expected to possess some background knowledge about the domain, objects or concepts are described in terms of their subparts and the properties of these subparts. This pattern of description has been identified in studies on text organization by [MCKE85a, MCKE85b] who called it a 'constituency schema'. Descriptions for "naive" readers, on the other hand, essentially describe the processes associated with the operation of the object. This pattern is called a 'process schema' by Paris. The author explains her findings by assuming that readers with background knowledge – in contrast to "naive readers" – can be expected to figure out how parts fit together to form an object capable of performing a function. Providing such information would thus contradict Grice's [GRIC75] maxim of quantity and should therefore be avoided.

Paris' analysis suggests that the user's level of domain knowledge affects not only the *amount*, but also the *kind* of information provided in descriptions. The author suggests that, based on a user model, dialog systems should also employ these two distinct strategies for generating descriptions which are well-adapted to the user. Since users are hardly ever either completely naive nor completely expert, she also suggests that a dialog system should *combine* both strategies, for example in describing complex physical systems: Whenever the user model indicates that the user possesses local expertise about an involved object which is to be described, the constituency schema can be employed; otherwise the process schema must be used. Paris outlines the points in each description schema at which a switch to the other schema is possible.

The proposed principles have been implemented by the author in TAILOR, a system which – roughly – takes a request for an object description as its input, accesses its user model, and generates a conceptual representation of an object description adapted to the user's level of expertise. After its concepts have been replaced by lexemes, this description is passed on to a generator which unifies it with a functional grammar to produce English sentences. The user model of TAILOR contains individual assumptions about the user's experience with respect to the domain of discourse. These assumptions are represented by a list of those items in the system's knowledge base which are known to the user, and by information about whether the user understands the basic concepts underlying the domain of discourse. The operation of the program is illustrated by a detailed example. In a final section, Paris briefly presents heuristics that might be used to acquire this user model.

A frequent type of user misconception is the misclassification of objects (i.e. their subsumption under a wrong superconcept in a concept hierarchy). Another type is the belief that objects have certain attributes or attribute values which they do not actually have. **Kathleen McCoy**'s contribution deals with how to behave after such misconceptions have been detected and classified as belonging to one of these categories. For each type of user misconception, she proposes three strategies for generating appropriate responses. The generated response patterns are then transformed into natural language responses using McDonald's [MCDO80] MUMBLE system.

In all response strategies, the similarity of objects plays an important role. Object similarity is defined via the similarity of the *attributes* of these objects. However, not all attributes are taken into account, but only those which are *salient* within the current dialog context. McCoy discerns three types of salience, each stemming from a different source: static salience encoded directly in the generalization hierarchy, dynamic salience gained by an attribute being explicitly mentioned in a discourse, and perspective salience gained by viewing the domain objects through the active perspective.

Perspective salience turns out not to be a substructure of the concept hierarchy, but to form an independent structure (namely a set of highlighted attributes) which is orthogonal to the concept hierarchy. Detailed examples from the field of financial advice are given to illustrate the effect of perspective on attribute salience and on one of the above-mentioned response strategies. Suggestions are made as to what influences the choice of perspective in a given dialog.

The chapter by **Anthony Jameson** is concerned with the modeling, monitoring and exploitation of a rather complex belief type, namely the user's belief about the user model (in the notation of [KOBS*], system beliefs are here of type S (3.1.1.) v (3.1.3.) U (3.1.1.) v (3.1.3.) S (3.1.1) U (1.1.1) v (2.1.1.)), and about the motivation of the system. The issues that arise in connection with this sort of belief can be most easily studied in everyday dialogs between human participants. A speaker generally not only exploits some model of the listener's beliefs, interests, and/or goals in deciding what to say; s/he also takes into account the fact that the listener will have

some *image* of this model and the way it is being used. The listener employs this image, which may be vague and/or inaccurate, to make substantive inferences that go beyond the conventional meaning of the speaker's dialog contributions.

Jameson's specific examples and his simulation model IMP are restricted to a specific kind of dialog setting, namely *evaluation-oriented dialog*. In this setting, the goal of one participant (the listener) is to evaluate some object along one or more dimensions, and that of the other participant (speaker) is to provide the information the listener needs. (In the specific examples used in the chapter, the system simulates a job applicant who is describing herself to a personnel interviewer.)

The issues raised and the model's performance are discussed in a sequence of increasingly complex situations. The first two show how the speaker, given a simple sort of information about the listener's values and expectations, can volunteer information which should produce a realistic object evaluation according to the listener's standards – or, if the speaker so desires, an evaluation which is either positively or negatively biased. (Note that non-cooperative motivation is seldom dealt with in research on dialog systems.) If the listener is aware of the values and expectations the speaker ascribes to him/her, s/he can go beyond the literal meaning of an utterance by considering alternative utterances that the speaker *would* have produced if certain things had been true about the object. This *pragmatic interpretation* of dialog contributions can greatly increase the efficiency of communication; but it can yield incorrect inferences if the listener's image of the speaker is not entirely accurate - e.g. if the listener does not recognize that the speaker is biased. By choosing his/her dialog contributions carefully, the speaker can correct a discrepancy between his/her image and his/her actual criteria for generating dialog contributions - or, if s/he desires, intentionally create and/or maintain such a discrepancy. The task of creating and maintaining an image becomes more complicated in a situation where the speaker is not entirely free to decide what aspects of the object to mention but must respond to questions of the listener.

The most complex situations considered by Jameson show that there exist two sorts of uncertainty concerning the image the speaker is projecting. First, the listener will often be quite uncertain about what evaluation standards the speaker ascribes to him. In some cases (though not all), the result is that the listener will be less inclined to make pragmatic interpretations of what the speaker says. Second, the speaker may of course be quite uncertain about the assumptions the listener is making about him/her. This uncertainty needn't cause problems as long as it is matched by a similar degree of uncertainty on the part of the listener; but if the listener has definite assumptions about the speaker's model of him/her and the speaker does not know what these are, the speaker will often not know what s/he can say without risking puzzlement and misunderstanding.

The most general implication of this chapter for practical user modeling is that it is not sufficient to devise techniques for making and exploiting correct inferences about the user. The system (and its designer, respectively) must also anticipate the specific inferences that the user will make on the basis of the system's behavior (a number of methods by which the system can do this are discussed in [WAHL*] under the heading 'anticipation feedback'). Jameson presents potential methods of conveying to the user a realistic image of the system's user model and its manner

of exploitation – or at least an image which is sufficiently indefinite to discourage the user from misinterpreting the system's behavior.

In the final chapter of this part, **Robin Cohen** and **Marlene Jones** explore the possible role of a user model in an envisaged large-scale expert system for educational diagnosis which is to support the assessment of school students' learning difficulties. Such an assessment, in general, includes an analysis of the student's developmental history, medical and behavioral information and, above all, the result of a number of educational tests concerning both general abilities and achievement in areas in which the student's performance is deemed to be inadequate. The system should be able to report and explain its results to widely different users, including experienced psychologists, parents, resource room teachers, regular classroom teachers, principals, etc.

Cohen and Jones investigate how a user model can assist in adapting system responses to the background knowledge and the goals of these diverse user groups. They propose that, in order to be able to do so, the user model must contain assumptions about the user's domain knowledge (including knowledge about diagnostic procedures, test interpretation rules and possible remedial activities); about his/her familiarity with the student's background (family facts, medical history) and test results concerning the student as well as their interpretation by the system; and about the user's goals and preferences, both on a global and on a speech act level. A simple stereotype approach (cf. [RICH*]) is not sufficient for modeling all these assumptions since background knowledge, for instance, might vary considerably within the same user group. An individual user model must therefore also be constructed (which possibly supplements and overrides the selected stereotype).

Such an envisaged user model would enable the diagnosis system to not just *answer* user questions, but also to add relevant information that the user might not know (e.g. about the student's background), but which is necessary for him/her to understand how the system arrived at this answer. In explanations, on the other hand, the system can leave out facts which the user can be assumed to know already. A model of the user's goals and preferences is necessary, for example, in order to take into account his/her bias with respect to what tests to administer (e.g. psychologist: those with which s/he is familiar; principal: those which can be used in funding formulae; parents: as few as possible to minimize stress). Cohen and Jones also outline possible enhancements of their model, particularly with respect to the recognition and treatment of user plans and user misconceptions, and with respect to incompatible appraisals of specific psychological methods.

8 The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise [1]

Cécile L. Paris

Abstract

A question-answering program providing access to a large amount of data will be most useful if it can tailor its answers to each individual user. In particular, a user's level of knowledge about the domain of discourse is an important factor in this tailoring if the answer provided is to be both informative and understandable to the user. In this research, we address the issue of how the user's domain knowledge can affect an answer. By studying texts, we found that the user's level of domain knowledge affected the kind of information provided and not just the amount of information, as was previously assumed. Depending on the user's assumed domain knowledge, a description can be either parts-oriented or process-oriented. Thus the user's level of expertise in a domain can guide a system in choosing the appropriate facts from the knowledge base to include in an answer. We propose two distinct descriptive strategies that can be used in a question-answering program, and show how they can be mixed to include the appropriate information from the knowledge base, given the user's domain knowledge. We have implemented these strategies in TAILOR, a computer system that generates descriptions of devices. TAILOR uses one of the two discourse strategies identified in texts to construct a description for either a novice or an expert. It can merge the strategies automatically to produce a wide range of different descriptions to users who fall between the extremes of novice or expert, without requiring an a priori set of user stereotypes.

1. Introduction

The tailoring of answers according to a person's domain knowledge frequently occurs in human communication. For example, an explanation of how a car engine works aimed at a child will be different than one aimed at an adult, and an explanation adequate for a music student is probably too superficial for a student in mechanical engineering. We have found further evidence of this phenomenon in nat-

[1] This research was supported in part by the Defence Advanced Research Projects Agency under contract N00039-84-C-0165, and the National Science Foundation grant ISI-84-51438.

urally occurring texts [COLL62, BRIT63, NEW67, WEIS73, CHEV78, ENCY82, BRIT84], where the information presented to readers varies with their assumed level of general knowledge.

Likewise, a question answering program that provides access to a large amount of data to many different users could be more effective if it could customize its answers to each user, retrieving from its knowledge base the facts that are most appropriate and useful for a given user. Much research to date has focused on tailoring an answer depending on a user's goals, but customizing an answer depending on what the user knows about the domain of the question has been overlooked. This is an important factor in tailoring an answer if the answer provided is to be both informative and understandable to the user. The answer should not provide information that is obvious to the user [GRIC75]. However, if the answer assumes knowledge that the user does not have, it may be very hard (if not impossible) for the user to understand the answer [WILS86, DAVS84]. In this paper we show the feasibility of incorporating the user's domain knowledge, or *level of expertise*, into a generation system and address the issue of how this factor might affect an answer.

Through an analysis of texts directed towards audiences at the two ends of the knowledge spectrum (naive and expert), we found that two distinct discourse strategies were used in texts describing objects. We postulate that the writers' choice of strategy might be based on the assumed domain knowledge of the expected readers. If so, then the reader's level of knowledge about the domain affects the *kind* of information provided as opposed to just the *amount* of information. In previous generation systems, the user's domain knowledge affected only the amount of detail provided in a response [WALL82].

In this work, we show how the two discourse strategies found in texts can be used to provide answers to users whose domain knowledge falls anywhere along a knowledge spectrum, from naive to expert. We have implemented these strategies in TAILOR, a computer system that generates descriptions of physical objects.

TAILOR uses one of the two discourse strategies identified in texts to construct a description for either a novice or an expert. It can merge the strategies automatically in a systematic way to produce a wide variety of different descriptions for users who fall between the extremes of novice and expert. This means TAILOR is able to generate descriptions to a whole range of users, rather that just for an *a priori* set of user stereotypes.

1.1. Previous Work on User Modeling in Question-Answering Programs

In studying the factors involved in tailoring the content of an answer to a user, research to date has focused mainly on the problems of inferring and using user goals, plans and beliefs [APPE82a, APPE85, CARB83, CARB88, CARB*, MCKE85c], recognizing and dealing with misconceptions [KAPS82, MCCO83, MCCO86, MCCO88, MCCO*, QUIL88, QUIL*], and superposing various stereotypes [RICH79b, RICH*]. The issue addressed here differs from these because we are not concerned with the users' goals in asking the question, nor with correcting

their view of the domain, but rather with providing an answer that is optimally informative given how much the user knows about the domain. We are not interested in *building a user model* using stereotypes (as was Rich), but in *determining an answer* based on a user model involving user types. As in [MCCO86, MCCO88], we are more concerned about using information from the user model to generate an answer than building the model itself.

While the need for a model of the user's domain knowledge in question answering systems has been noted by various researchers [LEHN77, MCKE85b], few programs have actually had one. The HAM-ANS system [HOEP84b] has a model of the user's knowledge, but this knowledge is mainly used for anaphora resolution and production. In our work, we are more interested in studying how a user's knowledge affects the *content* of an answer as opposed to its *phrasing*. Wallis and Shortliffe [WALL82], who have used the naive/expert distinction in providing an answer (or explanation), did so mainly by giving more or less detail, without addressing the issue of whether the level of detail was the only important factor to vary. The issue we confront in this work is identifying the role played by a user's level of knowledge in determining the content of an answer. The UNIX consultant (UC) [CHIN86] uses a user's knowledge level about the UNIX system to provide help to its users. UC, however, uses stereotypes for both the user and the knowledge base (set of unix commands). Stereotypes for the knowledge base include "simple," "mundane" and "complex." UC matches the user type against the command type to decide on the answer. In this work, we are looking at a different kind of domain, the domain of complex physical objects, in which this categorization of the knowledge base is not possible. For example, there is no reason to believe that knowing about microphones indicates more expertise about the domain of complex devices than knowing about telescopes. Another approach is thus required. Furthermore, we would like to be able to tailor answers to users whose domain knowledge level falls anywhere along a knowledge spectrum without necessarily having to classify users in several different stereotypes. Finally, we are more concerned in this work with exploiting the user model whereas Chin is concerned with building it.

1.2. The Domain

In our work, we are mainly concerned with describing complex devices such as telescopes, telephones and disk drives to users with different levels of expertise. Our choice of domain has been motivated by RESEARCHER, a program being developed at Columbia University. RESEARCHER reads, remembers, and generalizes from patent abstracts written in English [LEBO83, LEBO85, LEBO86]. The resulting knowledge base is organized in a generalization hierarchy. The abstracts describe complex devices in which spatial and functional relations are important. In this domain, the amount of information contained in the knowledge base is very large and the information can be very detailed. Moreover, the knowledge base contains several different kinds of information: spatial, functional, and attributive (properties attached to objects). A program can choose from among facts representing different kinds of information about an object, and facts at different levels of detail in the knowledge base, making the decision process a complicated one.

A request for the description of an object cannot be translated into a simple database query and thus cannot be answered by a straightforward retrieval from the knowledge base [TENN78, MCKE85b]. There are no clear constraints on what information should be included in the answer. Since the amount of information contained in the knowledge base is very large and the information very detailed, a program cannot just state all the facts contained in the knowledge base about the object as there will typically be too many. Rather, it needs to select a subset of facts to present to the user. As the answer will be composed of several facts, a generation program needs to organize these facts in order to construct from them a coherent text [MCKE85b]. When a generation system can choose among many facts, a user model representing what the user presumably knows about the domain can guide the system in choosing facts that the user understands and does not already know (and cannot easily infer), thereby improving the resulting answer.

Descriptions are important because they can be used to answer other types of questions. For example, to compare two objects, it may be necessary to describe each of them. Furthermore, with a knowledge base of physical objects, users are likely to request descriptions. Generating descriptions is thus a good first step towards developing a fuller question-answering program.

In order to focus on how the level of expertise affects a description, we have not considered how the goal of the questioner could affect the description. It is clear that in a sophisticated question-answering program the user goal should also play an important part. An answer for a user whose goal is to buy an object should include different kinds of information than an answer for a user who wants to repair this object. Detecting and using the user's goal to provide an appropriate response has been the focus of extensive research [APPE85, CARB83, MCKE85c]. In this work, being more concerned with the role played by the user's domain knowledge, we simply assume that users want to find some information about an object. A description should provide meaningful information about an object and allow the user to build a mental functional model of the object. Therefore, we assume that the goal of a description is to help the user construct a mental functional model of the object under consideration.

2. Identifying What Needs to Be in the User Model

Our goal is to provide a characterization of the role of the user's domain knowledge in generating descriptions that is computationally usable by a generation system. Even though we will not be addressing the problem of how to *determine* how much the user knows about the domain, we still have to ask what *kinds* of knowledge a user can possess about a domain that can affect generation and that can be explicitly represented in a user model. Instances of these kinds of knowledge will be the information contained in our user model. Having identified what needs to be in the user model, we will take the user model as given, and study how a system can use the information about a user's domain knowledge contained in the user model to tailor the answer to each user.

Knowledge is multidimensional, in the sense that it is possible to be knowledgeable about a domain along several dimensions, such as historical, economical or functional. In identifying the kinds of knowledge a user might have, however, we restrict ourselves to the types of knowledge that are explicitly represented in terms of our knowledge base.

Analysis of natural language texts suggests the existence of at least two kinds of domain knowledge that affect the type of descriptions that can be provided in our domain:

- *knowledge about specific items in the knowledge base.* We define "knowing" about an object to mean knowing about the existence of the object, its purpose and how this purpose is achieved (that is, how the various subparts of the object work together to achieve it). Knowing about an object thus means understanding the functionality of the object and the mechanical processes associated with it.
- *knowledge about various basic underlying concepts.* In a domain of complex physical objects, such concepts include electricity and voltages.

We define an *expert* user as one whose knowledge about the domain includes functionality of objects and mechanical processes. An expert user knows all the underlying basic concepts and the majority of the generalized objects contained in the knowledge base for a particular domain. An expert user does not necessarily know about *all* the objects contained in the knowledge base, however, and thus might ask questions about objects he or she does not know about. Given an object that is new but similar to a known one, an expert user has enough domain knowledge to infer how the parts of this new object work together to perform a function. For example, the expert user might already know about particular instances of this object, or, on the other hand, about generalizations of the object. A *naive* user is one who does not know about specific objects in the knowledge base and does not necessarily understand the underlying basic concepts.

A user is not necessarily naive or expert, however. The level of expertise is really a continuum, with most users falling along this knowledge spectrum. For example, a user may know about several objects in the knowledge base. Such users would not be considered naive, but, as there are many objects they do not know in the domain, they would not be considered experts either. In this work, instead of rating a user as having some intermediate level of domain knowledge, we use explicit parameters to indicate the user's knowledge. These parameters are a list of items in the knowledge base which are known to the user and information about whether the user understands the underlying basic concepts.

Figure 1 shows examples of user models in TAILOR. In (a), the user has local expertise about the microphone and understands the concept of electricity. Lack of expertise is assumed about other objects and concepts. In (b), the user is a naive user, and the user model is empty. In (c), the user has local expertise about both the microphone and the telephone. The concepts of electricity and magnetism are also understood. We retain the terminology *naive* and *expert,* for users at the two ends of the knowledge spectrum, only as a short-hand notation.

Our emphasis is on studying how object descriptions can be varied when the parameters contained in the user model vary. We will not try to categorize a user's

(a) The user has local expertise about the microphone and understands the concept of electricity:
Local expertise: *microphone*
Basic concepts: *electricity*

(b) The user is a ◻ naive user:
Local expertise: *nil*
Basic concepts: *nil*

(c) The user has local expertise about the microphone and the telephone and understands the concepts of electricity and magnetism:
Local expertise: *microphone, telephone*
Basic concepts: *electricity, magnetism*

Figure 1. Representing the user model explicitly

domain knowledge, attempt to determine what levels exist between the two extremes, or categorize a user into one of several user types. By explicitly representing elements of a user's knowledge, we gain the flexibility needed to tailor descriptions to specific users falling anywhere along the knowledge spectrum, without requiring a predefined set of stereotypes.

3. Two Descriptions Strategies Found in Texts: Constituency Schema and Process Trace

3.1. The Textual Analysis

To develop effective strategies for tailoring a description to a particular level of expertise, we began by studying descriptions in a variety of texts: adult encyclopedias [BRIT84, COLL62] and junior encyclopedias [BRIT63, NEW67, ENCY82], manuals [CHEV78, WEIS73] and high school textbooks (i.e., [BAKE57, VERW62]). This range of texts was chosen because it provided a good source of descriptions[2] and because these texts seem to address audiences at the two ends of the knowledge spectrum: naive and expert. Texts from adult encyclopedias are directed at an audience much more knowledgeable in general than the audience addressed by high school textbooks and junior encyclopedias. Likewise, the Chevrolet manual is aimed at knowledgeable users (professional mechanics) while the other manual claims to be directed towards novices. These texts thus constitute a good starting point for studying the differences between the descriptions given to naive users

[2] We hope that by choosing several sources, stylistic differences on our results are minimized. We studied about fifteen examples from each encyclopedia and textbook and a few from the manuals.

and those given to experts in a domain. We studied descriptions of devices, taking the description of the same object in all sources whenever possible. This allows for good comparisons of the texts, both in terms of content and organization. The descriptions we have studied are generally several paragraphs in length.

Given the desire to focus on how a reader's assumed domain knowledge can affect a description, encyclopedia texts have an added advantage besides providing examples of descriptions. They are directed to a general audience and people turning to them do so for a variety of reasons. Yet, they all read the same texts and thus acquire the same information, regardless of their goal for reading these texts. Encyclopedia texts are not directly aimed to help achieve particular goals. Rather, they provide general information intended to be useful to achieve any goal a reader may have. Thus an encyclopedia must provide its readers with some information about an object, without knowing or attempting to address a reader's specific goals. This property of encyclopedias gives us the opportunity to focus on how the level of expertise affects a description, without considering how the goals and specific beliefs of the reader could affect it.

We analyzed the different texts using methods developed by other researchers [HOBB78b, HOBB80, MANN84, MCKE85b], decomposing paragraphs in terms of their primitive rhetorical structure in an attempt to find consistent structures in the texts. We found that the texts fell into two groups: most of the descriptions in the adult encyclopedias entries and in the car manual for mechanics were organized around object subparts and their properties, while the descriptions in the junior entries and in the car manual for novices traced process information.

3.2. Texts from Adult Encydopedias

The texts from the adult encyclopedias and the Chevrolet manual can be characterized in terms of the discourse strategy the *constituency schema*, one of the textual structures posited in [MCKE85b]. In her work on natural language generation, McKeown studied the problems of what to say when there are many facts to choose from and how to organize a text coherently. To study this problem, she examined texts and transcripts in order to determine whether there were standard patterns of discourse structure used in naturally occurring texts. Where patterns of discourse structure could be identified for various discourse goals, these patterns could be used to guide a generation system in choosing and organizing facts to construct a text.

McKeown analyzed the texts by classifying each sentence as one of a set of rhetorical predicates. Rhetorical predicates characterize the structural purpose of sentences and have been discussed by a variety of linguists [SHEP26, GRIM75, HOBB78a]. Some examples are *constituency* (description of subparts or subtypes), *attributive* (providing detail about an entity or event) and *analogy*. McKeown found that, in the texts she studied, some combinations of predicates were more likely to occur than others, and that, for each discourse situation (such as providing a *definition*), some combination was the most frequent. McKeown encoded these standard combinations as schemas that are associated with a particular discourse situation. One of these schemas is the constituency schema which is used to describe an object

(or concept) in terms of its subparts and their properties. The constituency schema, shown in Figure 2 below, can be characterized by the four following steps:

1. [Identify the object as a member of some generic class, using the *identification* predicate][3]
2. Present the constituents of the item to be defined (subparts or sub-entities), corresponding to the *constituency* predicate
3. Present characteristic information about each constituent in turn, corresponding to the *depth-attributive* predicate
4. [Present additional information about the item to be defined, corresponding to the *attributive* predicate.]

{Identification (description of an object in terms of its superordinate)}[4]

Attributive* (associating properties with an entity) / Cause-effect*

Constituency (description of subparts or subtypes.)

{ Depth-identification / Depth-attributive

{ Particular Illustration / Evidence}

{ Comparison; Analogy} }+

{ Attributive / Explanation / Analogy }

Figure 2. The constituency schema

The descriptions from the adult encyclopedias and the car manual for mechanics followed the pattern shown in this schema: the descriptions are organized around the parts of the object. An example of such a description is show in Figure 3. This entry, describing the telephone, is taken from [COLL62].

In the first sentence, the telephone is described in terms of its subparts: the transmitter, the receiver and the housing. This proposition corresponds to the constituency predicate, the second step of the schema. The first step of the schema, identification in terms of a generic class, was omitted. Following this sentence, the two main subparts are described in turn: the transmitter, in sentences 2 through 8

---

[3] The steps included in square brackets are optional.

[4] We are using McKeown's notation: "{}" indicate optionality, "/" indicates alternatives, "+" indicates that the item may appear 1 or more times, and "*" indicate that the item may appear 0 or more times. Finally, ";" is used to indicate that the propositions could not be clearly classified as corresponding to one predicate. We changed McKeown's schema slightly by adding the *identification* predicate as an option for the first predicate of the schema.

	Text	Predicates
1)	The hand-sets introduced in 1947 consist of a receiver and a transmitter in a single housing available in black or colored plastic.	Constituency
		Depth-Attributive for the transmitter:
2)	The transmitter diaphragm is clamped rigidly at its edges	*Attributive*
3)	to improve the high frequency response.	*Cause-effect*
4)	The diaphragm is coupled to a doubly resonant system	*Attributive*
5)	-a cavity and an air chamber-	*Constituency*
6)	which broadens the response.	*Cause-effect*
7)	The carbon chamber contains carbon granules,	*Constituency*
8)	the contact resistance of which is varied by the diaphragm's vibration.	*Attributive*
		Depth-Attributive for the receiver:
9)	The receiver includes a ring-shaped magnet system around a coil and a ring-shaped armature of anadium Permendur.	*Constituency*
10)	Current in the coil makes the armature vibrate in the air gap.	*Cause-effect*
11)	An attached phenolic-impregnated fabric diaphragm,	*Attributive*
12)	shaped like a dome,	*Attributive*
13)	vibrates and sets the air in the canal of the ear in motion.	*Cause-effect*

Figure 3. Description of a telephone from an adult encyclopedia

and the receiver in sentences 9 to 13. Providing information about the subparts of an object is termed giving depth-attributive information. In this text, *depth-attributive* information includes the use of both the constituency and attributive predicates.

3.3. Texts from Junior Encyclopedias and from the Car Manual for Novices

The texts from junior encyclopedias, high school textbooks and the car manual for novices are organized in a significantly different manner. No known schema or other organizing structure consistently accounted for these descriptions. Moreover, identifying the rhetorical predicates of the sentences did not provide insights to a useful organizing structure. The structure resulting from using rhetorical predicates is one that contains mainly cause-effect relations. Although this structure is a simple schema, it is not useful for generation as it does not provide enough constraints in deciding what to include in a text. In looking for other types of organizing strategies, we discovered that the main strategy used in these descriptions is to trace through

	Text	Predicates
1)	When one speaks into the transmitter of a modern telephone, these sound waves strike against an aluminum disk 2) or diaphragm	Cause-effect renaming
3)	and cause it to vibrate back and forth	Cause-effect
4)	in just the same way the molecules of air are vibrating.	Cause-effect
5)	The center of this diaphragm is connected with the carbon button	Attributive
6)	originally invented by Thomas A. Edison.	Elaboration
7)	This is a little brass box filled with granules of carbon	Attributive
8)	composed of especially selected and treated coal.	Attributive
9)	The front and back of the button are insulated.	Attributive
10)	The talking current is passed through this box so that the electricity must find its way from granule to granule inside the box.	Cause-effect
11)	When the diaphragm moves inward under the pressure from the sound waves the carbon grains are pushed together	Cause-effect
12)	and the electricity finds an easier path.	Cause-effect
13)	Thus a strong current flows through the line.	Cause-effect
14)	When a thin portion of the sound waves comes along, the diaphragm springs back,	Cause-effect
15)	allowing the carbon particles to be more loosely packed, and	Attributive
16)	consequently less current can find its way through.	Cause-effect
17)	So a varying or undulating current is sent over the line whose vibrations exactly correspond to the vibrations caused by the speaker's voice.	Cause-effect
18)	This current then flows through the line to the coils of an electromagnet in the receiver.	Cause-effect
19)	Very near to the poles of this magnet is a thin iron disc.	Attributive
20)	When the current becomes stronger it pulls the disc toward it.	Cause-effect
21)	As a weaker current flows through the magnet, it is not strong enough to attract the disk and it springs back.	Cause-effect
22)	Thus the diaphragm in the receiver is made to vibrate in and out	Cause-effect

Figure 4. Description of a telephone from a junior encyclopedia

the process that allows the object to perform its function, that is, to mainly describe processes associated with the operation of the object. We characterize these descriptions as *process descriptions.* An example of such a description, from [BRIT63], is presented in Figure 4.

In this description, the author starts by explaining what happens when someone speaks into a telephone (sentence 1), and why a diaphragm located inside the

transmitter vibrates (3). While introducing the diaphragm as part of a causal link, the author also includes properties about the diaphragm: it is "aluminum" and "disc-shaped." Now that the diaphragm has been introduced as part of the process, it is described more fully in sentences (5) through (9) with various properties and structural relations. The description then reverts to tracing the process information, explaining how the soundwaves are transformed into varying current and describing how the telephone achieves its function of transmitting soundwaves.

As the vibration process in (3) can be decomposed into two substeps (i.e., "the diaphragm moves inward" and "the diaphragm springs back"), the author traces each one in turn (sentences (11) through (13) for the first substep, and (14) through (16) for the second). Sentence (17) summarizes these substeps, also indicating that the process trace will now continue at the top level, not the substep level. Sentence (18) continues the process trace by describing where current flows next. In the remaining text, the author explains how the varying current is changed back into soundwaves in the receiver of the telephone.

We see that the organizing principle of this text is the mechanical process description. The process description gets interrupted when descriptive information can be included concerning a subpart that was just mentioned as part of the process description. Furthermore, in this text, not only is the description made mainly through a process trace, but this process trace is given in great detail and substeps are explained if there are any.

The information contained in this group of descriptions corresponds to the causal links that connect the various processes contained in the knowledge base. To generate such a description, it is then necessary to follow these links, giving rise to a *process trace* that describes how the object functions. The algorithm used for the process trace, summarized in Figure 5, is as follows: given the chain of causal links that constitutes the functional information of the object, the first link is taken. If there is an important side effect at this point, it is mentioned. If a new part was

---

[For each object, given a chain of causal links]

(1) Follow the next causal link

(2) Mention an important side link

(3) {Give attributive information about a part just introduced}

(4) {Follow the substeps if there are any. (These substeps can be omitted for brevity.)}

(5) Go back to (1).

[This process can be repeated for each subpart of the object]

Figure 5. The process trace

introduced when the causal link was mentioned, attributive information about the part may be included. If the step just explained can be subdivided into substeps, the trace may continue at the substeps level. Finally, the next causal link is taken and the algorithm repeats. (The knowledge base contains various kinds of links between events, each kind having a different importance factor. TAILOR has an algorithm for automatically finding the main path for a given object. See [PARI 87a] for details.)

Substeps happen, for example, in the following case: "... causes the diaphragm to vibrate". This step can be divided into the two substeps: "the diaphragm moves inward" and "the diaphragm moves outward." Substeps can also arise when a complex object is made of several other complex parts. The strategy first describes how the parts work together to achieve the object's function. If a long description is desired, it is possible to step through each of the parts, describing how it achieves its own function. This is similar to the schema recursion for the constituency schema mentioned by [MCKE85b].

3.4. Tailoring Descriptions

Given that the two types of descriptions occur in texts, a system that generates device descriptions should also be able to provide the two kinds of descriptions. We have thus implemented both strategies in our generation system.

As a result of the difference in organizing structures, the type of information that is included in the descriptions is also different: in the texts from the adult encyclopedias and the manual for experts, the information included is mainly structural, while in the texts from the junior encyclopedias and the manual for novices, the information included is mainly functional (or process oriented). One of the main differences between the two audiences at which the two groups of texts were aimed is their assumed level of domain knowledge. We postulate that the writers' choice of strategy might be based on the assumed domain knowledge of the expected readers.[5] If so, the reader's level of knowledge about the domain affects the *kind* of information provided as opposed to just the *amount* of information, as was previously assumed [WALL82, SLEE82b]. This is significant as it adds a dimension (the different kinds of information available) along which a system can tailor its answers to users having different levels of domain knowledge.

In TAILOR, the constituency schema is used when a user has expertise about the domain of discourse, giving rise to a parts oriented description. Recall that we assumed that the goal of a description is to allow the user to form a mental model of the functionality of the object. Research in psychology indicates that expert users have more knowledge not only about individual components but also about the causal models involved and the interconnections among parts [LANC87, CHI81]. Expert users, then, are likely to have functional knowledge about the domain and to know how parts might interact with each other. As they can use this knowledge

[5] No psychological experiments have been conducted to confirm this hypothesis. It is based on observations about what happens in natural texts.

when reading the description,[6] they should be able to pull all the parts provided in the description together in order to "understand" the description as a whole. That is, they should be able to figure out how the parts fit together to form an object capable of performing a function. Since the reader is able to construct a mental model, it was unnecessary to include the process information in the description. Actually, assuming that the reader will be able to infer the processes involved, providing such useless information would contradict the principles of cooperative behavior [GRIC75].

On the other hand, a user who does not have enough knowledge to infer the processes linking the parts would be unlikely to understand a mostly structural description of an object and to be able to construct a functional mental model of the object from such a description. For this sort of naive user, if the description is to be informative and understandable, it must describe how the parts perform the function of the object. The description must therefore include process information. Previous research in reading comprehension strengthens our belief that a user who does not have knowledge about the functions of the various parts will not be able to make sense of a description centered around parts. Wilson and Anderson [WILS86] demonstrate the importance of prior knowledge in comprehending new texts, and show how readers can fail to understand a text mainly because the text contains implicit knowledge that the readers do not have. We thus propose to use the process trace when providing a description to a naive user.

To summarize, we suggest that the user's domain knowledge affects the content of a description with respect to the kind of information included, and not just to the level of detail, and postulate that the choice of strategy might be based on the assumed level of expertise of the reader/user. Namely, the process trace can be used when the expected readers are relatively naive about the domain of discourse, while the constituency schema can be used when the expected readers have expertise about the domain. We now show how a user's level of expertise can be incorporated in a generation system and how it can guide the system in choosing a discourse strategy.

4. Mixing the Strategies

The two strategies presented account for the main differences found between the adult and junior encyclopedia entries and we proposed to use them to describe objects to naive or expert users. Users are not necessarily either naive or expert in a domain, however. They may have *local expertise*, knowing about some objects in the domain and not others [PARI84]. Such users would not be considered naive users, but, as there are many objects they do not know in the domain, they would not be considered expert users either. The user models for such users would indicate for which objects of the knowledge base they have local expertise. We believe that, to describe objects to users with intermediate levels of expertise, a combination of

[6] Research in reading comprehension indicates that readers indeed use their previous knowledge in order to understand new texts [ANDR77, WILS86].

the two strategies presented for naive and expert users is appropriate. Based on the user model, a generation program can decide which strategy to use for which object.

As an example, suppose we are providing a description of an elevator to a user who knows the function of a motor and how this function is achieved, but not how the elevator itself works. In describing the elevator to this user it is necessary to first describe how the parts of the elevator work together, using the process trace strategy, since the user model would indicate local expertise about the "motor" only. This local expertise is too narrow to allow the user to understand how all the parts of the elevator work together to perform their required function. A process explanation is thus necessary. In describing the individual parts in turn, it should not be necessary to fully explain what the motor does, as the user already knows about it. The constituency schema strategy can thus be used to describe the motor. For the other parts, however, the process strategy is still appropriate in order to explain their mechanisms.

Such combinations were actually also found in naturally occurring texts. Figure 6 presents an example of a text that uses a combination of the constituency schema and the process trace to generate a description aimed at users with intermediate levels of domain knowledge. This text is taken from the Encyclopedia of Chemical Technology [MARH87]. The description starts with the constituency schema strategy but ends with a process trace: the "IR (Infra-Red) spectrometer" is first described in terms of its parts; each part is then described in turn (depth-attributive); finally, the authors revert to a process trace to describe the "thermocouple detector," assuming it is unknown to the reader. To fully understand this text, the reader must already know (or be able to infer information) about the IR spectrometer's purpose, the "IR radiation" and the "monochromator". In Figure 6, the text corresponding to the process trace is in italics.

---

(1) The IR spectrometer consists of three essential features: a source of IR radiation, a monochromator and a detector. (2) The primary sources of IR radiation are the Globar and Nernst glower. (3) The Globar is a silicon carbide rod heated to 1200 degrees C. (4) The Nernst glower is a rod containing a mixture of yttrium, zirconium, and erbium oxides that is heated electrically to 1500 degrees C. (5) Earlier IR spectrometers contained prism monochromators but today gratings are used almost exclusively. (6) Most detectors in modern spectrometers operate on the thermocouple principle. *(7) Two dissimilar metal wires are connected to form a junction. (8) Incident radiation causes a temperature rise at the junction and the difference in the temperature between head and tail causes a flow of current in the wires which is proportional to the intensity of the radiation.*

Text Decomposition

1. Constituency
2-4. Depth identification for the IR radiation
5,6. Depth identification for the monochromators
7. Depth identification for the thermocouple spectrometer
8. Process trace for the thermocouple principle.

Figure 6. Naturally occuring text combining the two strategies

4.1. Generating a Description Based on the User Model

Before starting to generate a text, TAILOR must decide whether to use the constituency schema or the process trace for the overall framework of the text. While constructing the description, TAILOR can switch to the other strategy, thus combining the two strategies in one text. This is done for users with intermediate levels of expertise. There is therefore a need to specify 1) how to choose an initial strategy and 2) how to combine the strategies. In TAILOR, the decision on which strategy to use at any point is based on the user model.

4.1.1. Choosing a Strategy for the Overall Structure of the Description

To choose the initial strategy, TAILOR looks into the knowledge base to see whether it is possible to describe the object using either strategy. In particular, it checks whether there is process information associated with the object. If the object has no such information in its frame, the constituency schema is chosen by default. This will happen in two cases: first, when the object's function cannot be broken down into a sequence of events (for example, a "cover" has no mechanism associated with it), and second, when the object's function and the way it is performed is identical to that of its superordinate in the generalization hierarchy (for example, the function of the *pulse telephone* is the same as that of a *telephone*). Process information in this case is stored in the frame of the superordinate. The pulse telephone frame contains the part that differs from the telephone, namely the pulse dialer. Although the constituency schema will be chosen for the initial strategy of the description of a pulse telephone, it will be possible to switch to the process trace to explain the function of its superordinate. Similarly, it will be possible to switch to the process trace to explain the function of the "new" (or different) part, i.e., the pulse dialer.

If there was process information (or a mechanism) associated with the object, it is possible to describe the object with either strategy. TAILOR examines the user model to check the user's local expertise. If the user model does not indicate any local expertise about either the object to be described or its superordinate in the generalization hierarchy, the process trace is chosen. Otherwise, the constituency schema is chosen. (If local expertise is indicated about the superordinate of the object, the user will most likely be able to infer the processes for the object given his or her knowledge about the superordinate; therefore, the constituency schema is still adequate for the description.)

The constituency schema is also chosen when the user model indicates local expertise about all or most of the functionally important subparts of the object to be described.

A distinction is made between parts that play a significant role in the mechanical process of a device and other parts, because knowledge about the former will help the user to comprehend the mechanism of the device. For example, the *housing* of a telephone does not play an important role in allowing the telephone to transmit soundwaves. Knowing about the *housing* or about a *cover*, the housing superordinate in the generalization hierarchy, does not help a user understand how the telephone works. Understanding how a microphone and a loudspeaker work,

on the other hand, is useful since the transmitter of a telephone is essentially a microphone, and the receiver a loudspeaker.

If the user already knows about most of the functionally important subparts of a device, he or she is likely to be able to infer how the parts fit together functionally, so it is not necessary to include a process description for the object. The overall description of a text in that case will be based on the constituency schema. "Most" is represented in TAILOR as a parameter that can be set to any value for experimental purposes. Its particular value in the current implementation is half the number of functionally important parts.[7] If the user does not have local expertise about most of the subparts of the device, the process trace is chosen, and a process explanation of how the subparts function together is provided.

To summarize, the constituency schema is chosen as the overall structure of the text in the following cases:

1. The object has no mechanism associated with its frame in the knowledge base.
2. The user model indicates local expertise about either the object or its superordinate in the generalization hierarchy.
3. The user model indicates that the user knows about most of the functionally important parts of the object.

The process trace is chosen as the initial strategy in all other cases. The basic decision algorithm is shown in Figure 7.

---

1) Is there a mechanism associated with the object to be described?

No: Use the constituency schema
Yes: 2) Is the object to be described (or its superordinate) in the user model?
(i.e., does the user have local expertise about this object or its superordinate?)

Yes: Use the constituency schema
No: 3) Collect all the functional parts of the object.
If the user has local expertise about most of these parts,
use the constituency schema
Otherwise use the process trace.

Figure 7. The decision algorithm for the initial strategy

Whether the constituency schema or the process trace is chosen as the overall strategy for the description, a mixture of structural and process information can still be provided depending on the user model. For example, if the constituency schema

[7] We do not claim that this is the optimal value for the threshold indicating at which point the constituency schema should be used. This threshold cannot be set with certitude without further experimentation. Also note that, while these heuristics allow the system to generate reasonable descriptions given the user's domain knowledge, there are no clear "best" descriptions.

is chosen initially, the process trace can be used for the unknown parts, providing the user with a process explanation for these parts. Likewise, if the process trace is taken at first, structural information about the parts already known to the user is still included. "Most" only determines the initial strategy (i.e., the overall structure of the text). It does not determine whether a mixture of structural and functional information will be included.

4.1.2. Decision Points Within the Strategies

To generate a description aimed at users with intermediate levels of expertise, the two strategies must be combined. We must specify *when* to switch from one strategy to the other and *under which conditions* to do so. The decision of which strategy to use at any point is based on information about the user's domain knowledge contained in the user model. Notice that it would be hard without a thorough psychological study to specify the exact conditions necessary to choose one strategy over the other and to switch from one strategy to the other. However, we have identified some heuristics that determine when to mix the strategies. (Should data from psychological experiments later become available, we could use the results as our heuristics to generate appropriate texts given the user's domain knowledge.)

To mix the strategies, we must specify *when* it might be possible to switch. Whenever an object is introduced and needs to be described, the system must decide whether to provide chiefly structural or functional information. Thus, each time an object needs to be described, a strategy must be chosen and a strategy switch might occur.

This gives us some clear decision points in the strategies:

- Within the constituency schema:
  - After the *identification* predicate: once the parent of an object in the generalization hierarchy has been introduced, a process trace can be provided for the superordinate.
  - After the *constituency* predicate: after mentioning the parts of an object, the constituency schema dictates filling the depth identification predicate for each subpart. Instead, we could provide a functional description of one or more of the parts. This has been done in the Encyclopedia of Chemical Technology text presented in Figure 6, for example.
- Within the process trace:
  - When a part is introduced while traversing the causal links, the process strategy dictates to include attributes of this part to describe it. Here, we could also choose to describe the part more fully with the constituency schema.
  - When the subparts have to be described, the constituency schema can be used to provide structural information about them instead of including functional information.

Figures 8 and 9 summarize the two strategies in their simplest form with the decision points.

Like the initial decision for a strategy, the decision to *switch* strategy is mainly based on the user model. If the initial strategy is the constituency schema, TAILOR

Constituency Schema (with decision points)

Identification (introduction of the superordinate)
If there is no local expertise for the superordinate
do a Process Trace (for the superordinate) before proceeding.
Constituency (description of the subparts)
For each part, do:
If there is local expertise about this part (or its superordinate),
do **Depth-identification**
Otherwise do a Process Trace (for the part)
Attributive

Figure 8. The constituency schema strategy and its decision points

Process Trace (with decision points)

Next causal link
Properties of a part mentioned during the process trace
If a fuller description of the part is desired,
do Constituency Schema (for the part)
Substeps
Back to next causal link

Repeat for each of the subparts:
If there is local expertise about this part (or its superordinate),
do Constituency Schema
Otherwise do a Process Trace

Figure 9. The process trace strategy and its decision points

considers switching to the process trace for the superordinate and for the subparts of the device being described. The test for switching is largely the same as the initial test. TAILOR first checks whether the object under consideration (either the superordinate or each of the subparts) has some process information in its frame, or in its superordinate's frame. If so, TAILOR examines the user model to see whether the object or its superordinate is included in the user's local expertise. Switching is considered only if it is not included.

Two more conditions are tested before switching occurs. If the process explanation of the object includes *basic concepts* of which the user lacks knowledge, the process explanation would involve steps the user is not likely to comprehend. As a result the process explanation will not be understandable. For example, if the process explanation of a part involves the concepts of electricity and magnetism, it will not be informative if the user does not understand them. In that case, switching does not occur.

TAILOR keeps a record that switching was not allowed because of lack of knowledge about basic concepts, and, after the initial description is given, the user is given the option to ask for the process description for the part, even though it contains concepts he or she might not understand.

The last condition that needs to be satisfied before the switch can occur is a test on the length of the text constructed so far and, in case of subparts, the number of parts that need to be described using the process trace. This is done to avoid generating very long texts. A variable, which can be arbitrarily set to any value, sets a threshold beyond which the process trace will not be included at this point.[8] TAILOR remembers that the process explanations for some parts were not included because of length, and, after the text corresponding to the initial strategy is produced, TAILOR asks the user whether it should provide these explanations. When all the conditions are satisfied, TAILOR decides to switch to the process trace; otherwise, it continues with the constituency schema.

If the process trace is chosen for the overall structure of the text, a test similar to the one described above is performed when an object has been introduced as part of the process explanation. In this case, however, switching is considered only if the user model indicates local expertise about the part just introduced. As above, a test on the length of the text generated so far and the number of parts to be described using the constituency schema is also done to determine whether the switch should occur when the part is mentioned as part of the process or, at the user's request, after the initial text is produced. Again, when all the conditions are satisfied, TAILOR switches to the constituency schema for the part.

The user model is also examined to decide whether substeps should be traced: if the substeps involve any basic underlying concepts and the user model indicates that the user knows these concepts, the substeps are traced. Otherwise, the program does not trace through the substeps, as they might confuse the user.

Figures 10 and 11 show examples of texts generated by TAILOR by combining the two strategies. In Figure 10, based on the user model that indicates local expertise about *loudspeaker*, TAILOR chooses the constituency schema. The *telephone* is first identified with its purpose, and its subparts are introduced. Then, the constituency schema directs TAILOR to provide attributive information about each of the subparts. At this point, it is also possible to switch to the process trace for some subparts. TAILOR examines each subpart to decide whether there is a need to switch strategy for any of them. Two of the subparts have a mechanism associated with them (or with their parent in the generalization hierarchy): the transmitter and the receiver. It is thus possible to describe either of these parts with the process trace. Because the user model indicates local expertise about the *loudspeaker*, the receiver's superordinate, TAILOR decides to use attributive information and not a process explanation to describe this part. As no local expertise about the transmitter or the microphone is indicated in the user model, TAILOR chooses to provide process information for the transmitter, switching momentarily to the process trace. After the process explanation for the transmitter is completed,

[8] Note that this test can also be used to test schema recursion while going through the constituency schema, when the user model indicates local expertise about a part and switching is not allowed.

Telephone

User Model: Local Expertise — *Loudspeaker*
Basic Concepts — *electricity*

The telephone has two main functional parts: the transmitter (an instance of a microphone) and a receiver (an instance of a loudspeaker). Because the user knows one of the two parts of the telephone, the system decides on the constituency schema strategy at first. However, before providing structural information about each subpart, the system consults the user model and decides to switch strategy to describe the process of the transmitter since the user has no local expertise about it.

TAILOR output:

The telephone is a device that transmits soundwaves.	Identification
The telephone has a housing that has various shapes and various colors, a transmitter that changes soundwaves into current, a curly-shaped cord, a line, a receiver to change current into soundwaves and a dialing mechanism.	Constituency
The transmitter is a microphone with a small diaphragm. A person speaking into the microphone causes the soundwaves to hit the diaphragm of the microphone. The soundwaves hitting the diaphragm causes the diaphragm to vibrate. The vibration of the diaphragm causes the current to vary. The current varies, like the intensity varies.	Switch to process trace for the transmitter
The receiver is a loudspeaker with a small aluminum diaphragm. The housing contains the transmitter and it contains the receiver. The housing is connected to the dialing mechanism by the cord. The line connects the dialing mechanism to the wall.	Attributive information for the other subparts

Figure 10. Combining the strategies: using the constituency schema as the overall structure of the text and switching to the process trace for one part

TAILOR returns to the constituency schema, and attributive information is provided about each of the other subparts. The process trace is shown in italics in the figure.

Figure 11 provides the description of a radio station. In this example, the process trace is chosen for the initial description of the object, and switching to the constituency schema occurs for one of the parts. The user model indicates local expertise about radio transmitters. The process trace is chosen for the overall structure of the text, because the user does not know most of the parts of the radio station. The description starts by introducing the object with its

Radio Station

User Model: Local expertise — *radio transmitter*
Basic Concepts? — *electricity*

A radio station has a few functionally important parts, only one of which is known to the user. The process trace is thus chosen as the overall structure of the text. TAILOR switches to the constituency schema for the radio transmitter as the user model indicates local expertise about it.

TAILOR output:

A radio-station is a device that broadcasts a signal when a human speaks into a microphone. Because the human speaks into the microphone, a current is produced. Then, the current flows through the wire into the radio-transmitter.	Process Trace
The radio-transmitter has an oscillator that produces a varying current when a battery produces another current, an amplifier that produces a strong varying audio-frequency signal from a varying audio-frequency signal, a mixer, and an amplifier to produce a strong radio-frequency signal from a radio-frequency signal. An amplifier has a vacuum-tube to produce a strong current from a power source and a metal grid. The oscillator has a high-resistance resistor, a battery, a capacitor, a low-resistance resistor and a transistor.	Switch to constituency schema for the radio transmitter
The current flowing through the wire into the radio-transmitter causes the strong varying radio-frequency signal to be produced. Then, the strong varying radio-frequency signal flows into the antenna of the radio-station. The antenna is long and made of metal. Because the strong radio-frequency signal flows into the antenna, the strong radio-frequency signal is broadcasted.	Continue the process trace

Figure 11. Starting with the process trace and switching to the constituency schema for one part

purpose and following the main sequence of events that occur when the object performs its function. When the radio transmitter is introduced as part of the process explanation, TAILOR switches to the constituency schema to describe it, as the user model shows local expertise about radio transmitters. After the radio transmitter is described with its subparts, the process explanation is resumed.

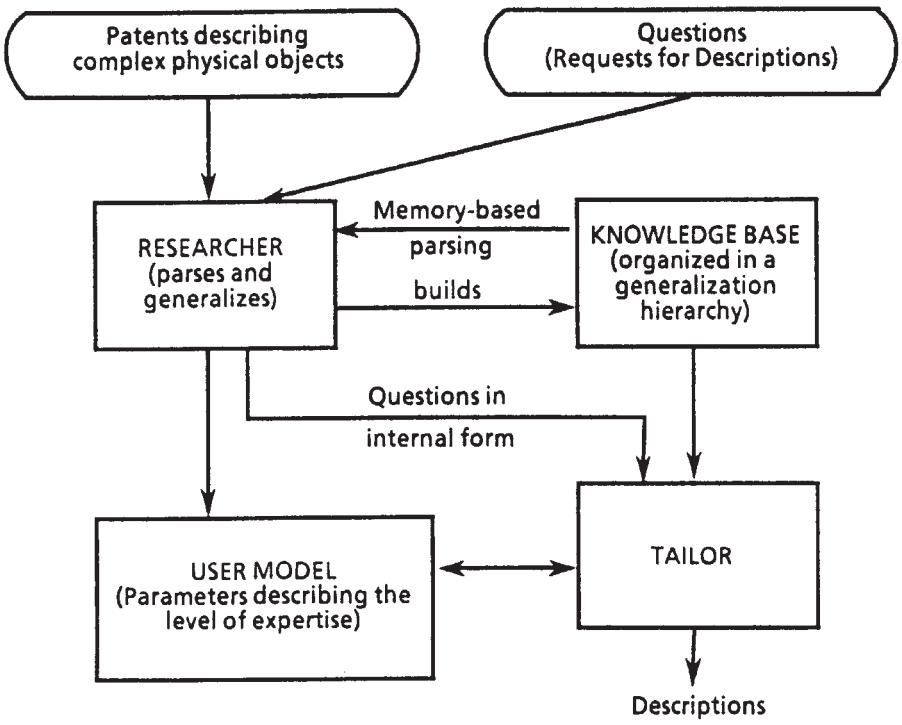


Figure 12. RESEARCHER and the TAILOR system

5. TAILOR

5.1. Overview of the System

We have implemented the discourse strategies presented above in TAILOR, a program that generates descriptions tailored to a user's level of expertise. The discourse strategies guide the program to choose the appropriate information from the knowledge base, under the constraint of the user model. TAILOR generates descriptions aimed at users anywhere along the knowledge spectrum. TAILOR uses the knowledge base built by RESEARCHER, as depicted in Figure 12, and looks at the information contained in the user model to decide on the strategy to employ. At this point, TAILOR is not really an interactive system and there is little feedback from discourse. In TAILOR, the user issues a request for a description and the description is generated. The object that was just described is added into the user model, as TAILOR assumes the user now has local expertise about it. TAILOR does not keep a record of all the questions asked by the user and all the answers provided. No parsing of the questions is done (the input consists of a request for a description). In the ideal system, RESEARCHER would parse the question using the same parser as that used in reading patent abstracts, produce the request and hand it to TAILOR. The question (along with other factors) could also be used to determine the level of expertise. The user model includes the parameters that TAILOR uses to constrain its decision process during generation. The user model is not determined by the program but given as input.

RESEARCHER's knowledge base contains detailed descriptions of complex devices, including both structural and functional information about the objects. We use a frame-based knowledge representation [WASS83, WASS85] in which the basic frames represent objects. These structures are organized in a generalization hierarchy. In addition to the generalization links, or *instance-of* links, there exist two additional kinds of links joining entities: *part-of* links, which indicate that an entity is a part in a larger structure, and *relations*, which convey information about spatial or functional relationships. Functional relations correspond to the various events (or processes) that occur. Finally, there are links between relations, that is, links between events. These links include *cause-effect* relations, *temporal* relations (such as "X happens at the same time as Y") and *analogical* relations (such as "X corresponds to Y").

A top-level diagram of TAILOR is shown in Figure 13. Input to TAILOR includes: (1) a request for the definition of an object, and (2) the level of expertise of the user, that is, the set of parameters that describe the user's level of knowledge about the domain, including:

- A list of basic underlying concepts the user knows about.
- The specific objects the user knows in the domain, that is, the user's local expertise (a list of pointers into the knowledge base).

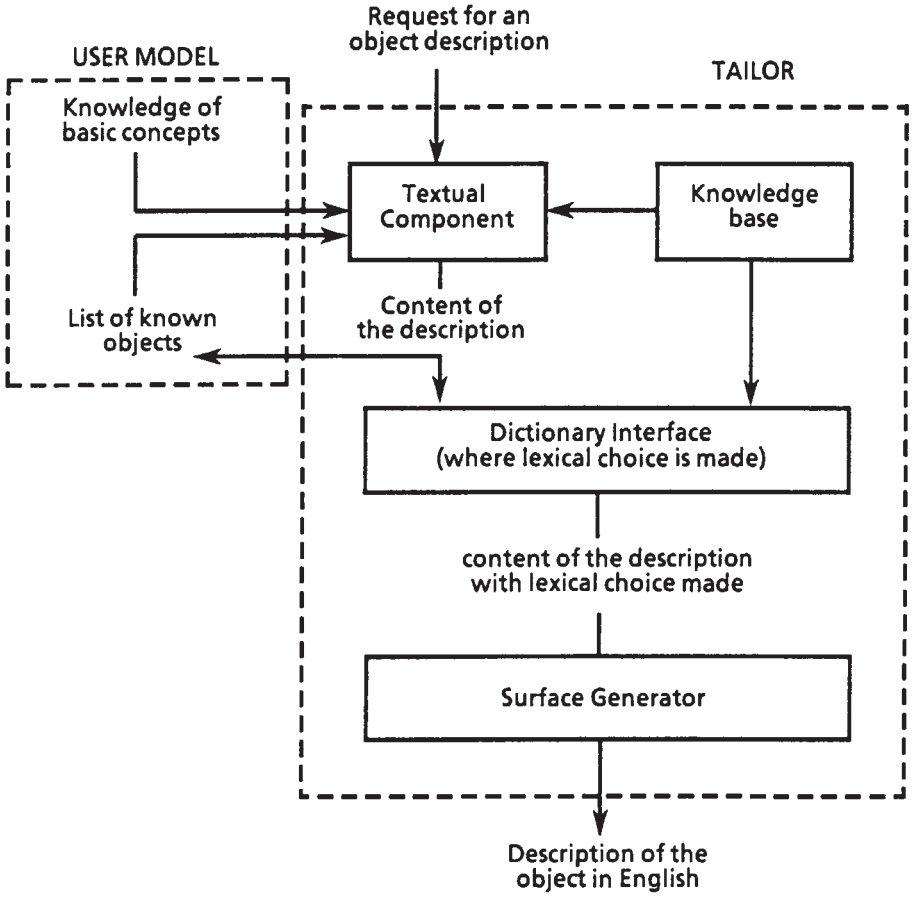


Figure 13. The TAILOR system

The *textual component* of the system decides what to include in the description. It looks into the knowledge base and, based on the user model and the discourse strategies, chooses appropriate facts. The output of the textual component is a conceptual representation of the content of the description. This representation is passed through an *interface*, which makes syntactic and lexical choices for the various concepts included in the description. The interface uses the focus of a proposition and the past discourse to guide its decision process. (However, as our emphasis in this work is on the *content* of a description as opposed to its *phrasing*, we have not studied in depth the complexity and subtleties of lexical choice.) Finally, a *surface generator* constructs English sentences. The surface generator is based on the one used by [MCKE85b] in the TEXT system. This generator unifies the input with a functional grammar [KAYM79] to produce English sentences. We have extended and improved the performance of this program, and augmented the functional grammar it uses [PARI85b, MCKE87].

5.2. Implementation of the Strategies

The constituency schema and the process trace strategies are implemented using an augmented transition network (ATN) [WOOD73]. The arcs joining the various nodes in the network specify what information is to be retrieved from the knowledge base, under what conditions (the arcs contain a test), and which node to go to next. Figures 14 and 16 present the nets used for the strategies. (In these figures, the schemas do not include the links allowing the program to switch from one strategy to the other.)

In the constituency schema, shown in Figure 14, the arcs correspond to the predicates from the schema. These predicates define the type of information to be taken from the database.[9] They are:

- **identification**, presenting the more general concept of which the present object is an instance.
- **constituency**, giving the components of an entity, if there are any.
- **attributive**, providing different attributes of an object (such as its shape or material).
- **cause-effect**, providing some causal relations between entities or relations.

The process of filling the ATN for the constituency schema strategy to describe a *microphone* is shown in Figure 15. The constituency schema would be chosen by TAILOR if the user model exhibits local expertise about the microphone (or if the user is classified as an expert). In this example, the identification predicate is first applied to the microphone. The identification predicate provides the superordinate of the object together with the function of the object. The constituency predicate is then applied, providing the subparts of the microphone, together with their properties or purposes. Finally, the depth-identification predicate is applied to each subpart, the *doubly-resonant system* and the *diaphragm*. The English output shown in the figure (and in the other figures) is the actual output from TAILOR.

[9] More details can be found in [PARI87a].

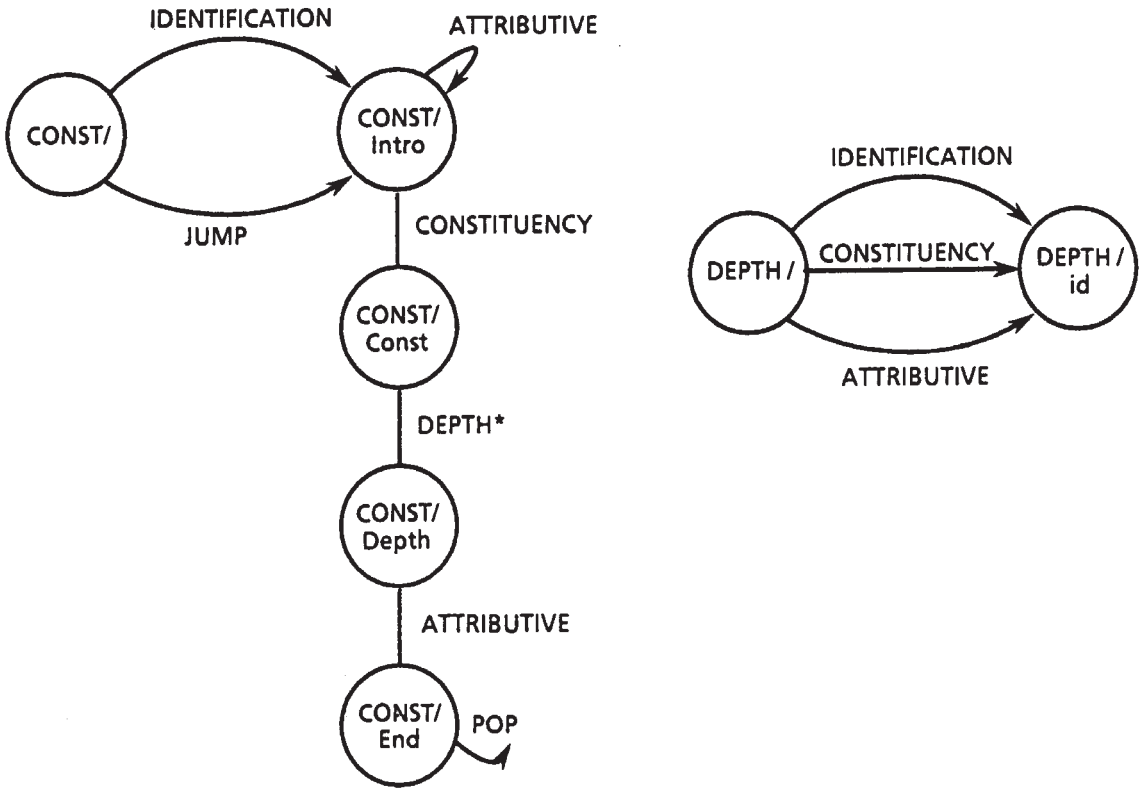


Figure 14. The constituency schema

The network for the process trace is shown in Figure 16.[10] An example of following the process trace strategy is presented in Figure 17. In this network, the arcs dictate how to trace the knowledge base to form an answer, but they are not linguistic predicates as in the network corresponding to the constituency schema. They mainly dictate how to follow the causal links in the knowledge base. (Details about the process trace can be found in [PARI87a, PARI87b]). The arcs are:

- **Next-main-link**: this arc dictates to follow the next link on the *main path*. The main path is the sequence of events that is performed in order for an object to achieve its function. (TAILOR finds the main path and sidelinks automatically.)
- **Side-link?**: *A side-link* is a link that is not part of the main path, but that comes off an event on the main path. This arc tests to see whether there is a side link caused by an event at this point. The decision to mention the side link is based on the importance of that link.[11]
- **Attributive**: This arc is similar to the attributive predicate in the constituency schema. If information about a part just introduced is available in the knowledge base, this arc will be taken.

[10] The network shown here is actually slightly simplified for clarity. The full network can be found in [PARI87a].

[11] We have mentioned that there are different kinds of links between events. The types of links are ranked in order of importance, and the most important ones are mentioned.

```
; Stepping through the Constituency Schema to describe
; a MICROPHONE

Applying the predicates to MICROPHONE:

  Identification predicate: DEVICE;
                              (used-for: change soundwaves into current)
  Constituency predicate:  DIAPHRAGM
                              (shape disc, material aluminum),
                           DOUBLY-RESONANT-SYSTEM
                              (used-for: broaden response)

  Depth-constituency for DOUBLY-RESONANT-SYSTEM:

                           CARBON-CHAMBER, AIR-CHAMBER

  Depth-attributive for DIAPHRAGM:

                           (edges clamped)
```

TAILOR output:

The microphone changes soundwaves into current. It has a disc-shaped aluminum diaphragm and a doubly-resonant system to broaden the response. The system has a carbon chamber and an air chamber. The diaphragm is clamped at its edges.

Figure 15. Stepping through the constituency schema

- **Substeps?**: If an event consists of several substeps, the substeps are traced first. To traverse the *substeps*, the subroutine substep is called for each substep. This subroutine is very similar to the main graph, but does not allow for a further decomposition of events.
  In Figure 17, the link "the diaphragm vibrates" can be divided into substeps, namely "the diaphragm goes forward" and "the diaphragm goes backward". The process trace can be continued at the substep level. Then, once the substeps have been traced through, the trace returns to the top-level description. (Only one substep is fully shown in Figure 17 for brevity. See [PARI85a] for details.) We can also choose to not follow the substeps in order to generate a shorter description. This factor is incorporated into the arc test.

By representing the two strategies in this formalism, we immediately obtain the control structure necessary to switch strategies, since it is possible to jump from a node in one part of the network to a node in a different network. The decision points are thus marked as special tests on the arcs joining the nodes.

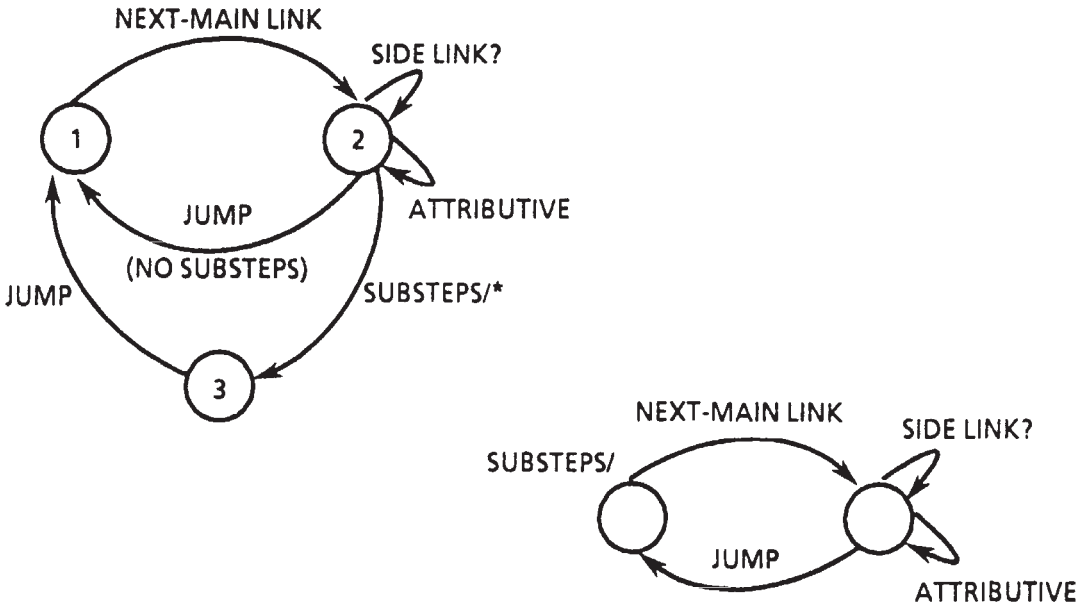


Figure 16. The process trace

As a general test for deciding which strategy to use to describe a part, we look into the user model to check if a superordinate of the part (or the part itself) is known to the user. If the part is known, the constituency schema is used. Otherwise, the process trace is chosen. This test is invoked before beginning generation and at any point in the schema where it is possible to switch strategies. The process of stepping through the ATN and switching strategy is shown in Figure 18. (The corresponding generated text was shown in Figure 10.) Note that to avoid generating very long texts, substeps are omitted when the process trace is chosen to describe a subpart.[12]

6. Further Work and Related Issues

6.1. Recursive Use of the Strategies

TAILOR can also use the two strategies recursively, although this was not one of our main concerns. TAILOR can currently call the constituency schema recursively for each subpart of the top level object being described, but not for the components of these subparts. Similarly, the process is only traced one level down, that is, only the top-level relations are broken into substeps. This kind of recursion could occur as deeply as the knowledge base allows for. We think that such a level of detail is appropriate only if the user asks for a longer description. We have implemented a mechanism that allows the user to request such a description, either initially or after

[12] After generating the description, TAILOR asks the user whether it should now provide a process explanation at the substeps level.

; The program traces the process information for a naive user

Causal link (1): {M-CAUSES} *relates the two relations:*
[ONE] P-SPEAKS-INTO [MICROPHONE]
[SOUNDWAVES] P-HITS [DIAPHRAGM]

Attributive Information about DIAPHRAGM: [material: aluminum]
[shape: disc]

Causal link (2): {M-CAUSES} relates the two relations:
[SOUNDWAVES] P-HITS [DIAPHRAGM]
[DIAPHRAGM] P-VIBRATES

; Substeps: "Soundwaves increasing" and "soundwaves decreasing"

Substep1 {M-CAUSES} *relates the two relations:*
[SOUNDWAVES] P-INCREASES [DIAPHRAGM]
[DIAPHRAGM] P-SPRING [Direction: forward]

{M-CAUSES} *relates the two relations:*
[DIAPHRAGM] P-SPRING [Direction: forward]
[GRANULES] P-COMPRESS

{M-CAUSES} *relates the two relations:*
[GRANULES] P-COMPRESSES
[RESISTANCE] P-DECREASES

{M-CAUSES} *relates the two relations:*
[RESISTANCE] P-DECREASES
[CURRENT] P-INCREASE

Substep2 {M-CAUSES} *relates the two relations:*
[SOUNDWAVES] P-DECREASES [DIAPHRAGM]
[DIAPHRAGM] P-SPRING [Direction: backward]

...

; [The remainder of the trace is omitted here for brevity.]

TAILOR output:

A person speaking into the microphone causes the soudwaves to hit the diaphragm of the microphone. The diaphragm is aluminum and disc-shaped. The soundwaves hitting the diaphragm cause the diaphragm to vibrate. The soundwave_intensity increasing causes the diaphragm to spring forward. Because the diaphragm springs forward the granules compress. This causes the resistance of the granules to decrease. Because the resistance decreases the current increases. The soundwave_intensity decreasing causes the diaphragm to spring backward. This backward causes the granules to decompress. Because the granules decompress the resistance increases. The resistance increasing causes the current to decrease. The vibration of the diaphragm causes the current to vary. The current varies like the intensity varies.

Figure 17. Tracing the process information for a naive user

User Model: Local Expertise: Loudspeaker
; Stepping through the Constituency Schema to describe a TELEPHONE.
; Switching to the Process Trace to describe the TRANSMITTER.

Applying the predicates to TELEPHONE:
  Identification predicate: DEVICE;
    (used-for: change soundwaves into soundwaves)
  Constituency predicate: DIALING_MECHANISM
  TRANSMITTER
    (used-for: change soundwaves into current)
  LINE
  CORD
  RECEIVER
    (used-for: change current into soundwaves)
  HOUSING
    (properties: color: various, shape: various)

Need to switch to process trace for the TRANSMITTER.
Introduction:
  Identification predicate: MICROPHONE

Causal link (1):	{M-CAUSES}	*relates the two relations:* [ONE] P-SPEAKS-INTO [TRANSMITTER] [SOUNDWAVES] P-HITS [DIAPHRAGM]
Causal link (2):	{M-CAUSES}	*relates the two relations:* [SOUNDWAVES] P-HITS [DIAPHRAGM] [DIAPHRAGM] P-VIBRATES

[*Substeps omitted*]

Causal link (3):	{M-CAUSES}	*relates the two relations:* [DIAPHRAGM] P-VIBRATES [CURRENT] P-VARIES
Side Link (4):	{M-EQUIVALENT-TO}	*relates the two relations:* [CURRENT] P-VARIES [SOUNDWAVE-INTENSITY] P-VARIES

Returning to the Constituency Schema:
Applying the predicates to RECEIVER:
  Identification: RECEIVER is a LOUDSPEAKER
    (difference: small aluminum diaphragm)

Applying the predicates to HOUSING:
  Attributive: HOUSING r-contains TRANSMITTER
  HOUSING r-contains RECEIVER
  HOUSING r-connected-to DIALING_MECHANISM by CORD

Applying the predicates to LINE:
  Attributive: DIALING_MECHANISM r-connected-to WALL by LINE

Figure 18. Switching strategy

a description has already been provided by the system and the user wants additional information. We could extend this work to have TAILOR decide whether a deeper recursion is necessary for a particular user. As we already mentioned, we think that the user's domain knowledge, in particular knowledge of basic concepts, plays a role in determining the depth needed: at some point, one cannot subdivide the processes any further if the user does not know basic underlying concepts, as the process explanation will cease to be understandable. For example, explaining how a disk drive works in terms of current and magnetism will not be informative if the user does not understand these concepts. The process explanation, in such a case, should be kept at a higher level. The depth of the knowledge base itself may also affect the depth of the description, as parts or processes may be considered too minor to be mentioned beyond a certain threshold.

6.2. Determining the Level of Expertise

In this work, we have not addressed the issue of determining the level of expertise of the user. This is obviously an important question that needs to be studied. We believe that it is possible to infer the user's level of expertise. Some relevant factors are:

- **The user type.** Some classes of users may be likely to be naive while others may be likely to be expert. Identifying a user as part of these classes can give insight into how much that user might know, and provide a starting point. This is similar to using stereotypes to model the user [RICH79b, CHIN86].
- **The question type.** The kind of question asked ("What is a tape recoder?" as opposed to "Does this disk drive have three bearings?") can give further information about how much the user knows.
- **The depth in the knowledge base** (both in the components tree and the generalization hierarchy) of the object of the question ("Describe a disc drive" as opposed to "Describe the track assembly in a disk drive") [PARI84].
- **The previous discourse.** If a user asks the same question twice, it is indicative that the answer was not understood, perhaps because it assumed knowledge the user did not have.
- **Inference rules.** Inferences rules relating objects in the knowledge base can be used to extend an existing user model as in [SLEE85]. For example, an inference rule might indicate that expertise about microphones also implies expertise about loudspeakers, as their mechanisms are similar. Knowing that the user is familiar with some items of the knowledge base might thus allow a system to infer expertise about other items [KASS87a, CHIN*].
- **Asking the user questions.** It is possible to ask the user a few questions about himself to get a starting point for the user model. This is the approach taken in both GRUNDY [RICH79b] and UMFE [SLEE85] for example.

It would be necessary to study how the user's domain knowledge can be inferred from these factors and how these factors affect each other. This seems to be a difficult problem.

Finally, our user model at this point is coarse grained, in that it contains a list of objects that the user knows and whether he or she understands the basic underlying concepts. A more detailed model might include exactly which facts the user knows about objects, and specifically which basic concepts are understood. We have shown that a system benefits from a user model that indicates a user's knowledge about the domain, even when the model is not a detailed one. While we feel that a detailed user model would be much harder to obtain, it would be interesting to see whether such a model would allow a system to provide more appropriate answers.

6.3. Incorporating the User's Goal

It is clear that the goal of the user also influences the content of an answer, or of a description. It would be interesting to examine how the goal and the level of expertise combined can help a program in choosing appropriate facts from the knowledge base. In this research, we found that the level of expertise affects the kind of information to include in a description. This is also clearly true for a user's goal: the information provided in the description should help the user in pursuing his or her goal. Thus, depending on a user's goal, the information included would vary. The interaction of these two factors can become quite complex. However, until both factors are fully understood individually, we feel that it will be very hard to determine exactly how they interact.

6.4. Feasibility and Extensibility of This Approach

In this work, we make the assumption that a system that tailors its answer to its users will be most useful. This is only true provided that this tailoring does not hinder the system's performance or increase its complexity significantly. We argue that tailoring a description to a user's level of expertise using the method described above will not add to the cost of generation and yet might provide better answers. Whether a generation system tailors its answers to users or not, it needs a control structure that will guide the decision process both in determining facts to take from the knowledge base and in organizing them, lest the resulting text be incoherent. A discourse strategy is one way to guide its decision process that has been used successfully in previous generation systems. The two discourse strategies used by TAILOR are of comparable complexities to those used in other systems and, in fact, one is identical. Other systems employing discourse strategies must decide which strategy to use to produce a text. The decision is usually based on the question type and the structure and content of the knowledge base. In TAILOR, the user model plays a role, but this does not add any cost to the decision process. Furthermore, to combine the strategies does not add cost either, especially because of the formalism used to represent the strategies. By representing both strategies with an ATN, the control structure necessary to combine the strategies is readily available. It is not more costly to jump from one node in one network to another node in the same network than to go from one network to another.

While this work was done only with respect to generating descriptions of complex devices, we think this approach will be useful in any information seeking

environment to which users with different background and knowledge levels have access. Such an environment could be a large knowledge base of facts (such as an encyclopedia), a help system, an expert system which needs to communicate with specialists, students and knowledge engineers, or simply with different types of users, such as the expert system for educational diagnosis proposed by [COHE*], and tutoring systems. Providing different information in an answer might also be done in explaining the behavior of an expert system which is used both as a teaching tool and as a problem solving engine.

Independently of whether the texts provided here are the most appropriate for a given user, we have added a new dimension along which a system can tailor its answers to users having different levels of domain knowledge, and TAILOR's ability to combine strategies is important, because it allows TAILOR to generate a greater variety of texts than would otherwise be possible. For example, for the pulse telephone which has a superordinate and three functionally important parts, TAILOR can generate eight different descriptions, even though the constituency schema is always chosen as the initial strategy (as the pulse telephone has no mechanism in its frame).[13] The ability to combine strategies allows a generation system to have distinct strategies clearly defined without losing flexibility. Discourse strategies help a system to construct a text with a given structure, and it is desirable to have such strategies to constrain the generation process. Given the strategies and no mechanism for combining them, a system is limited in the kinds of texts it can generate by the number of strategies. If combining strategies can be achieved, the variety of texts that can be generated is greatly increased.

By explicitly representing the user's knowledge about the domain, we have gained the ability to tailor texts to a whole range of user, without having a predefined set of user types.

7. Conclusions

In this paper we have demonstrated that the user's domain knowledge can be used as a factor in tailoring an answer. In particular, we have shown how the description of a complex physical object might be tailored to a user's level of expertise. We presented different kinds of knowledge users can have, explaining how a system can take them into consideration in order to generate a description. From our studies of texts, we have found two distinct discourse strategies that are used in describing complex devices. We postulated that the level of expertise of the user affects the *kind* of information given as opposed to just the *amount* of detail provided. Even though we conducted this study in the domain of complex devices, we believe this result extends to other domains, that is, that varying the kind of information to include in an answer is important in other domains as well, in order to tailor the

[13] Note that this number takes into account the fact that the heuristics chosen in TAILOR limit the number of combinations allowed. Furthermore, knowledge about basic concepts plays no role for this particular object. If it did, there would be yet more combinations.

answers to users with different levels of domain knowledge.[14] We thus proposed that a user model containing information about the user's domain knowledge can be used in a question-answering system to guide the decision process. We described two distinct descriptive strategies that can be used in a question-answering program and showed how they can be mixed to include the appropriate information from the knowledge base, based on the information contained in the user model.

Finally we presented TAILOR, a program that generates descriptions tailored to users with various levels of expertise. TAILOR employs one of the two discourse strategies described to generate a text for a novice or an expert. TAILOR is also able to automatically mix the strategies to provide device descriptions tailored to users whose domain knowledge falls anywhere along the knowledge spectrum. By representing explicitly the user's domain knowledge in terms of parameters, TAILOR does not require an *a priori* set of stereotypes but can provide a wide variety of descriptions for a whole range of users.

Acknowledgements. Many thanks go to Kathleen McKeown and Michael Lebowitz for their help in both the research and the writing of this paper, and to TjoeLiong Kwee and Jean Marc Lagoutte for their work on the functional grammar. We also thank Robin Cohen for her useful comments.

8. References

[ANDR77], [APPE82a], [APPE85], [BAKE57], [BRIT63], [BRIT84], [CARB83], [CARB88], [CARB*], [CHEV78], [CHI81], [CHIN86], [COHE*], [COLL62], [DAVS84], [ENCY82], [GRIC75], [GRIM75], [HOBB78a], [HOBB78b], [HOBB80], [HOEP84b], [KAPS82], [KAY79], [LANC87], [LEBO83], [LEBO85], [LEBO86], [LEHN77], [MANN84], [MARH87], [MCCO83], [MCCO86], [MCCO88], [MCCO*], [MCKE85b], [MCKE85c], [MCKE87], [NEW67], [PARI84], [PARI85a], [PARI85b], [PARI87a], [PARI87b], [QUIL88], [QUIL*], [RICH79b], [RICH*], [SHEP26], [SLEE82b], [SLEE85], [TENN78], [VERW62], [WALL82], [WASS83], [WASS85], [WEIS73], [WILS86], [WOOD73]

[14] Note that, for other domains, one would need to investigate the kinds of information that exist and which are appropriate for the different user type.

9 Highlighting a User Model to Respond to Misconceptions [1]

Kathleen F. McCoy

Abstract[2]

Responses to misconceptions given by human conversational partners often contain information refuting the possible reasoning which may have led to the misconceptions. This work describes generating such responses by reasoning on a highlighted model of the user to identify possible sources of the error. Domain-independent structural characterizations of a highlighted user model which suggest the source of several types of misconceptions are identified. Each is associated with a response strategy. The system can respond to a particular misconception by consulting the highlighted user model for one of the identified structural configurations and responding with the associated strategy. Transcripts of misconceptions corrected by human conversational partners show, however, that misconception responses must reflect the user's perspective on the objects involved. This context-dependent aspect of correcting misconceptions is captured by a new notion of object perspective which serves to highlight certain aspects of the user model due to the preceding discourse.

1. Introduction

When people interact with a database or expert system, it is reasonable to expect that they might reveal a misconception about an object modeled by the system. Since a human conversational partner would correct such a misconception if it was important to the current goals of the conversation, our database and expert systems should also be equipped with this ability.

A study of transcripts of humans responding to misconceptions revealed that a response to a misconception important to the current discourse goals of the participants can be viewed as consisting of three parts: (1) a denial of the incorrect information, (2) a statement of the correct information, and (3) justification for the

---

[1] This paper is a revised and extended version of [MCCO88]

[2] Some of this work was done while the author was at the University of Pennsylvania and was partially supported by the ARO grant DAA20-84-K-0061 and the NFS grant MCS81-07290.

denial and correction given. Further, the justification often seemed to refute support that might have led to the misconception.

In order for a computer system to mimic this behavior, a way of generating appropriate justifications must be found. The justification hinges on the support accorded the misconception. If we could (domain independently) characterize both the various kinds of support someone might have for a particular kind of misconception and an appropriate justification refuting that support, then the behavior could be mimicked. For instance, suppose the user gives an object a property it does not have (e.g., "What is the interest on this stock"). One way a person might come to such a misconception is by either confusing the object being discussed (e.g., stock) with a similar object that does have the property (e.g., a bond) or by making a bad analogy from such an object. A highlighted user model configuration that indicates such support would be one containing an important object which has the property in question and is similar to the object involved in the misconception. A reasonable response to use when such a user model configuration is present is one that points out the error so it does not continue (e.g., "did you mean bond?"). If we can characterize all support/justification pairs in this kind of domain independent way, a computer system could appropriately respond to entire classes of misconceptions.

The task of characterizing the possible supports for a misconception in a domain independent fashion, however, seems insurmountable since the support someone might have for a misconception is so varied. Further transcript analysis shows, however, that for a given type of misconception (e.g., a misclassification), only a few kinds of support were corrected for in the justification part of the response. In fact, one could abstract out a small number of correction strategies used by the human conversational partners, based on the kind of justification given for the denial and correction. Therefore, since the number of support/justification pairs is small, it is feasible to characterize correctable support for a misconception and associate each with a correction strategy whose justification refutes the support. A generation system that has access to a highlighted model of the user's beliefs about the domain could respond to a misconception by looking in this model for one of the characterized supports and respond with the associated strategy.

Note that in order for this method of correcting misconceptions to mimic what is found in human conversation, the user model must be highlighted by previous discourse. The kind of response given by a human conversation partner is not only dependent on the beliefs about the person being corrected, but also on the context in which the misconception occurred. For instance, we could imagine the following dialog where the user exhibits the misconception that T-bills have a penalty. A reasonable response is shown.

U. I am interested in investing in some securities to use as savings instruments. I want something short-term and I don't have a lot of money to invest so the instrument must have small denominations. I am a bit concerned about the penalties for early withdrawal. What is the penalty on a T-bill?

R. Treasury Bills do not have a penalty. Were you thinking of a Money Market Certificate?

This response might be prompted by R thinking that U came to the misconception by confusing T-Bills with a Money Market Certificate.

On the other hand, it is reasonable that the response might be different given a different preceding dialog. For example:

U. I am interested in investing in some securities. Safety is very important to me, so I would probably like to get something from the government. I am a bit concerned about the penalties for early withdrawal. What is the penalty on a T-bill?

R. Treasury Bills do not have a penalty. Were you thinking of a Treasury Bond?

This response may have been prompted by R thinking that a confusion between T-Bills and T-Bonds was the source of the misconception.

These responses could be generated by the methods described if the user model was affected by previous discourse and if this highlighting was taken into account when accessing the user model. Currently, when a user model is accessed, all user knowledge has equal importance. When people engage in a conversation, however, certain aspects of their domain model become more important than others. This importance is more than just a highlighting of those things that have been explicitly mentioned. Rather, certain things that are somehow related to those things explicitly mentioned in previous discourse are also highlighted. In fact, a way of looking at the domain is usually established.

A new notion of object perspective will be introduced that will enable this highlighting effect of previous discourse to be incorporated into the user model. Thus when the user model is accessed, certain things in it will be highlighted while other things will be suppressed. I will show how this variable highlighting of the user model can explain why the response to a particular misconception by a particular user may vary.

2. Knowledge Available

The work being done here is in the context of a natural language interface to a database or expert system. It is an attempt to define a module of an interface which could generate a cooperative natural sounding response to a misconception – the kind of response that would be generated by a helpful human conversational partner. In this work, a *misconception* is defined to be some discrepancy between system beliefs and user beliefs (as exhibited through the conversation). Upon encountering such a misconception, the job of the misconception corrector module is to attempt to bring the user's knowledge into line with the system's knowledge.

The scope of this work is limited by several assumptions about the kind of knowledge available. First, since we are concentrating on *object-related* misconceptions, we assume the system's model of the world contains an object taxonomy with attribute/value pairs attached to the objects.

Second, we assume that we have a user model available. Thus we will not be concerned with inferring a model of the user. Inferring the user model is the subject of several other papers in this volume (see [RICH*, CHIN*, FINI*] for example). We further assume that the model of the user is in the same form as the system

model. That is, it contains an object taxonomy with attribute/values attached. The content of the user model, however, may be quite different from the content of the system's model of the world. Thus, the way objects fit into the object taxonomy and which attributes are associated with which objects may differ between the two models. I assume that the user model starts out as a model of a canonical user and is updated as the conversation progresses.

Third, we assume the system has available to it certain pieces of contextual and discourse information which serve to highlight the user model. This highlighting (explored below) is gained from a new notion of object perspective and from a record of items and attributes which have been explicitly focused on in the discourse.

Given the kind of information assumed in the system's and user's models of the world, there are two kinds of misconceptions that may occur: misclassifications (the user may place an object in the object taxonomy where it does not belong) and misattributions (the user may give an object an attribute/value it does not have). We are interested in generating a response to these kinds of misconceptions in situations in which the goals of the conversation warrant that a "full" misconception response be given. For each of these kinds of misconceptions, a small number of response strategies[3] were found in the transcript study that could be supported by the assumed knowledge bases. These strategies were abstracted into response schemas. In Section 4, examples of the response strategies found for misclassifications will be examined. For each strategy an abstract specification of the content will be given. Next we will look at what beliefs about the user might have prompted the use of each strategy, and characterize these beliefs in terms of the structure of the highlighted user model. A misconception can be responded to by looking in the highlighted user model for one of the pre-identified structural characterizations and instantiating the associated schema. Section 5 examines response strategies for misattributions. The sections following that will concentrate on the highlighting of the user model due to object perspective. Object perspective will be defined and it will be shown how the highlighting gained from object perspective affects responding to misconceptions. In particular, perspective's influence on the assessment of object similarity (crucial in the identified structural configurations) will be discussed.

The methods described here have been implemented in the ROMPER system (Responding to Object-related Misconceptions using PERspective). The system takes as input a specification of the information that is inconsistent with the system's model of the world, the current perspective (described below), and a record of past focus. It produces a formal specification of the response. This response specification is passed into the Mumble system [MCDO80] which, using a grammar and dictionary written by Robin Karlin [KARL85], produces an actual English response.

The implemented system works in the financial securities domain. In order to show the generality of this approach, two different domains will be used in this paper. The motivation for the system's method of choosing a response strategy will use examples from the domain containing whales and fish. In the last sections of the paper (those pertaining to the new notion of object perspective), the examples

[3] The dialogs contained in this paper were not taken directly from the transcripts. They are used to illustrate the kind of responses found.

will be taken from the domain of financial securities since the ideas can be better motivated with a more complex domain.

3. Related Work on Correcting Misconceptions

The method for responding to misconceptions described in this paper should be contrasted with the way that misconceptions are handled by AI systems today. For the most part misconceptions have been left to the intelligent computer aided instruction (ICAI) systems which basically use an *a priori* listing of misconception-response pairs (see [BROW78, SLEE82b, STEV79, STEV80, WOOL83, WOOL84a], etc.). Responding to a misconception is reduced to matching the user's input against the *a priori* listing and outputting the associated response should a match be successful. The major problem with these systems is that it is impossible to list *every* misconception and, due to the system's inability to reason about the misconception itself, the system is completely at a loss when faced with a misconception absent from the *a priori* listing.

The work of Sleeman [SLEE82b] on inferring defective algebra rules (mal-rules) is an attempt to avoid the hand-encoded *a priori* listing of misconceptions. Sleeman proposes on-line inference of mal-rules based on the answer the student has given to a particular problem. The problem is worked backward from the answer using correct rules as long as possible. A mal-rule is then inferred which "closes the gap" between the problem gotten by working backward from the student's response and the actual problem given. Although Sleeman's work is a major improvement over the *a priori* listing approach, it still has several problems. First, there is no measure of how *reasonable* or *likely* a particular mal-rule is. In addition, once a mal-rule has been inferred, no indication is given concerning how the misconception should be corrected.

[KAPS79and MAYS80a] differ from the previous work mentioned in that it, like the work described here, is concerned with correcting an entire *class* of misconceptions. Thus a system based on Kaplan and Mays' methods can handle a potentially infinite number of individual misconceptions. They were concerned with correcting *presumption failures* occurring in the context of a natural language interface to a database system. They worked on detecting and correcting such misconceptions based on domain independent linguistic cues from the user and an enhanced model of the domain. For instance, the query "Which faculty take courses?" indicates a presumption failure. A truthful response of "none" to this query would confirm the user's erroneous belief that faculty can take courses. Mays suggests correcting a query like the one above by (1) denying that the "take" relation can hold between faculty and courses, and (2) describing all correct alternatives which can be reached by abstracting on each of the objects and on the relation involved. This method would produce the following kind of response:

R. I don't believe that faculty can take courses. Faculty teach courses. Students take courses.

Although responses such as this would probably be helpful to the user, they have the potential (given a complicated domain) for being overly verbose and containing information that the user does not care about. This paper can be seen as an extension of Kaplan and Mays' work in that it offers a more sophisticated way of correcting a misconception once it has been detected.[4]

[QUIL*] discusses an approach to correcting misconceptions that is very similar to the one advocated in this paper. In Quilici's case, however, the class of misconceptions is different. While this work is concerned with correcting misconceptions about objects, Quilici concentrates on correcting misconceptions about actions. His user model contains information about plan knowledge and he has characterized support for several types of misconceptions in terms of this knowledge. As is the case with this work, each kind of support has an associated response strategy. When a misconception is detected, the system analyzes its user model and system knowledge to determine the probable support and responds accordingly.

4. Misclassifications

Three major strategies for correcting misclassifications that could be supported by the assumed knowledge bases were found in the transcripts. The first is exemplified by the following dialog:

U. I thought whales were fish.

R. No, they are mammals. You may have thought they were fish because they are fin-bearing and live in the water. However, they are mammals since, (while fish have gills) whales breathe through lungs and feed their young with milk.

Let us first examine the content of this response. It can be seen as consisting of three parts. In the first sentence we have the denial of the incorrect information and the statement of the corresponding correct information. The remaining sentences comprise the justification for the denial and correction given. Notice that it refutes a kind of support that may have led to the misconception. It is easiest to see this by examining an abstraction of the response and looking at the content of the justification:

MISCONCEPTION = X *is-a* Y
RESPONSE =

1. X *is-NOT-a* Y
2. X *is-a Type-of*(X)
3. X is like Y because both share *attributes-of*(X) ∩ *attributes-of*(Y)
4. BUT X has *attributes-of*(X) − *attributes-of*(Y)
5. WHILE Y has *attributes-of*(Y) − *attributes-of*(X)

The justification is contained in 3-5 of the rule. Notice that it can be seen as a concede/override pair. First, information is conceded that actually supports the

[4] Correction was not the major thrust of Kaplan or Mays' work.

misconception. This consists of attributes that whales and fish have in common. These attributes are a potential source of the misconception since an object that shares many attributes with a superordinate is likely to belong to that superordinate class. By conceding the intersecting attributes, R is acknowledging the apparent support for the misconception lest U think that that part of his/her domain model is wrong.

The response goes on to override the conceded information by offering attributes that differentiate whales and fish. These are attributes that whales have but fish cannot, and attributes that fish have that whales do not.

By analyzing the justification in this way, not only do we get an abstract representation that can be used by a generation system, but we get some insight into when it would be reasonable to use such a response. It makes sense to use a response that first concedes a similarity between the misclassified object and the object that was wrongly given as its superordinate (henceforth called the *posited superordinate*) and then overrides that conceded information, when such a similarity is a possible source of the misconception. This might be the case when *the highlighted user model shows that the user believes that such a similarity exists.* Thus a generation system may generate its response by instantiating the above rule if the user has misclassified the object X as a Y and an analysis of the highlighted user model shows that the user believes X is similar to Y. Because of this reason for using this response strategy, it has been termed the "like super" response strategy.

A second kind of strategy is exemplified by the following dialog. Imagine a recent conversation centered on sharks being large fish that people are typically afraid of. Following that, the conversation turns to classifying whales. The user says:

U. I thought whales were fish too.

R. No, they are mammals. You may have thought they were fish since they are like the fish, sharks, in that both are large aquatic creatures and both scare people. However, whales are mammals since, (while fish have gills) whales breathe through lungs and feed their young with milk.

Again, this response can be seen as consisting of the three parts with the denial and correction being contained in the first sentence. As in the previous example, the remaining sentences comprise the justification in the form of a concede/override pair. The conceded information, while dependent on attributes, differs from that given above. The abstract content rule can perhaps best illustrate this difference:

MISCONCEPTION = X *is-a* Y

RESPONSE =

1. X *is-NOT-a* Y
2. X *is-a Type-of* (X)
3. $\exists$ Z (Z *is-a* Y $\wedge$ X is like Z
   because both have *attributes-of* (X) $\cap$ *attributes-of* (Z))
4. HOWEVER X *is-a Type-of* (X)
   because X has *attributes-of* (*Type-of* (X)) $-$ *attributes-of* (Y)

The major difference between this response and the like-super response is that instead of conceding a similarity between the misclassified object and the posited superordinate itself, a similarity between the misclassified object and a *descendant* of the posited superordinate is conceded. This response is reasonable if the user arrived at the misconception by reasoning: "I don't know how to classify whales, but I do know how to classify sharks. Sharks are fish and whales are very similar to sharks. Therefore it is reasonable to believe that they are classified the same." Evidence for this kind of reasoning might be a similarity found in the highlighted user model between the misclassified object and some descendant of the posited superordinate. Finding such a similarity would justify the use of this "like-some-super" strategy.

A final strategy found for misclassification is exemplified by the following:

U. I thought whales were fish.
R. No, they are mammals.

This is the degenerate "no support" strategy which contains no justification for the correction. At first glance the use of this strategy seems rather puzzling. Since I have assumed that the misconception is important, one would expect a full response to be given. However, no justification was included in the response.

This response can be explained in the following way. The model for correcting misconceptions says that there are a small number of ways in which a correction of a particular kind of misconception can be justified. Each kind of justification refutes a kind of faulty reasoning that may have led to the misconception. Thus, in order to respond to a misconception, the highlighted user model is consulted in order to find evidence for one of the pre-identified kinds of faulty reasoning.

This last response is an example of a response in a situation where no evidence was found for any of the pre-identified kinds of faulty reasoning. If R could not figure out what led to the misconception, then R could not include justification to refute the support. The most that R could do in this situation was deny the incorrect information and state the corresponding correct information. For this reason the response has been termed the "no-support" response and it can be used when the highlighted user model contains none of the evidence that would key off any of the other correction strategies.

5. Misattributions

Misattributions are the second kind of misconceptions that may arise given the kind of knowledge assumed. A misattribution occurs when the user gives an object an attribute/value that does not exist in the system's domain model. In the case of misattributions, the justification took the form of offering another statement that would change what the user said in a small way in order to make it a correct statement. This is illustrated in the following dialog which is an example of the first kind of correction strategy found. Imagine that we are discussing sea animals and have previously discussed sharks and their attributes. We move to whales:

U. I thought whales had gills.
R. No, they breathe through lungs. Were you thinking of sharks?

The response might go on to discuss the similarities and differences between whales and sharks.

Notice what was done in this response. We have the denial and the correction as before. The justification offers another object that is similar to the object involved in the misconception, but for which the misattribution is true. This content is captured in the following rule:

MISCONCEPTION = X has *attribute* Y with *value* V
RESPONSE =
1. X does not have *attribute* Y with *value* V
2. X has *attribute* Y with *value* W
3. ∃ Z (Z has *attribute* Y with *value* V) did you mean Z?

This response would be effective if the user's misconception results from either confusing the object being discussed with some other object that has the misattributed property, or through an incorrect analogy with such an object. Mention of the object in the case of confusion prevents continuation of the confusion. In the case of an incorrect analogy, mention of the other object alerts the user to the inappropriateness of the analogy and also prevents the continued use of the analogy.

It makes sense to use the "wrong object" strategy when a possible source of the misconception is either an object confusion or the use of a bad analogy. Evidence for this would be the existence of some object from the highlighted user model that has the misattributed property and is similar to the misconception object.

A second kind of correction strategy used for misattributions is shown here:

U. I thought whales had gills.
R. No, whales breathe through lungs. Did you mean you thought they had fins?

This is very similar to the last response except, instead of offering another object, the strategy offers a similar attribute that makes the user's statement correct. The content rule for this response follows:

MISCONCEPTION = X has *attribute* Y with *value* V
RESPONSE =
1. X does not have *attribute* Y with *value* V
2. X has *attribute* Y with *value* W
3. ∃ Q ∃ U(X has *attribute* Q with *value* U ∧ similar(Y,Q)
   did you mean X has *attribute* Q with *value* U?)

This strategy, termed the "wrong attribute" strategy, is used when there is a similarity of attributes within the highlighted user model.

The no-support strategy also occurs in the case of misattributions.

6. A Rule for Choosing a Strategy

Given the association of highlighted user model configurations with response strategies given in the previous sections, we can come up with a method for deciding

how to respond based on the type of misconception along with an analysis of the highlighted user model. Basically the highlighted user model analysis looks for one of the pre-identified configurations and, if one is found, suggests instantiating the associated strategy. The following rule captures the way this is implemented in the ROMPER system.

IF misconception = "X *is-a* Y"
  THEN
    IF *similar*(X,Y)
      THEN instantiate like-super schema
    ELSEIF ∃ Z (Z *is-a* Y) ∧ *similar*(X,Z)
      THEN instantiate like-some-super schema
    ELSE instantiate no-support schema
ELSEIF misconception = "X has *attribute* Y with *value* V"
  THEN
    IF ∃ Z ((Z has *attribute* Y with *value* V) ∧ *similar*(X,Z))
      THEN instantiate wrong-object schema
    ELSEIF ∃ Q ∃ U ((X has *attribute* Q with *value* U) ∧ *similar*(Y,Q))
      THEN instantiate wrong-attribute schema
    ELSE instantiate no-support schema

This rule is a simplification since it goes through the configurations in a pre-determined order and chooses the first strategy with a matching configuration. Thus it places preferences on some strategies over others. While this preference is well motivated in the choice of a strategy containing justification over a no-support strategy (see [MCCO87] for details) it is a simplification for choosing between two strategies containing justification. An arbitrary preference has been made for the like-super strategy over the like-some-super strategy and for the wrong-object strategy over the wrong-attribute strategy. In reality a more complicated model which is perhaps based on the discourse situation is needed.

For instance, suppose we have just been discussing sharks and you misclassify whales as fish. It probably makes more sense for me to check the configurations that might involve sharks (or any other object recently focused on) rather than those that involve things foreign to the conversation so far. In this situation if the highlighted user model indicates a similarity with whales and sharks, the like-some-super strategy should probably be used even though the highlighted user model might also indicate a similarity between whales and fish. Precisely how this should be done is an area of future research.

7. Highlighting and Object Similarity

In the previous section a rule was introduced that shows how the ROMPER system determines which schema to use for a particular misconception. This rule calls for

analyzing a highlighted user model. In this section we show how the highlighting of the user model is taken into account in the above rule. Notice that the user model analysis in each case described above hinged on an assessment of object similarity. It is in this assessment that the highlighting on the user model is taken into account in the ROMPER system. ROMPER's similarity metric allows several contextual components to be taken into account. One of these components is the highlighting of the user model gained from object perspective.

To date, most AI systems do not assess object similarity in a way that is context-dependent. Several systems that do assess object similarity (e.g., [RUME73, MCKE82, CARB84, WEIN84]) use a metric based on distance in the generalization hierarchy. Basically, two objects that have a common immediate superordinate (i.e., are siblings in the hierarchy) are seen as very similar, while objects whose lowest common ancestor is several levels up in the hierarchy are seen as quite different.

One problem with using a metric based on distance in the generalization hierarchy has to do with the way in which this distance should be defined. The metric yields ambiguous results when objects can be classified in more than one way. With such a rich classification, there may be several lowest common ancestors of the objects being compared. A decision must be made about which of these lowest common ancestors should be considered since the similarity assessment of the objects might vary widely as a result. For instance, a treasury bond and a corporate bond may be assessed as being very similar since they have a common immediate superordinate of bonds. On the other hand, both of these objects can be classified along other dimensions. If we look at a treasury bond as being a treasury issue which is a type of US Government security, it seems quite different from a corporate bond which is a corporate security.

A second major problem with the distance similarity metric is that it is *context invariant*; contextual information has no way of affecting the assessments. However, in order for our model of correcting misconceptions to work, it is crucial that the similarity metric be sensitive to context. In addition, human judgements of object similarity [TVER77] have been found to shift both when the set of objects under discussion are altered (e.g., a violin and an electric guitar may be judged quite similar when in a group with a clarinet and an oboe, and may be judged quite different when the other members of the group are a cello and an electric bass), and when the salience of attributes is altered (e.g., in a group containing a red triangle, a blue triangle, and a red square, the red triangle might be judged similar to the blue triangle when attribute shape is stressed, but may be judged similar to the red square when attribute color is stressed).

One metric that avoids these problems was introduced by Tversky ([TVER77]). Tversky's metric, rather than relying on distance in some space, is based on the common and disjoint features of the objects involved. The metric, termed a contrast model, allows context to be taken into account in several places.

Suppose we have two objects a and b where A is the set of properties associated with object a and B is the set of properties associated with object b. Tversky's measure can be expressed as:

$$s(a,b) = \theta f(A \cup B) - \alpha f(A - B) - \beta f(B - A)$$

for some θ, α, and $\beta \geq 0$.

In the above equation θ, α, and β are parameters which represent the importance of each component of the equation. The function f maps over the features and yields a salience rating for each. In essence, the contrast model states that the similarity of two objects is some function of their common features minus some function of their disjoint features. The importance of each particular feature involved (determined by the function f) and the importance of each component of the equation (determined by θ, α, and β) may change with context.

Although Tversky discusses in general terms how these functions might be set, he gives no concrete methods for doing so. For instance, to set θ, α, and β he turns to the relative prominence of objects *a* and *b* in the discourse. The more prominent an object is, the more its attributes should have an impact on the similarity rating. Thus, finding the relative prominence of objects *a* and *b* in the discourse would help set these values. If *a* is relatively more important, then functions θ and α should be greater than β, resulting in the attributes of the more prominent object having a greater influence over the similarity assessment. While I would conjecture that information about the focus of the discourse [GROS81, SIDN83a, GROS83] might give an indication of an object's prominence and would therefore be useful in setting the values of θ, α, and β, in this work I have assumed a value of 1 for the θ, α, and β and have concentrated on setting the f function.

We turn, then, to the problem of finding a value for the f function: the measure of salience for each property of the objects involved. Other work, such as [CARN70b, WEIN84], has hand-encoded salience values for attributes of individual objects directly into the knowledge base, permanently setting the f function. This approach is not sufficient for setting the f function for Tversky's metric since it is crucial that the f function be able to change with context. In order to make this happen, the salience values computed by f must change with context.

To see this, consider our ability to explicitly mention an attribute to increase its salience. In the example earlier with the red and blue triangles and the red square, if the request for a similarity judgement had been preceded by "Look at the pretty colors of these objects", the red triangle and red square would have probably been judged to be most similar. Thus we see that explicit mention of an attribute in a discourse is one way that the f function might be affected by previous discourse.

Explicit mention of an attribute is only one way in which the salience of the attributes may change dynamically. Another aspect of dynamic salience comes from the point of view, or perspective, applied to the domain. A building can be referred to as being an architectural work, for example, or as being someone's home. The two different views of the building cause different sets of attributes to become salient. Notice that this set of attributes is in addition to the attributes that have been explicitly mentioned in the discourse. From an architectural work point of view, attributes like the architect's name, date of building, and particular architectural features become salient. On the other hand, from the home point of view, attributes like the kitchen size, number of bedrooms, and living space become important. If we can find a way of modeling how these "precompiled" sets of attributes become highlighted in a discourse, we will have a principled method for setting the f function needed for Tversky's similarity metric. The next section discusses how this highlighting can be modeled by a computer system.

8. Object Perspective

The notion of point of view or object perspective has been noted by other researchers in artificial intelligence. Its ability to explain the changing attribute salience has been attributed to a limited inheritance mechanism (see e.g. [GROS77b, BOBR77, TOU82]). According to the limited inheritance theory, an object viewed from a particular perspective is seen as having one particular superordinate, although in fact it may have many. The object inherits properties only from the superordinate in perspective. Therefore different perspectives on the same object cause different properties to be inherited (and therefore highlighted).

While the limited inheritance theory of object perspective is intuitively appealing, in practice it is problematic (see [MCCO85a] for details), and is unable to handle some additional effects which should be handled intuitively by object perspective. Some of these effects will be discussed here.

The first has to do with the availability of object attributes. The proposed limited inheritance mechanism makes attributes inherited from superordinates other than the one in perspective *unavailable*. This seems a bit too strong. When we discuss a building as an architectural work, I may comment on the number of bedrooms the building has. While you may think my comment irrelevant to the current conversation, you would still be able to understand it and even evaluate its truth or falsity. In the limited inheritance account of object perspective, however, this would not be possible. You would not even be able to tell that the number of bedrooms is an attribute the building might have.

A second problem with the limited inheritance account of object perspective concerns its close ties with the generalization hierarchy. The account says that an object viewed through a particular perspective inherits the attributes only from the superordinate in perspective. What attributes does that superordinate have? The superordinate itself may have multiple classifications and thus potentially multiple perspectives. Thus, in order to figure out what attributes a particular concept should inherit, we must figure out not only what perspective it is being viewed from, but also what perspective the perspective superordinate is being viewed from, and so on. But this seems to be much more work than is necessary!

A third problem with the limited inheritance account of object perspective has to do with the scope of a perspective; a perspective may be taken on an individual object. However, during the course of a conversation it is usually the case that more than one object will be discussed. When this happens, usually the same kinds of things are discussed about the objects. In essence, a particular highlighting of attributes (or point of view) seems to be in force during the conversation. Yet this highlighting is applied to different objects – some of which may not even have the same superordinates. What seems to be happening is that the conversational partners are viewing an entire group of objects from the same perspective. This cannot be accounted for by the previous definition of object perspective unless each of the objects under discussion can be said to have the same (immediate) superordinate.

A final effect which is not accounted for by the limited inheritance mechanism, yet seems to hinge on the view being taken of the domain, has to do with the heightened importance of some objects during a discourse. Like the importance of

attributes, the relative importance of some objects in the discourse cannot solely be accounted for by explicit mention. Some objects are more likely to be mentioned and discussed in a discourse than others. For instance, when discussing a building as an architectural work, I might reasonably mention the library down the street that was designed by the same architect. On the other hand, I will probably not mention my apartment. Along the same lines, in discussing that building as a home, my apartment is a likely candidate for mention in the conversation. Thus, this effect seems to be tied to the notion of object perspective, yet is not addressed by the limited inheritance account.

In light of these problems, a new definition of object perspective is needed. We need the new definition to be able to account for the dynamic highlighting of "precompiled" groups of attributes. At the same time, we need it to avoid the problems of the limited inheritance account. We propose that the following account be used:

1. Instead of tying perspective into the generalization hierarchy of objects as has been done in the past, the new notion of perspective is independent of that hierarchy. "Perspectives" which can be taken on the objects in the domain will be defined and will be located in a structure which is orthogonal to the generalization hierarchy.
2. A number of perspectives are available for any domain of discourse and any given domain object may be viewed from any one of several perspectives for that domain.
3. Each perspective comprises a set of attributes with associated salience values. It is these salience values that dictate which attributes are highlighted and which are suppressed.
4. One such perspective is designated *active* at any particular point in the discourse.[5]

This notion of object perspective works as follows. An object or group of objects is still said to be viewed through a perspective. In particular, any object which is accessed by the system is viewed through the current *active* perspective. However, instead of dictating which attributes an object inherits, the active perspective affects the salience values of the attributes that an object possesses (either directly or inherited through the generalization hierarchy). The active perspective essentially acts as a filter on an object's attributes. By raising the salience of the attributes, it highlights those attributes which have a high salience rating in the active perspective. By lowering the salience of the attributes, it suppresses those attributes which are either given a low salience value or do not appear in the active perspective.

Let us look at how this new notion of object perspective handles the problems with the old account. The first two problems are eliminated rather easily by the definition itself. In this new account, the perspective does not affect what attributes

[5] Saying that exactly one perspective is active is actually a simplification. It may be the case that a number of perspectives can be active. In this case the resulting "active perspective" will be some function of the individual active perspectives. The exact nature of this function is an open research question.

an object inherits; it simply affects the salience values on those attributes. As a result, we do not have the problem, that of some attributes being unavailable. Instead, their salience rating makes them unimportant in the current context.

The second problem, that of potentially having to assess perspectives on many objects in order to evaluate the perspective on the object under discussion, is handled by decoupling the perspectives and the generalization hierarchy. In the new definition, "perspectives" are defined broadly enough so as to be applicable to any domain object (see below for an example). As a result, one avoids the problem of having to evaluate the perspective of a superordinate in order to determine how that superordinate's attributes should be highlighted.

The third problem with the previous notion of object perspective was that it was not able to handle the feeling of a perspective being evident for a whole group of objects. This new notion of object perspective handles this by forcing it to be the case. As long as the active perspective does not change, every object that is accessed by the system is viewed through the same perspective.

Finally, this notion of object perspective is able to account for the feeling of heightened object importance within a perspective. An object is important within a perspective if it has many of the attributes to which the perspective gives a high salience rating. It can be thought of as the whole becoming highlighted because the parts are highlighted. In other words:

$$important(X) = \sum_{X \ has \ P} salience(P)$$

In the example given earlier, the library was highlighted (and thus worth mentioning) in the architectural works perspective because it contained many attributes highlighted by that perspective. On the other hand, my apartment was unimportant in that perspective because it didn't contain any attributes made salient in the architectural works perspective.

9. Using Perspective to Set f

We return to the problem of setting the f function in Tversky's metric. f maps over the attributes of the objects and returns a salience rating for each. So far we have identified three sources of salience values:

1. static salience encoded directly into the generalization hierarchy.
2. dynamic salience gained by an attribute being explicitly mentioned in a discourse.
3. perspective salience gained by viewing the domain objects through the active perspective.

I propose that the f function for Tversky's metric be set by taking into account all three of the above components of salience. This would yield an f function that was context-dependent and would help the similarity metric exhibit many desirable properties.

10. Modeling a Domain with Perspectives

In some natural language systems, a model of a particular domain includes the usual object taxonomy containing all of the objects in the domain and all of the attributes associated with those objects. In some systems (e.g., [CARN70b, WEIN84]), the attributes of the objects in the domain model are given static salience values to reflect the attributes' importance. The norm in building knowledge bases is to omit even the static salience values and to assume equal importance for all attributes an object possesses. We will show how a typical knowledge base can be augmented with perspective information and how this information is taken into account when accessing the knowledge base.

Figure 1 shows a small segment of a typical domain model. The domain is that of financial securities. Three of the objects from this domain, Money Market Certificates, Treasury Bills, and Treasury Bonds, are shown with the attributes they possess.

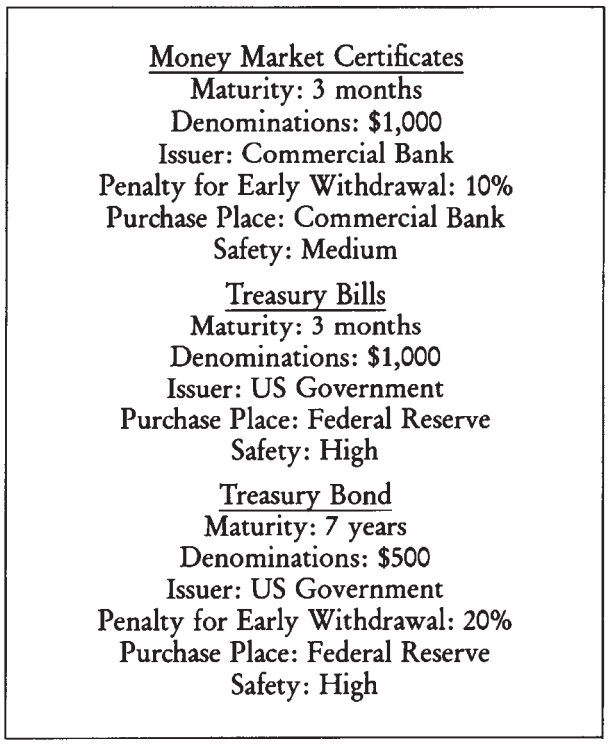


Figure 1. Objects in "flat" domain model

In systems as they are defined today, a group of objects defined in this way and arranged in a generalization hierarchy would constitute the domain model. If a system were to access any one of the objects in the domain it would be given the objects with attributes as they are listed in the figure.

In order to get a dynamic highlighting of the domain model, we must build, in addition to the object taxonomy, a separate structure containing the perspectives that can be taken on the domain objects. This means that we must think about the

different points of view that can be taken on the objects in the domain and compile sets of attributes from our model which capture the important domain concepts in that point of view. There will be attributes in each perspective that do not occur with all of the objects in the domain. At the same time, there will be attributes of individual objects that do not appear in a particular perspective. The perspective simply captures the attributes that are important in a particular point of view. The perspectives found in Figure 2 might be reasonable for the domain where we are assuming salience values from low salience of 0 to high salience of 1).

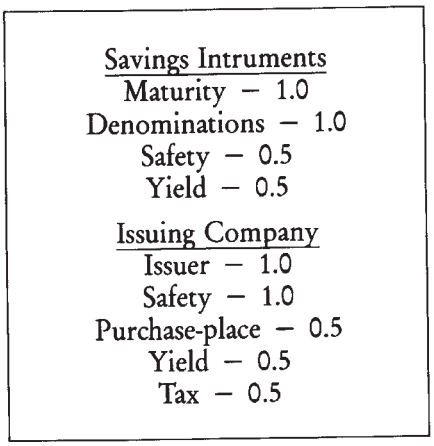


Figure 2. Sample perspectives

The perspective of Savings Instruments highlights maturity and denominations, and somewhat highlights safety and yield. This indicates that when people are discussing securities as savings instruments, they are most interested in how long their money will be tied up and in what denominations they can save their money. The perspective of Issuing Company, on the other hand, highlights different attributes. When securities are discussed from this perspective, things like the name of the company and the stability of an investment in the company become important. Other attributes of the securities are ignored (recall that attributes not mentioned in the perspective get assigned a low salience rating).

Notice how the objects look when accessed, depending on which of the two different perspectives are active. For instance, through the savings instruments perspective the objects' attributes take on the salience shown in Figure 3.

Attributes are not shown in the figure that have 0 salience. Recall that when no static salience is given, the attributes of the objects derive their salience directly from the active perspective. Attributes in the perspective that the objects do not have (e.g., yield) are ignored. Attributes of the objects not occurring in the perspective are given the lowest salience rating. The very same objects look different when viewed through the other perspective. This is shown in Figure 4.

The salience values derived from perspective can be used for many tasks. Section 12 will show how they can be used in conjunction with Tversky's similarity metric to generate context sensitive responses to misconceptions.

Another task that perspective is useful for is in generating more concise responses to users. For instance, the schemas for correcting misconceptions intro-

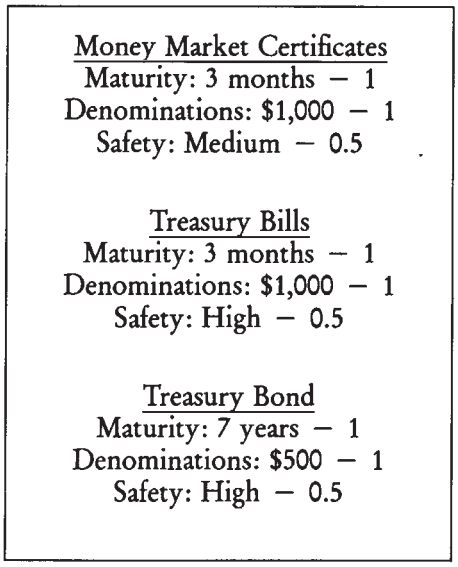


Figure 3. Objects throught savings instruments perspective

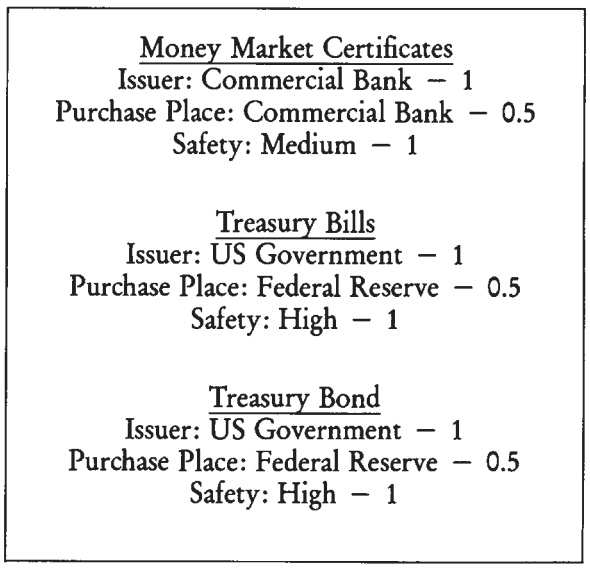


Figure 4. Objects throught issuing company perspective

duced earlier call for the inclusion of sets of attributes in many of the responses. For example, the like-super schema used for misclassifications calls for listing the attributes the misconception object and the posited superordinate have in common. In many cases this set of attributes may be quite large. In addition, the user is not likely to be interested in the entire set. Perspective is used in ROMPER to limit the listed attributes to those that are likely to be most interesting to the user by listing those attributes that are given high salience in the active perspective.

Cohen and Jones use a notion similar to object perspective to limit the information included in their responses [COHE*]. The difference between the two notions is that while the attributes given high salience in a perspective can be described as what things are currently of interest, their notion is more closely tied to what the system believes the user knows. They limit the information included in a response to that which the system believes the user does not know. The idea is that the unknown information is most helpful and interesting to the user. Cohen and Jones' work is in the context of an expert system where the system knowledge is partitioned into several classes. Different types of users are assumed to know about different classes of knowledge. When the system is asked to explain its reasoning, the "perspective" chosen is one that includes information from the classes of knowledge that the user is believed not to know. Thus their notion can be seen as a crude highlighting mechanism where the knowledge classes unknown to the user are highlighted. In the case of Cohen and Jones, however, this "highlighting" is achieved in response schemas instead of in the knowledge base access as is the case with the perspectives described here.

11. Choosing the Active Perspective

In the implemented ROMPER system, the perspective being taken on the domain is input to the system. In order for the notion of object perspective to be truly beneficial, however, there must be a mechanism for choosing the active perspective based on previous discourse. While this topic is still very much open to investigation, some preliminary research has revealed several factors that might influence the choice of active perspective.

Perhaps one of the most influential pieces of information useful in choosing a perspective is the user's current goal. The task of determining and tracking the user's goals throughout a discourse is discussed in this volume in [CARB*]. In [MCKE85c] the user's goal completely determines which perspective is active. In their work, each perspective which can be taken on the domain objects is indexed by potential goals. Thus, once the system has determined what the user's goal probably is, it has also determined what perspective the user has probably taken on the domain objects.

While the user's goal is a good source of information to use to determine the probable perspective, other factors may also influence this choice. These include the attributes and objects mentioned so far in the dialog. The mentioned attributes are obviously thought to be important and one would therefore expect them to be given a fairly high salience rating in the active perspective. Thus, the choice of active perspective can be narrowed down to those in which the mentioned attributes appear with high salience.

Using the mentioned attributes to determine the active perspective is similar to what is done in [MORI*] in determining *isotopy*. The notion of isotopy is similar to that of object perspective. It is a kind of coherence found in a discourse established by semantic ties or threads found from one exchange to the next. The semantic ties from isotopy are based on the attribute/values mentioned in the dialog. For instance, in Morik's system, a user booking a hotel room might first ask about the view the room has and then about whether or not the hotel has a night club. While these two attributes might seem unrelated, they do have the common feature (isotopy) of determining how quiet/loud the room is. Thus the system, using this information, might offer additional information which also addresses the quietness/loudness of the room. The notion of isotopy is similar to perspective in that the attributes which provide the coherence in isotopy would likely be found in the same perspective as defined in this paper. For instance, one could imagine a "loudness" perspective in the hotel domain which contained all of the attributes which contributed to the loudness of a hotel room. A helpful system might offer attributes given high salience in this perspective to a user who has shown interest in the loudness of the room by referring to several of the attributes given high salience in that perspective.

Another source of information which can be used in determining the active perspective is the objects mentioned so far in the dialog. One would expect that the active perspective would deem the mentioned objects important. Therefore the system might look for perspectives that give high salience ratings to many of the attributes associated with objects that have been mentioned in the discourse.

In this section I have identified several factors which influence the choice of active perspective. This choice, however, is a question which remains as an open research topic. Still unanswered are questions such as: when does the active perspective change? How long is a perspective active? Is there a relationship between a discourse unit [GROS85] and perspective? Is there any structure to the space of perspectives that would put constraints on moving from one active perspective to another? These questions must be taken up in future research on perspective.

12. Perspective's Influence on Responses

In this section an example is given which indicates how the choice of perspective influences the correction of a misconception. Recall that in correcting a misattribution one of the correction schemas used by ROMPER called for a similar object to be offered as a possible object of confusion. A study of transcripts reveals, however, that this schema may be instantiated in different ways depending on the context. Consider once again the following dialogs that were first seen in the introduction:

U. I am interested in investing in some securities to use as savings instruments. I want something short-term and I don't have a lot of money to invest so the instrument must have small denominations. I am a bit concerned about the penalties for early withdrawal. What is the penalty on a T-bill?

R. Treasury Bills do not have a penalty. Were you thinking of a Money Market Certificate?

In this case the money market certificate was seen as being similar to the treasury bill and therefore included in the response. A different object might be used in a different context. Consider:

U. I am interested in investing in some securities. Safety is very important to me, so I would probably like to get something from the government. I am a bit concerned about the penalties for early withdrawal. What is the penalty on a T-bill?

R. Treasury Bills do not have a penalty. Were you thinking of a Treasury Bond?

The difference in these two responses can be explained by different perspectives being taken on the objects. Suppose we are given the objects, attributes, and perspectives from Section 10. The dialog preceding the first example could lead to the establishment of savings instruments as the active perspective[6] since it mentions the perspective "by name" and explicitly mentions several of the attributes made important by the perspective.

A system attempting to respond to this misconception might proceed by attempting to instantiate the wrong object schema described in Section 5. Recall that

[6] Note that the ROMPER system does not choose the active perspective – it is given as input to the system. This example is simply used to illustrate perspective's influence on misconception responses.

this schema is applicable when there is a similar object which has the property involved in the misconception. The system might collect all objects which have the attribute in question and then test their similarity with the object involved in the misconception. In our knowledge base there are two objects which have the attribute involved in the misconception: Money Market Certificates and T-Bonds.

Assume that the f function in the metric is set solely on the basis of the salience values given by the perspective. Further assume that a positive value returned from Tversky's metric indicates high similarity, while a negative value indicates low similarity. Applying the Tversky metric using the salience values attached by the savings instrument perspective (and assuming a value of 1 for θ, α, and β) we get:

s(T-Bill, MM-Cert) = f(maturity, denom) − f(safety)
= 2 - .5 = 1.5 → high similarity

s(T-Bill, T-Bond) = f(safety) − f(maturity, denom)
= .5 - 2 = -1.5 → low similarity

With these calculations the system would choose the Money Market Certificate as the possible object of confusion and respond something like:

R. Treasury Bills do not have a penalty. Were you thinking of a Money Market Certificate?

Contrast the above calculations with calculations that might occur given a different active perspective. The discourse preceding the misconception utterance in the second example suggests the active perspective of "Issuing Company". Using the salience values attached by this perspective the similarity metric would produce the following calculations:

s(T-Bill, MM-Cert) = f() - f(issuer, safety, purchase)
= 0 - 2.5 = -2.5 → low similarity

s(T-Bill, T-Bond) = f(issuer, safety, purchase) − f()
= 2.5 - 0 = 2.5 → high similarity

In this case a reasonable response by the system might be:

R. Treasury Bills do not have a penalty. Were you thinking of a Treasury Bond?

As the examples show, changes in the active perspective can account for the same misconception being responded to in two different ways.

13. Conclusions

This paper has described a new method for responding to misconceptions that relies on an analysis of a highlighted user model to generate a response that is most likely to benefit the user. A number of strategies were abstracted from a study of transcripts. Each strategy was associated with a "distinguished structure" in the

highlighted user model which could explain its use in a given situation. A system can use this pairing to decide how to respond by looking in the highlighted user model for evidence of one of the "distinguished structures". The corresponding strategy can be used to respond.

In addition, the paper has described a new notion of object perspective which is able to model one aspect of the dynamic highlighting of the user model due to previous discourse. It was shown how perspective could account for certain contextual effects on responses to misconceptions. In particular it was shown how the highlighting due to perspective could cause the Tversky similarity metric to return different values for the same objects. Since object similarity occurs in all of the "distinguished structures" of the highlighted user model, the perspective taken on the domain influences the response strategy used in correcting misconceptions.

This work has proceeded from the perspective of responding to a misconception given a detailed user model. At the same time, it could be looked at as a potential method for inferring user models. User model configurations that support a misconception have been identified. If the user exhibits a misconception, parts of the user model might be inferred by attributing to the user some of the pre-identified supporting beliefs. More work needs to be done in order to determine the feasibility of this approach.

14. References

[BOBR77], [BROW78], [CARB84], [CARB*], [CARN70b], [CHIN*], [COHE*], [FINI*], [GROS77b], [GROS81], [GROS83], [GROS85], [KAPS79], [KARL85], [MAYS80a], [MCCO85a], [MCCO87], [MCCO88], [MCDO80], [MCKE82], [MCKE85c], [MORI*], [QUIL*], [RICH*], [RUME73], [SIDN83a], [SLEE82b], [STEV79], [STEV80], [TOU82], [TVER77], [WEIN84], [WOOL83], [WOOL84a]

10 But What Will the Listener Think? Belief Ascription and Image Maintenance in Dialog

Anthony Jameson

Abstract

If the users of dialog systems which employ user models are to interpret the system's behavior accurately, they must have some understanding of the content of the user model and the purposes for which the system is exploiting it. This fundamental issue has received little attention in user modeling research, but the users and designers of such systems can learn much from their own experience in everyday dialog. Listeners know that the form and content of a speaker's utterance depends on how the speaker assesses the listener's goals and beliefs, and also on the speaker's own dialog motivation. Listeners thus take these factors into account when interpreting utterances, and in particular their interpretations may be incorrect if their assumptions about the speaker are inaccurate. Consequently, speakers often make special efforts to control the image that the listener has of their dialog motivation and their model of the listener. In this chapter, these phenomena are analyzed systematically and illustrated with a simulation model for a restricted type of everyday dialog.

1. Introduction

As the title suggests, this paper was originally motivated by an interest in everyday human dialog. Nonetheless, the issues it addresses represent fundamental and little-studied problems for those who are mainly interested in dialog systems which employ user models. In terms relevant to these goals, the main theme is the importance of the way the users of such systems perceive (a) the content of the user model and (b) the dialog motivation of the system that exploits it.

This theme is an instantiation of a general theme in the literature on human-computer interaction. Here attention has often been drawn to the mental model, or **user's model**, that a user forms of an interactive system (see e.g. [NORM86a, pp. 45-48]). The user's ability to learn to work with a system effectively and to interpret its behavior correctly is said to depend on his or her understanding of the way the system works, though this understanding may be metaphorical rather than technical and literally accurate. Accordingly, designers have often been advised to aim for a system of which a reasonably simple but useful user's model can be formed, and to help the user in various ways to acquire such a model.

Although there has not yet been much empirical research on these ideas, they correspond with general experience and are thus worth taking seriously in designing

dialog systems which include a user modeling component. We will now consider, in turn, two aspects of such a system which are more problematic for the user to perceive appropriately than is generally thought.

1.1. The User's Model of the User Model

It is sometimes suggested that users will simply expect a user model to be complete (in the relevant respects) and accurate (this appears to be presupposed, e.g. by [CARB*, JOSH82, JOSH84a]). If this expectation were justified, it might be unnecessary for users to have any specific knowledge about a system's user model. But even brief inspection of the chapters in this book reminds us that this expectation is unrealistic. Except for the most straightforward cases, user modeling is an inherently speculative and error-prone enterprise, so that in turn users will often only have some notion of what the system thinks about them if they are aware of the system's level of user modeling competence and of the particular strategies employed. It is thus worthwhile to consider how easy this would be for the users of systems which employ various specific techniques. For example, how likely is it that users could anticipate what stereotypes (e.g. level of expertise) were being applied to them, and at what time, by a system using the techniques described by Rich, Finin, Chin or Paris [RICH*, FINI*, CHIN*, PARI*]? Or that they could guess what conclusions about their desires, goals or plans were being inferred from their questions by a system using Morik's isotopy principle [MORI*] or the strategies described by Carberry [CARB*]?

The answers to these questions, and the importance of their implications, are not the same for all of these techniques and systems, but it is in any case clear that it is not always an easy task for the user to form a definite and accurate picture of the system's user model.

1.2. The User's Model of the System's Dialog Motivation

In everyday human dialog, the way in which a listener interprets a speaker's utterance can depend strongly on the motivation that the listener ascribes to the speaker, e.g. whether the listener is being straightforward or diplomatic, cooperative or self-serving. A commonly held view of the role of these factors in human-computer communication is expressed as follows by Maclaran [MACL82, p. 215]:[1]

> It is especially interesting to apply Gricean principles to man-machine interaction since here the goal of co-operative behaviour is more nearly realized: we do not have to worry about deceit, politeness or any of the other interfering factors in man-man communication.

Here again, a critical look at this assumption quickly reveals it to be unjustified.

First, even if the user knows that a system is designed so as to be generally cooperative, he or she may not know what its specific priorities are. For example Shneiderman, a leading advocate of user-oriented interface design, makes several

[1] See also the views cited by Winograd & Flores [WINO86, pp. 156f].

recommendations concerning the formulation of error messages, including the recommendation that they should avoid placing the "blame" for an error on the user [SHNE87, pp. 312-322]. This specific goal can at times conflict with the equally cooperative goal of informing the user precisely and accurately as to what has gone wrong. Thus if a system assigns exceptionally high priority to the former goal, users may be misled by its error messages – unless they realize that the system is "just being polite". (This could happen, for example, if the user was being exceptionally careless in the way he or she entered commands and the system continually apologized for being "unable" to parse them.)

Second, systems may be equipped with dialog motivation which is at least partly noncooperative, even if outright deception is avoided. This is rather obvious for systems which are consulted by occasional users for information or advice and whose owners have an interest in presenting a biased picture. But it can also happen with commercial software packages acquired as tools by individual users. No one expects the advertising brochures or even the printed documentation for such systems to volunteer helpful warnings about the system's unnecessary and irritating limitations or to provide welcome advice on how to get around the copy-protection mechanism; why should one expect such behavior from the on-line tutorial or the intelligent help package that is delivered with such a system? Here again, users can be misled if they are unaware of the system's motivation, which is less predictable in some cases than in others.

In these examples, one should actually speak of the motivation of the system's designers rather than that of the system itself. But in dialog systems as sophisticated as most of those described in this book, the nature of the system's behavior is not determined directly by decisions of the designer; it depends, among other things, on goals which can vary from one application, situation, or user to the next. It is therefore appropriate to think in terms of the motivation of the system itself.

1.3. Approach Taken in This Paper

The above discussion has had to be rather futuristic and hypothetical, since little experience has been acquired to date with the use of systems which employ user models or whose motivation raises complex questions. But fortunately, all of us have extensive relevant experience as participants in everyday dialog. When interpreting the utterances of others, we are accustomed to taking into account not only their dialog motivation but also their model of us as listeners. When generating utterances ourselves, we are aware that they will be interpreted in this way. It is thus appropriate to begin the study of these processes with a systematic analysis of the corresponding phenomena in everyday human dialog. The implications of this analysis for human-computer communication will be considered in the concluding section of this chapter.[2]

Although the phenomena of interest are very general, it is necessary to restrict our attention to a particular type of dialog, so that the analysis does not become

[2] A similar strategy is applied by Thomas (e.g. in [THMA83, pp. 189-192]), in his study of the role of dialog motivation in the use and interpretation of English quantifiers.

excessively abstract, and so that a simulation model can be used to help explore and illustrate all of the main ideas. It is best to choose a type of dialog in which the nature of the speaker's model of the listener can be simpler than in most of the chapters in this volume, and in which contrasting dialog motivations of the speaker are easy to conceptualize.

These conditions are fulfilled by what may be called **evaluation-oriented dialog**, in which the two participants are an **Evaluator** and an **Informant**. (To facilitate exposition, we arbitrarily assume that each Evaluator is male and each Informant female.) The first defining characteristic of this type of dialog is the Evaluator's main goal: to evaluate a given **Object** – which may be, e.g., a thing, a person, or a course of action – according to certain criteria. The Evaluator may want to use his evaluation as the basis for some decision (e.g. if he is a customer deciding whether to purchase a particular product); or he may simply be curious about the Object (as when he is asking a friend whether the party she went to last night was enjoyable).

The Evaluator can ask questions of the Informant, whose main dialog goal is to provide him with some of the information that he needs to apply his evaluation criteria. The Informant makes purely veridical factual statements. In particular, she does not express evaluations of the Object herself but rather leaves it to the Evaluator to determine the evaluative implications of the facts that she presents. Similarly, the Informant may not make any utterance that serves some other possible goal of the Evaluator, e.g. she may not offer advice as to how the Object might be improved.

In many practically important situations, the Object is some aspect of the Informant herself. For example, if the Evaluator has just given a public talk and asks a colleague (the Informant) what she thought of it, he may want to use his colleague's responses not to evaluate his talk but rather to evaluate her personal reactions to it. In a personnel interview, the interviewer or potential employer takes the role of the Evaluator, while the job applicant is simultaneously the Informant and the Object which is being evaluated. It is this specific type of dialog that will be used in the examples in this chapter.

Since this paper is mainly concerned with the way the Informant decides what to say to the Evaluator, the Informant will usually be referred to more simply as the **speaker** (or **S**) and the Evaluator as the **listener** (or **L**).

The theoretical ideas in this paper were developed in parallel with a dialog system called Imp which takes the role of the Informant in an evaluation-oriented dialog.[3] This simulation model will be referred to repeatedly to illustrate the main points. The system is not restricted to any particular topic area, but it does have to be supplied in each case with a good deal of domain-specific information, to be described below. Imp was originally developed and tested in a situation where an Informant was offering a room for rent to the Evaluator [JAME83]. For the examples used in the present paper, the domain-specific information was supplied by a young woman who had recently been interviewed for a position as typesetter in two small printing firms.

---

[3] The name **Imp** is due to the centrality of the notion of an **impression** in the simplest versions of the system.

1.4. Structure of the Chapter

The simulation model helps us to think about the way in which many diverse factors jointly determine a speaker's behavior – factors which would typically be examined individually in a more traditional psychological or linguistic study. In the interest of clarity, I will begin by discussing an especially simple situation in which very few of these factors are relevant. Then eight further situations will be defined, each of which introduces one or more new complications. In Situations 2 through 6, these complications involve mainly the speaker's dialog motivation and the way the listener takes it into account when interpreting the speaker's utterances. In Situations 7 through 9 they concern the way the speaker's model of the listener is perceived by the listener. It is not until Situation 5 that an actual dialog will take place, since the most basic concepts are easiest to introduce if the speaker simply produces a monolog.

In papers which include discussions of computer simulation models of human processing, it is often difficult for the reader to distinguish between (a) general normative or descriptive claims about how people (should) handle certain tasks and (b) details of a specific system which are either arbitrary or dictated by technical constraints. To help make explicit the status of the various points made in this paper, and also to facilitate selective reading, I will discuss each of the nine situations in a standard sequence of six subsections with the following headings:

1. Definition of the Situation
2. How the Situation Can Arise
3. Appropriate Strategies
4. Realization in the Simulation Model
5. Resulting Behavior
6. General Issues Raised

The first subsection lists several postulates, mainly about the speaker's beliefs and dialog motivation. Taken together with the postulates of all previous situations, they characterize the situation to be analyzed. These postulates are intended to define useful idealizations of situations that occur in actual dialogs.

Subsection 2 sketches some of the circumstances in which a situation approximated by the definition can occur. These remarks are not systematic or complete; they are intended only to suggest the range of applicability of the analysis and to help the reader call to mind relevant experiences.

The third subsection lists some appropriate strategies for a speaker in the given situation. The normative claim is that it is reasonable for S to apply these strategies when pursuing the postulated goals. The descriptive, or empirical claim is only that at least some people sometimes apply these strategies. Any stronger claim, even in terms of average behavior, would in many cases be false or misleading. For example, as Sampson [SAMP82] reminds us, some speakers consistently neglect even the most elementary appropriate strategies discussed in the early situations. Since I assume that the reader's own experience as a dialog participant will provide abundant examples of the phenomena discussed, I will not quote specific observed

examples of dialog behavior (as do, e.g., [CARB*, MCCO*, PARI*, QUIL*] in this volume).

Subsection 4 sketches the representational techniques and procedures used to implement the strategies introduced in Subsection 3 in the simulation model.[4] No normative or empirical claim is made about any additional details presented in this subsection, and in fact the implementation requires many arbitrary decisions and simplifications which are clearly inapplicable to human processing. Since many such decisions are reflected in the various figures which illustrate how the system works, these figures should be viewed not as summarizing the theoretical points made in the other subsections but rather as depicting metaphors which may facilitate thinking about these points. In particular, no attention is given to the many possible shortcuts and heuristics which would be needed for efficient real-time application of the strategies in question. Readers with no particular interest in how the simulation model works can safely skip this fourth subsection.

The fifth subsection lists the main consequences of the application of the strategies introduced in the third subsection. These consequences are illustrated with examples of the simulation model's performance, and in fact some of them became apparent to me only through experimentation with the model and discussion of its behavior with others. Once noticed, however, all of the consequences mentioned can be seen to follow generally from the "appropriate strategies"; features of the examples which merely reflect implementation details receive no attention.

The final subsection for each situation comments on questions which the preceding subsections called to mind but did not address, and in some cases it gives pointers to research which does deal with these questions.

The entire analysis is made from the point of view of the speaker, and in principle all statements about the dialog should begin with "S believes ...", e.g. "S believes that L will misinterpret her comment". Since this type of formulation can be extremely cumbersome, especially in the more complex situations, I will often write, e.g., "L will misinterpret S's comment", leaving it to the reader to bear in mind that reference is actually being made to a belief of the speaker.

2. Situation 1: Generating an Informative Monolog

2.1. Definition of the Situation

It will be assumed in the first four situations that the speaker in effect has the opportunity to generate an unconstrained monolog about the Object. For example, a potential employer, as listener, might say to a job applicant "Tell me about yourself." In this first situation, we assume that the speaker is motivated to produce in the listener an unbiased and sufficiently definite picture of the Object. The only

[4] The system is implemented in the Lisp-embedded object-oriented language Orbit designed by Koenraad De Smedt, but it will not be necessary to refer to any specific properties of this language.

reason why she might withhold any information is that mentioning it did not seem to be worth the effort.

2.2. How the Situation Can Arise

These assumptions are actually less characteristic of a face-to-face employment interview than of a case where S is an impartial reference writing a confidential letter of recommendation. But even in an interview, the candidate might have this sort of cooperative motivation if, for example, she only wanted to be hired if she were truly the best candidate.

2.3. Appropriate Strategies

2.3.1. Necessary Assumptions of the Speaker

Even in this simple situation, in order to behave appropriately, the speaker must be able to make some reasonable assumptions about the **evaluation standards** of the listener. Specifically, S must be aware that there are various aspects of the Object that L is concerned about (e.g. the candidate's reason for leaving her previous job). For each such aspect, or **item**, there are various **possibilities** which might be true of the Object (e.g. different types of reasons for leaving a job). L presumably not only considers some of these possibilities more desirable than others but also has some expectation as to how likely they are to be found in the Object.

S must therefore try to guess what items, possibilities, values, and likelihoods L has in mind. How she should do this is a complex question which will not receive much attention in this chapter (although it will be mentioned in connection with Situations 7 through 9).

In order to be able to volunteer information relevant to a given item, the system must have an idea of what **comments** might conceivably be made which are germane to that item, and what implications they have. One in general has the option of making a vague comment on an item, which leaves open several possibilities that L would consider worth distinguishing, or a specific comment which tells L all that he might like to know for the purpose of evaluating the Object. The reason why cooperative speakers do not always make the latter type of comment on all items is that any comment requires a certain amount of effort for the speaker to produce and for the listener to interpret, and precise comments often require more effort than vague ones. Thus although the concept of the **effort** associated with various comments is an elusive one,[5] S must have some way of taking this consideration into account.

2.3.2. General Strategy

A reasonable general strategy is to consider each relevant item and to decide what, if anything, is worth saying about it. That is, S must choose for each item, from

[5] Discussion of some of the factors involved can be found, e.g., in [GAZD79, chap. 3, GRIC75, MCCL78, WINO77].

among the possible comments for that item which are true of the Object, the one comment which is the most worth making, i.e., the one whose benefits most clearly justify the amount of effort involved.

2.3.3. Assessing the Potential Benefits of Making a Comment

What are the benefits which in this situation can result from the production of an informative comment? It has often been pointed out that the value of a statement to the listener can depend on both (a) the extent to which L would have suspected that it was true without being told and (b) the consequences for L of its being true rather than false (see e.g. [GREW83, HARR63, chap. 4, HINT75]). It is in general difficult to use these criteria to arrive at a precise, quantitative definition, but this can be done quite easily within evaluation-oriented dialog. The key concept is that of an **impression**. At any given moment and for any given item, the listener can be said to have an impression of the Object with respect to that item. The nature of this impression depends on which possibilities the listener (still) thinks might be realized by the Object. Before the speaker has said anything about a given item, the listener's impression concerning it will be relatively indefinite, since all of the possibilities must be considered (although some are less likely than others). If S makes a certain comment concerning the item, L can in general eliminate one or more possibilities, so that the impression will become more definite. It will in general also become somewhat more or less **favorable** than it was before, depending on whether the possibilities eliminated were among the less or more desirable ones.

There are at least two types of impression change that can be desirable for a speaker in this situation. First, S should consider it desirable if a comment makes the impression with respect to an item more definite, since this reduces L's uncertainty about the evaluation of the Object with respect to that item (and thus overall). But since even a fairly definite impression may be considerably more or less favorable than justified by the truth about the Object, it is also desirable that a comment reduce, or at least not create, such a discrepancy.

The desirable consequences of any given comment must somehow be integrated and weighed against the effort required, so that the **net desirability** of the comment can be determined. The comment with the greatest net desirability can then be verbalized; or if no comment seems worth the effort, S should say nothing about the item in question.

To recapitulate: for each evaluation-relevant aspect of the Object, S should verbalize the comment (if any) whose prospective benefits, in terms of an impression change in L, most clearly justify the effort associated with it.

2.4. Realization in the Simulation Model

A convenient representation for L's evaluation standards is a data structure which may be called an **evaluation form**. This representation is reminiscent of the sort of rating form which is often found in practical situations for the systematic evaluation of objects or persons (e.g. papers submitted to conferences, or applicants to universities). In Imp the evaluation form consists of a set of **items**, each of which

Item 9: Family Situation

Possibilities:

		Value	Probability
A.	Married without children.	-35	.15
B.	Married with children.	-20	.05
C.	Living together.	-10	.35
D.	Living apart.	0	.25
E.	No steady relationship.	10	.20

Formulations:

	Possibilities left open	Effort
I am married.	A B	50
I am not married.	C D E	50
I have <number> children.	B C	50
I don't have any children.	A D E	50
I live alone.	D E	50
I have a boyfriend.	C D	75
I am married and I don't have children.	A	100
I am married and I have <number> children.	B	100
I am living with my boyfriend.	C	100
I have a boyfriend, but we don't live together.	D	100
I don't have any steady relationship.	E	100

Figure 1. An item from the evaluation form assumed by the Informant of the examples.

concerns some evaluation-relevant aspect of the Object. Twelve items were supplied by our example Informant to characterize the assumptions she made during the first of her two interviews. One of these is reproduced in Figure 1.

Each item is a sort of multiple-choice question accompanied by a scoring key. An item specifies several conditions, or **possibilities**, which are defined in such a way that exactly one of them must be realized by the Object. With each possibility, two numbers are associated: (a) a **value**, i.e., a number of points which may be assigned to the Object if it satisfies the condition; and (b) a **probability**, i.e., an estimate of the a priori likelihood that the Object will satisfy that condition. The item shown in Figure 1 reflects, among other things, the speaker's belief that her prospective employer would be disappointed to hear that she was married, though less so in the (unlikely) case that she already had children.

If the listener were actually to use an evaluation form to evaluate the Object, he would attempt, for each item, to determine which of the possibilities was realized by

the Object. If he succeeded, he would add the corresponding value to his rating of the Object. If the information available was not sufficient to rule out all possibilities but one, he would use the probabilities specified for the remaining possibilities to estimate how many points the Object deserved for the item.[6] Imp in effect acts as if it expected that the listener would use the information it provided to fill in the evaluation form.

The system possesses no methods for formulating entirely new comments on its own. It must be provided, for each item, with a list of possible comments which are representative of the infinite number of comments that might conceivably be made. For example, Figure 1 shows a number of things that can be said about a female applicant's family situation. For each comment, Imp must be told which possibilities are consistent with it on a straightforward interpretation and how much effort is associated with it. These numerical ratings of effort are impossible to justify in detail, but for our purposes it is sufficient if the major differences among comments are reflected in different numerical ratings.

It is convenient to view **silence** as a special sort of comment which can be made about any item, which (when interpreted straightforwardly) is consistent with all possibilities, and which involves no effort.

To be able to quantify the desirability of a given impression change, Imp operationalizes the concept of an **impression** in terms of a probability distribution, determined by the values and the probabilities associated with the possibilities which have not yet been eliminated from consideration. Figure 2 shows graphically the impression that the listener is assumed to have initially concerning the "Family situation" item. It also shows the impressions that would remain after each of two possible comments that the speaker might make.

The idea of how favorable an impression is can now be captured in terms of the mean, or **expected value** of the probability distribution in question; for example, in Figure 2 the first comment rules out the three least favorable possibilities, and as a consequence the expected value rises; it would actually rise more sharply if it weren't for the fact that the two least favorable possibilities are considered relatively improbable to begin with.

The second important property of an impression, the degree of uncertainty associated with it, can be quantified in terms of the standard deviation of the probability distribution. In general, the greater the number of possibilities that have not yet been eliminated, the greater the degree of uncertainty associated with an impression (this is illustrated in Figure 2); but the standard deviation captures this concept better than a mere count of the remaining possibilities would, since it is sensitive to the relative likelihoods of the remaining possibilities and the differences in their values.

---

[6] This way of using an evaluation form only makes sense if the items are independent of each other, in the sense that knowing what possibility is realized for one item cannot affect one's assessment of the probabilities associated with any other item. A simple way of avoiding such dependencies is to mention within a single item any properties of Objects which are substantially correlated. This is why, for example, the distinctions "married vs. single" and "children vs. no children" are not captured in separate items. Rowe [ROWE84] presents a much more sophisticated, practically oriented technique for evaluating objects from the point of view of the user of a database query system.

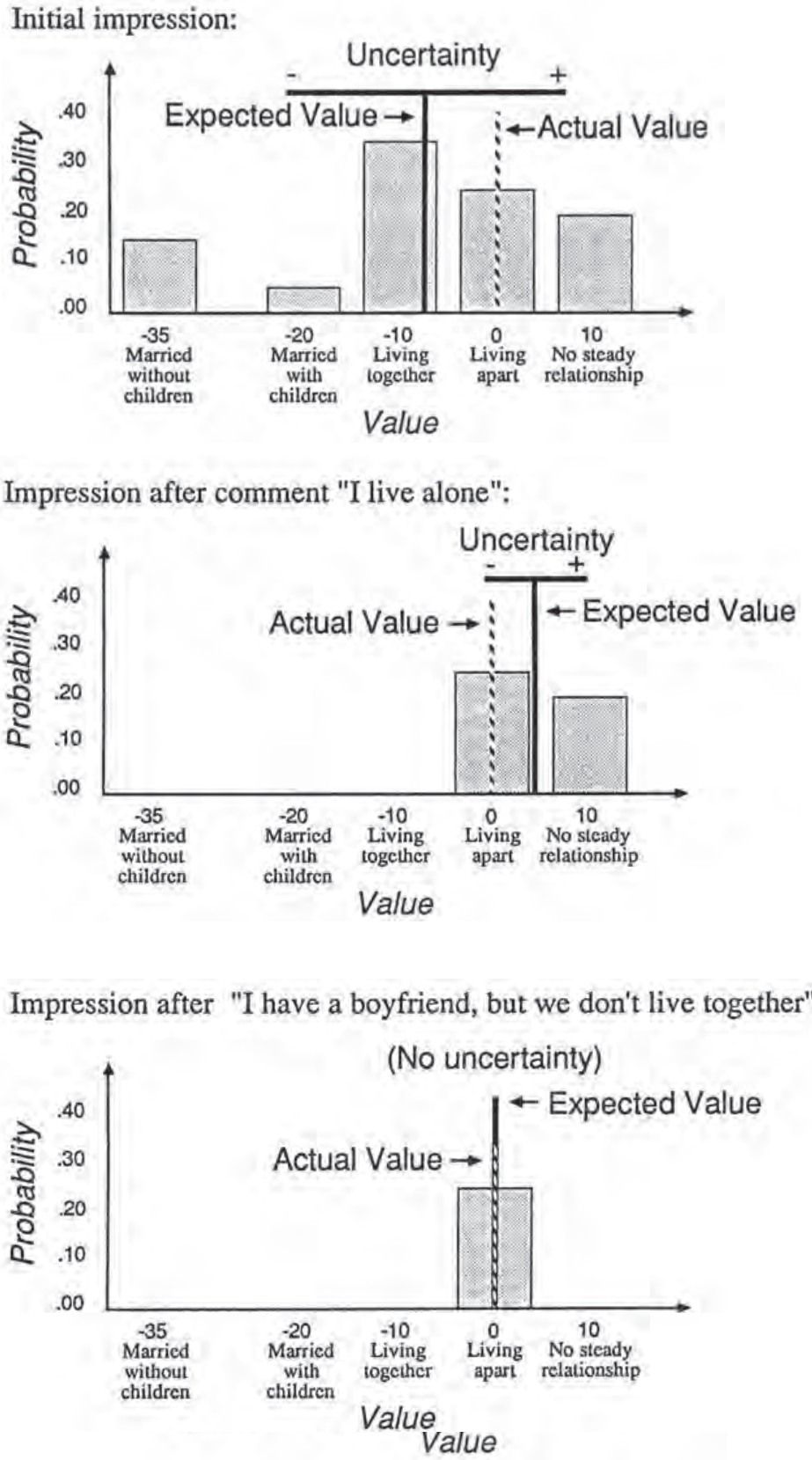


Figure 2. Illustration of the concepts of an **impression** and its associated **expected value** and **uncertainty** (see discussion in text).

The two properties of an impression change – reduction in uncertainty and reduction in discrepancy from the actual value of the Object – are largely independent consequences which can be considered desirable. Imp simply takes the sum of these two quantities to determine the total benefits which can be expected to result from a given comment. Thus in Figure 2, the second comment produces somewhat greater total benefits than the first one does, since it reduces all of the uncertainty and all of the discrepancy between the expected value and the true value. But it is likely that a speaker might prefer the first comment to the second one on the grounds that the additional benefits produced by the second one do not justify the additional effort involved.

In finally deciding which comment to make on a given item, Imp subtracts the rating of the effort associated with each possible comment from its total expected benefits to yield a quantification of the net desirability of the comment (cf. Figure 3). A "talkativeness" parameter is also required to take into account both the individual differences in the general willingness of speakers to make comments and the arbitrariness of the units used to quantify the amount of effort associated with a comment.

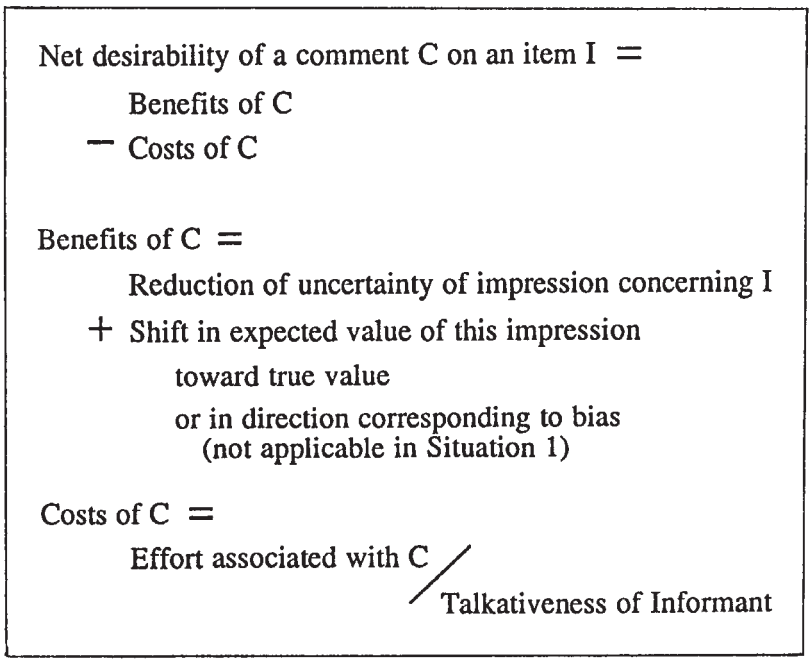


Figure 3. Formulas used by Imp in Situations 1 through 4 to determine how worthwhile it is to make a given comment.

The comment with the highest net desirability is the one which is actually made. In many cases the most desirable comment will simply be silence (which, as can be seen from the above definitions, always has a net desirability of zero), since

the benefits that would be brought about by the other possible comments are less than the associated cost in terms of effort.

2.5. Resulting Behavior

The first two columns of Figure 4 show what comments Imp volunteered in this situation on each of the twelve items in the evaluation form. The way in which all of the comments are connected into a single long sentence, which is rather unnatural in this situation, will be discussed in connection with Situation 5. The following general tendencies result from the application of the strategies sketched above:

1. On items which are exceptionally important – i.e. about which the listener experiences a great deal of evaluative uncertainty – the speaker will in general make some comment even if the truth about the Object is not at all unusual.

With such an item, the effort required by a comment may be justified by the reduction in uncertainty that results, even if L's initial impression is not inappropriately favorable or unfavorable. The system's first comment in Figure 4 provides an example: a year and a half of work experience is a normal amount for this type of applicant, but it is worthwhile to reassure L that the alternative possibilities, some of which would be much more or less favorable, are not realized.

2. Moderately important items will in general be commented upon only if the truth is either exceptionally desirable or undesirable.

The second and third comments mention the most disappointing possibilities for their respective items, thus making L's vague initial impressions not only more definite but also more in line with reality. This second benefit is necessary here to warrant the generation of a comment, since the reduction of L's modest amount of uncertainty would not be a substantial benefit. The same is true for the comments on Items 6 and 7, except that these comments are exceptionally favorable. On Items 10 and 12, S would say something if the truth deviated sharply from L's expectations – e.g. if the applicant lived in some distant town; but since the extreme possibilities are improbable, L will not worry about these items if nothing is said about them.

This tendency for extreme possibilities to be mentioned relatively frequently is strengthened by the fact that intermediate possibilities are often more awkward to describe. Thus it requires less effort to say that one's previous work was "very challenging" or "purely routine" than to characterize it as "sometimes pretty challenging, but usually not".

3. There may be some items about which the speaker would say nothing no matter what the truth was.

With topics such as the degree of variety of the applicant's previous work experience (Item 2), there is no possibility which is so desirable or undesirable as to be worth mentioning (unless L asks about the topic specifically, as in Situation 5).

2.6. General Issues Raised

The goals of correcting inaccuracies and reducing uncertainty make sense for a cooperative speaker in many types of dialog situation. For example, if a speaker is

Item	Comments Chosen		Pragmatic Interpretations Expected (after comments in third column)	
	Situation 1 (objective bias)	Situations 2 - 4 (positive bias)	Situation 3 (positive projected bias)	Situation 4 (objective projected bias)
1. Amount of work experience	I'VE BEEN WORKING FOR ABOUT A YEAR AND A HALF	I'VE BEEN WORKING FOR MORE THAN A YEAR	1. Between 1 and 2 years experience.	[Comment is inconsistent with projected objectivity.]
2. Variety of work experience			2.	
3. Complexity of previous work	BUT MY WORK HAS BEEN PURELY ROUTINE		3. Not especially challenging work.	Not especially challenging work.
4. Vocational training	AND UNFORTUNATELY I NEVER RECEIVED ANY VOCATIONAL TRAINING		4. No higher-level vocational training.	No higher-level vocational training, but some training.
5. Work-related hobbies			5.	
6. Reason for changing jobs	BUT THE REASON I WANT TO LEAVE MY PRESENT JOB IS THAT I'M LOOKING FOR MORE CHALLENGING WORK	AND THE REASON I WANT TO LEAVE MY PRESENT JOB IS THAT I'M LOOKING FOR MORE CHALLENGING WORK	6.	
7. Nonvocational schooling	AND I WENT TO UNIVERSITY PREPARATORY SCHOOL	AND I WENT TO UNIVERSITY PREPARATORY SCHOOL	7.	
8. Plans for future education			8.	
9. Family situation			9.	
10. Place of residence			10.	Does not live especially far away.
11. Frequency of previous job changes			11.	
12. Nature of position ultimately desired			12.	Is not aiming for position unattainable in this company.

Figure 4. Monologs produced by Imp in Situations 1 through 4.

giving the listener advice on how to perform a certain task, it may be worthwhile not only to prevent L from performing inappropriate actions but also to reassure him when his intended action is in fact appropriate. The relative weight that S should attach to these two goals presumably varies from one specific context to the next.

The way in which the key theoretical concepts in this section are realized in the simulation model is dictated largely by a desire to keep the system simple, but the operationalizations are reasonably consistent with the way people actually evaluate objects, according to relevant empirical evidence. For example, studies reported by Wyer [WYER74, chap. 2] can be interpreted as supporting the ways in which the concepts of **expected value** and **uncertainty** are operationalized. On the other hand, research within the paradigm of Information Integration Theory (see e.g. [ANDN81] for a brief summary) has shown, for example, that people often evaluate an Object less favorably if they hear one extremely positive and one moderately positive fact about it than if they hear only the extremely positive fact. This result is incompatible with the evaluation-form metaphor; but note that this is not a problem for the present analysis, since presumably few speakers are aware of this sort of phenomenon.

Persons who witness demonstrations of the Imp system often remark that they find a particular utterance implausible in the sense that they, as speakers, would never have produced such an utterance. In some instances reactions of this sort have exposed inadequacies in the theory underlying the simulation model and have led to revisions of both. But in most cases it turns out that the observer implicitly ascribes different specific values and expectations to the Evaluator than those which have been fed into the simulation model: when the observer is shown the relevant part of the evaluation form used (i.e. the sort of information given in Figure 1), the system's behavior seems reasonable after all. In short, this problem in evaluating the system's performance illustrates one of the general points made in the Introduction: It can be difficult to interpret the behavior of a user modeling system if one does not know the contents of the user model. In Situations 7 through 9, the simulation model will be sophisticated enough to take this difficulty into account.

3. Situation 2: Positive or Negative Bias

3.1. Definition of the Situation

In this situation the speaker is not necessarily attempting to be as cooperative as possible but rather may have the goal of presenting the Object to the listener in an especially favorable or unfavorable light. As before, however, S restricts herself to strictly truthful factual comments which are not explicitly evaluative.

3.2. How the Situation Can Arise

Bias is of course not found only in contexts, such as the employment interview, where it can produce tangible benefits for the speaker. The creation of a favorable (or unfavorable) impression in the listener may be seen by the speaker as a desirable end in itself.

Negative bias is even more improbable than objectivity in an employment interview, but it might be found, e.g., if the candidate didn't want the job at all but was obligated to apply in order to continue to qualify for unemployment benefits.

3.3. Appropriate Strategies

Within the constraints of this situation, positive or negative bias on the part of S can only affect her criteria for selecting comments from among the set of possible truthful comments about the Object. Specifically, it is now no longer necessarily desirable to eliminate discrepancies between the listener's impressions and the truth about the Object; instead, the individual impressions should be made as favorable (or unfavorable) as possible. The other considerations discussed in the previous situation, e.g. the desirability of reducing L's uncertainty, are still relevant, though they may be given different weights relative to the first consideration.

3.4. Realization in the Simulation Model

The Imp system includes a parameter **bias** which can take the values **objective** (as assumed implicitly in Situation 1), **positive**, or **negative**. This parameter determines how possible comments are rated in terms of their expected impact on the listener's impressions. As shown in Figure 3, when the bias is positive or negative the important aspect of the shift in the expected value of an impression is now the extent to which it is in the positive (or negative) direction. Thus an impression shift toward the true value is actually undesirable if it doesn't happen to be in the desired direction.

3.5. Resulting Behavior

For concreteness, the consequences of bias will be described for the case where S's bias is positive, as in the examples. Because of the symmetric relationship between positive and negative bias, the consequences of negative bias are easy to derive.

1. The speaker now seldom volunteers a comment which would cause an impression shift in the negative direction.

As can be seen in the third column of Figure 4, the comments made in Situation 1 on the routine nature of the speaker's previous work and her lack of vocational training are not made in this situation. Their net desirability is now less than that of silence: they eliminate all uncertainty, but the way they shift L's impressions downward is now considered undesirable. The net desirability of the comment is

therefore small or even negative, and it certainly does not justify the effort of making the comment.

2. The speaker will sometimes choose a vague formulation which leaves open one or more possibilities that are more favorable than the one actually realized, even though a more precise comment would involve no more effort.

This is illustrated by the comment on the first item in Figure 4. The statement that the candidate has been working for "more than a year" leaves open the possibility that she might have several years of experience and thus leaves an impression which, though vague, is considerably more favorable than a definite and accurate impression would be. To be sure, the listener will probably not be so naive as to take this statement at face value; the reasons for this will be examined in the next situation.

3.6. General Issues Raised

The phenomenon of bias is obviously not restricted to contexts in which some Object is being evaluated. Behavior analogous to that described above can be found, e.g., in the attempt of a lawyer to lead a jury to a particular substantive conclusion. By abstracting away from almost all problems directly involving natural language, the above analysis focuses attention on simple selectivity as a mechanism of bias, as opposed, say, to rhetorical techniques involving particular expressions (cf. [HOVY86]).

Whatever particular benefits the speaker may expect from a biased selection of information, they must somehow be assessed and integrated with other factors such as the extent to which the listener's uncertainty is reduced and the amount of effort required to make one's comments. Biased speakers may tend to attach different weights to the various considerations than objective speakers do. For instance, they may attach less weight to the goal of reducing uncertainty, aiming almost exclusively for impression shifts in a particular direction. Another possibility is that such a speaker may increase her general talkativeness in order to compensate for the fact that she is, because of her selectivity, producing fewer utterances.

4. Situation 3: Anticipating the Pragmatic Interpretation of Comments

4.1. Definition of the Situation

When anticipating the impact that a given comment would have on the listener, the speaker has up to now assumed that each comment would be taken at face value: the only possibilities which the listener was expected to rule out were the ones which were incompatible with a straightforward interpretation of her comment. But in fact listeners often rule out further possibilities by noticing that the speaker has chosen not to say certain things which she might well have said. It will be

assumed in this situation that the speaker is sophisticated enough to realize that her comments will be interpreted in this way.

4.2. How the Situation Can Arise

Whenever a listener has a reasonably clear idea of the criteria and knowledge a speaker uses in generating utterances, he can consider what S would have said in various circumstances and thus extract information from S's utterances which goes beyond their conventional implications.

4.3. Appropriate Strategies

The basic reasoning that L can use to interpret the fact that S did not say certain things is summarized in the following schema: "Possibility P is not ruled out by the speaker's comment C [which may be silence]; but then again, if P were the possibility that was realized, S would surely have said something like C' instead of C, since C' would have had a more desirable impact on my impression; therefore, P is apparently not realized." If L uses this schema, perhaps ruling out one or more of the possibilities allowed by a comment, we will say that he has made a **pragmatic interpretation** of that comment.[7]

S must anticipate L's pragmatic interpretations of her comments if she wants to maintain an accurate picture of L's impressions of the Object. (It will become increasingly important for S to keep track of L's impressions in the more complex situations to be examined below.)

4.4. Realization in the Simulation Model

Figure 5 summarizes the two procedures that Imp uses to generate comments (some parts of the procedures will only become applicable in Situation 4). As can be seen at the beginning of the first procedure, Imp begins by selecting the most desirable comment, using the same criteria as in the previous situations. Before verbalizing this comment, however, it now uses the second procedure to determine what L will be able to infer by applying the pragmatic interpretation schema given above.

There is one aspect of this procedure for anticipating pragmatic interpretations that will not be discussed in detail until Situation 8, although it does have some effect on Imp's behavior even in the earlier situations: the specification (in Step 2), that a possibility should be ruled out only if it is "clear" to L that S would have made some other comment if that possibility were realized. This condition becomes

---

[7] This concept is more general than Grice's conception of the way in which a conversational implicature is "worked out" by the listener [GRIC75, p. 50]. The notion of conversational implicature presupposes, among other things, (a) that the speaker is observing the cooperative principle, and (b) that the interpretation made by the listener depends on L's recognition of S's expectation that L will make that interpretation. By contrast, a pragmatic interpretation as defined here could be made by a listener who was eavesdropping on a speaker who was being blatantly uncooperative.

To volunteer a comment on a given item:

1. Find the comment on that item with the greatest net desirability [cf. Figure 3] (given your bias and the evaluation standards you ascribe to L).
2. Anticipate L's Pragmatic Interpretation of this comment [see below] (assuming that he has the image of you that you want to maintain).
3. [Applicable in Situations 4 - 9:]
   If this pragmatic interpretation is empty [i.e., the comment is incompatible with your image], then
   Reject this comment.
   Return to Step 2, using the comment with the next greatest net desirability.
4. Record L's pragmatic interpretation and verbalize the comment.

To anticipate the pragmatic interpretation of a comment C on a given item (given a certain image that you want to maintain):

1. Consider the pragmatic interpretation initially to be the set of all possibilities for the item that are consistent with a straightforward interpretation of the comment.
2. For each such possibility P,
   Is it clear to L that some alternative comment C' would be considered more desirable than C by a speaker who corresponded to your image, if P were realized?
   If so, eliminate P from the pragmatic interpretation.
3. Return as the pragmatic interpretation the set of possibilities that remain. [In Situations 4 - 9, this set may be empty.]

Figure 5. Two of Imp's main procedures for selecting comments.

important in cases where L cannot predict exactly what the speaker would say in various particular circumstances.

4.5. Resulting Behavior

1. If S says nothing about a given item, she will sometimes anticipate that L will rule out one or more of the possibilities for that item which are especially favorable (or unfavorable).

This is illustrated with Items 3 and 4 in Figure 4 (second column from the right): Since S would have considered the most favorable possibilities worth mentioning, she anticipates that L will interpret her silence on these items as implying that these possibilities are not realized.

2. S will often expect some of her vague comments to be given a more precise and accurate interpretation.

For instance, S now sees that her attempt to create an inflated impression of her amount of work experience (Item 1) will not be successful after all: L will recognize that if S had substantially more than one year of experience she would have said so.

3. The pragmatic interpretation of biased comments tends to undo, in part, the distorting effects of bias.

In the example just mentioned, the ultimate impact of S's biased comment is just about the same as that of the corresponding unbiased comment of Situation 1. But S can still derive some benefit from her bias, as with her silence on Items 3 and 4. Although L rules out the most favorable possibility for each item, he continues to reckon with the possibilities of intermediate favorability, whereas it is the least favorable possibility that is actually realized.

4.6. General Issues Raised

The strategy of simulating the listener's interpretation of an utterance before verbalizing it is applicable in many contexts (Jameson & Wahlster [JAME82] present a different sort of application and a discussion in more general terms). The present application is unusual in that the listener is in turn assumed to interpret an utterance by simulating the behavior of the speaker. Thus S's simulation of L's interpretation process involves largely considering how she herself would behave under various circumstances.

In the present situation, S only anticipates a pragmatic interpretation after having made a definite decision about what comment to make. It would in principle be possible for her to adopt a more sophisticated strategy, anticipating the pragmatic interpretation of each candidate comment when deciding which one is most desirable. This strategy could, among other things, help S to avoid investing unnecessary effort in making a comment which, after pragmatic interpretation, had exactly the same impact as a comment requiring less effort. On the other hand, it would require considerable extra processing on the part of S and could even involve an infinite regress if S thought that L knew that S was choosing her comments in this way. It would be interesting to investigate the conditions under which speakers apply some more complex strategy of this sort.

5. Situation 4: Discrepancy Between Actual and Projected Biases

5.1. Definition of the Situation

In Situation 3, the speaker implicitly assumed that the listener had a remarkable degree of awareness of the specific way in which she selected her comments. In the present situation, we will consider the possibility that L incorrectly assesses the nature of the speaker's bias. In other words, S now assumes that she has a certain **projected bias** which L ascribes to her and which may differ from her actual bias. We also assume that S will not attempt to change the nature of her projected bias, but on the contrary will attempt to behave in such a manner that the listener will continue to ascribe this bias to her.

5.2. How the Situation Can Arise

Social or group norms often produce a situation in which the listener expects the speaker to have a certain type of bias and the speaker wants to avoid violating this expectation. There are obviously many situations in which one is expected to be objective whether one wants to or not, but in some contexts (such as the personnel interview) positive bias is considered perfectly normal, and in others (such as eulogies), it is actually obligatory. More generally, in order to be fashionable or politically acceptable in a given community, one is often constrained to exhibit positive or negative bias with respect to certain Objects.

S may also want to project a particular bias because of the consequences that this will have for the way L interprets her utterances. For example, as will be seen in this section, the self-descriptive statements of a job applicant will often be interpreted in a more favorable manner if she manages to come across as being objective. This sort of motivation will be discussed in connection with Situation 6, which looks at attempts by S to create an expectation about her bias that did not previously exist.

5.3. Appropriate Strategies

The speaker should continue to be guided by her actual bias in deciding what she would like to say, since this bias determines the nature of the impression changes that she considers desirable. But when anticipating the pragmatic interpretations that L will make of her utterances, she should consider how desirable the various possible comments would be to a speaker who had the bias which she is attempting to project, since this is what L will be considering when interpreting her comments.

As we will see, it can happen in this situation that the comment that S would most like to make would not be made by a speaker with the bias which is being projected, no matter what the truth about the Object was. In this case, the comment would seem peculiar to L, who would not be able to understand why it was made, and he might then suspect that S's bias was not what it seemed to be. To prevent this from happening, S should avoid making any such comment and choose instead one which she considers less desirable but which is at least consistent with her projected bias.

5.4. Realization in the Simulation Model

In addition to the parameter which represents its actual bias, Imp has a parameter representing its assumed projected bias. These two parameters can have the same value (as implicitly assumed in Situation 3) or different values. The actual bias determines the system's initial choice of a comment on a given item, whereas the projected bias is consulted when the system anticipates the pragmatic interpretation that the listener would give to the comment. Thus in Figure 5 the "image" on which the pragmatic interpretation is based includes S's projected bias.

Step 3 in the first procedure in this figure realizes the strategy of avoiding comments which would be incompatible with this image.

5.5. Resulting Behavior

The right-hand column of Figure 4 shows the pragmatic interpretations expected in this situation, which can be compared with those of the previous situation. Note first that one narrower pragmatic interpretation, and two new ones, are made because the projected bias is objective rather than positive. With respect to Items 4, 10, and 12, on which S says nothing, L will now rule out the least desirable possibility, because of his faith that S would have made some corresponding comment if this possibility had been realized. This faith is, of course, not justified: in the case of Item 4, the least desirable possibility is in fact realized. This illustrates the first major consequence of a discrepancy between the actual and projected biases:

1. The listener's pragmatic interpretation of a comment may be incorrect, excluding the possibility that is actually realized.

The second main consequence has already been mentioned:

2. The speaker will sometimes be forced to make some comment other than the one she would most like to make, since the preferred comment would be inconsistent with her projected bias.

This happens in Figure 4 with Item 1: S determines that for any amount of work experience consistent with the formulation "more than a year", there is some more precise formulation which would be preferred by an objective speaker. The vague comment must thus be rejected, and S falls back on the more precise specification "about a year and a half". (This happens to be exactly the comment that an objective speaker would make in this case; but this is not a necessary consequence of the speaker's general strategy.)

5.6. General Issues Raised

5.6.1. Examples from Empirical Research

An illustration of the importance of both bias and projected bias was provided by Ceci and Peters [CECI84] in their study of the way faculty advisors rated students in confidential and nonconfidential letters of recommendation. Ratings were systematically higher in the nonconfidential letters, suggesting that the writers adopted a positive bias in order to avoid spoiling their relationship with their student. On the other hand, it was also found that this effect was well-known among the sort of people who tend to read letters of recommendation – i.e. that the projected bias of nonconfidential letter-writers was also positive. The authors not only hypothesize that this projected bias tends to undo the effects of the actual bias when the letters are read; they also speculate that some writers only adopt a positive bias in the first place because they expect that the projected bias will be positive and that therefore an objective letter would lead to the student's being underrated.

An experiment reported by Thomas (e.g. in [THMA83, pp. 189-192]) showed that the conditions under which subjects used potentially misleading formulations such as "Some X are Y" depended on whether their motivation was cooperative or competitive. The way other subjects interpreted such formulations depended on their assumptions about the motivation of the subjects who produced them.

5.6.2. Similar Patterns with Other Parameters

The distinction made here between actual and projected bias is in principle applicable to all of the other parameters that influence S's selection of comments, including S's assumptions about L's evaluation standards (discrepancies involving these will be examined in Situations 7 through 9). The importance of "projected talkativeness", for instance, is illustrated by the possibility that a job applicant who had few positive things to say about herself might try to come across as a "quiet type", so that her reticence would not give rise to much in the way of pragmatic interpretations.

5.6.3. More Deeply Embedded Discrepancies

This situation examined the consequences of a discrepancy between S's actual bias and the bias that S believes that L ascribes to her. Further discrepancies can involve more deeply embedded beliefs. For example, S may think that the situation is as follows: L recognizes S's positive bias but thinks that S is confident that she is projecting objectivity. L thus expects S to anticipate some incorrect pragmatic interpretations that L would not in fact be so naive as to make.

More generally, there are an infinite number of propositions of the form

$$\{\text{S believes that L believes that}\}^{N}\ \text{S's bias is B},$$

in which the bracketed expression is repeated N times. With N equal to 0 or 1, we obtain propositions about S's actual and projected biases, respectively; the more complex example just given involves, among others, a proposition with N equal to 2. In principle, B can take on arbitrary values in all of these propositions. But if the value of B for some N is different from the value for N+1, S thinks that L has some incorrect belief concerning her bias (at some level of embedding). This incorrect belief can in turn cause L to make further incorrect inferences. (Clark & Marshall [CLHH81], demonstrated a similar point with respect to the use of definite referential expressions.)

We will not go on to consider the consequences of discrepancies like the one in the example just given, or of still more complex discrepancies, since the conditions under which S can come to have such a set of beliefs become increasingly complicated and improbable. Rather, we will continue to assume – as implicitly in the present situation – that the values of B in all propositions of this form are equal, except that the value when N is zero may differ from the other values. This amounts to the assumption that S believes (not necessarily correctly) that L believes that S's bias is **mutually known** to the two of them (see [KOBS*] for an explication of this notion).

6. Situation 5: Responding to Specific Questions

6.1. Definition of the Situation

In this situation an actual dialog takes place. The listener asks the speaker questions which may be quite open or very specific; but each question clearly concerns some evaluation-relevant item or a set of related items. The speaker answers these questions in accordance with normal dialog conventions, but she may also take the opportunity to volunteer additional relevant information which falls outside of the scope of the question as strictly interpreted. To keep the complexity of the present analyses within reasonable bounds, we must also assume that S is not inclined to use L's questions as evidence concerning L's beliefs or values.

6.2. How the Situation Can Arise

This last assumption is not as restrictive as it may seem, even though many user modeling systems rely strongly on the user's questions as evidence concerning his or her goals, values and/or knowledge (see e.g. [CARB*, CHIN*, KOBS85a, chap. 4, MORI*]). Most such inferences by S must be based on assumptions about the reasons why the questioner has posed a given question. But within evaluation-oriented dialog as defined in the Introduction one cannot assume, for example, that the Evaluator only asks questions about the topics that he considers most important. He may ask some questions in order to satisfy his curiosity, to show polite interest, or even to avoid revealing anything about his evaluation standards. More generally, there are many asymmetric dialog situations in which a questioner is free to select topics more or less at will, whereas the answerer is constrained to be helpful and relevant. (This sort of situation is especially relevant for human-computer communication, since the computer system will often be expected to play the more subservient, constrained role.) In such situations it is reasonable for the answerer to keep in mind the goals and beliefs that the questioner can be assumed to have but not to try to draw any conclusions from the fact that he has asked specific questions.

6.3. Appropriate Strategies

6.3.1. Effects of Question Focus

It is convenient to define a **topic** in the present context as an item or a set of items which are related in content. The simplest type of question to analyze is what may be called an **attention-focusing question**, e.g. "Tell me about your education". Such a question invites S to say something about a given topic (the **question topic**) but does not specify the amount or type of information desired. An appropriate sort of response is what may be called a **focused monolog**: S can choose comments on any item within the topic, or even outside of it, applying essentially the same strategies as in the previous situations; but she must take into account the fact that L has chosen to focus attention on the question topic. She can do this by modify-

ing the way she assesses the prospective benefits and costs of producing any given comment.[8]

One reasonable way of taking question focus into account can be explained with the help of an analogy. Suppose that the Object is a picture hanging on a wall and that S and L are standing in front of it. In the previous situations, it was as if L had requested S to look at the entire Object and decide what was worth saying about it. Now, L is leading S closer to the picture and asking her to attend to some part of it when deciding what is worth saying. The less important the topic of L's question, the closer S and L must move, in order to bring it clearly into view; but typically, S and L will not move so much closer that the part of the Object asked about appears as large to them as the entire Object originally did.

The **importance** of a topic can be defined as the total amount of uncertainty associated with the items within that topic. Thus "Work experience" is an important topic in the example situation because the information about the candidate which concerns her work experience is likely to determine L's evaluation of her to a large degree. "Hobbies" is unimportant in that it is unlikely that the information provided concerning it will affect the evaluation strongly.

The following principle can now be formulated: the less important the question topic, the more strongly S should consider the benefits of making a comment on some item within that topic to be temporarily **magnified**, i.e., she should assess the desirability of each such comment as if the benefits were greater than they would be if she were simply volunteering the comment as part of a monolog. On the other hand, this magnification should not be so strong that a comment on an unimportant question topic will in general seem just as desirable as a comment on a very important one.

Once S has said whatever she chooses to on the focused part of the Object, she can consider whether there are other points that are worth mentioning. If she makes a comment on an item outside the question topic, she will be asking L to step backward to bring a larger part of the Object into view. The features mentioned will therefore not look larger to the same extent as the features within the focused part, and the effort required to make and interpret the comment will be greater because S and L have to step backward and increase the scope of their attention. These ideas can be expressed less metaphorically using the concept of the **common topic** which relates an item to the question topic. This can be defined as the least important topic that includes both that item and the question topic. For example, if the question asks about S's work experience, a comment on S's vocational training would not concern any part of the question topic, but it would be related to it via the common topic of "Competence at this type of work". The more important (or broader) the common topic is, the more the listener must expand the scope of his attention in order to see the relationship between the comment and the question topic.

---

[8] The concept of **focus** is defined and used in more complex ways in the work of Carberry [CARB*], McKeown [MCKE85b], and Reichman [REIC78], among others. The general issue of why speakers often volunteer information which falls outside of the scope of the question to which they are responding is treated e.g. by Schwitalla [SCHW79] and Siklossy [SIKL78].

The second principle governing the generation of a focused monolog is thus: the prospective benefits of a comment on an item outside the question topic should be magnified to the same extent as if L had asked a question on the common topic of that item and the question topic. But the effort associated with such a comment is greater than it would be if this common topic had in fact been asked about – the more so, the more important (i.e. broader) the common topic is, relative to the question topic.

Besides attention-focusing questions, there are many other types of questions that L might ask, each type placing a different sort of obligation on S to produce a particular sort of information. It is not necessary here to consider how S should handle all of these types of obligation (two types are dealt with in the simulation model); but a general point is that after S has fulfilled the obligations of the question she can proceed to generate a focused monolog, following the principles just introduced, starting with the question topic.

6.3.2. Preparatory Expressions

In this situation it is natural, if not inevitable, for S to preface her utterances with connectives (e.g. **but**), sentential adverbs (e.g. **unfortunately**) and longer idiomatic expressions (e.g. **I should mention that ...**) which give the listener some advance notice of the sort of impact that the utterance will have. Since these expressions do not fall into any single linguistic category, they will be referred to as **preparatory expressions**. Within evaluation-oriented dialog, the use of some of these expressions reflects the evaluation standards that S ascribes to L. On the one hand, this means that the criteria for their use can be specified using the same concepts that have been used to characterize the way S selects comments. On the other hand, this fact implies that such expressions can figure importantly in S's attempts to maintain a given image in situations where it is not entirely clear what evaluation standards she ascribes to L (Situations 7 through 9).

Some preparatory expressions (e.g. **You'll be glad to hear that ...**) announce that some particular impression of the listener is about to become either more or less favorable, in the sense defined in Situation 1. Others announce that the impending impression shift will be in the same (or the opposite) direction as the preceding one (e.g. **and, on the other hand**; cf. also Weydt's analysis of the German sentential adverb **immerhin** [WEYD79]). In some cases not only the direction but also the size of an impression shift is critical in determining the applicability of an expression (e.g. **unfortunately** is not used when the anticipated disappointment is of negligible significance); in these cases the size of the shift, like the prospective benefits of the comment, is magnified by the focusing effects of the question in the way discussed above.

Preparatory expressions like **by the way** announce that the comment to follow does not address the question topic directly; the conditions for their appropriate use are therefore related to the considerations which determine whether such a comment is worth making at all.

A somewhat different sort of preparatory expression is appropriate when L has asked a direct question whose answer S expects him already to have inferred from S's previous utterances, either directly or via pragmatic interpretation. Although

S is obligated to answer such a question, it is often advisable for her to make it clear that she considers the answer superfluous, for example by prefacing it with **of course**. Especially when she thinks that L should have inferred the answer via pragmatic interpretation, the use of such an expression serves to bolster S's image by reassuring L that he would have been justified in doing so.

6.4. Realization in the Simulation Model

The content relationships between items in the evaluation form are represented by the keywords that are associated with each item (e.g. for the item of Figure 1, the keywords are **motivation, competing-goals**, and **family-situation**). A topic can be specified in terms of a list of keywords: it is the set of all items whose associated keywords include those in the given list. In particular, the **common topic** which relates an item to a given question topic is specified by the keywords that the item has in common with the question topic.

The **importance** of a topic which consists of a single item is simply the uncertainty of L's impression with respect to that item at the beginning of the dialog, i.e., the standard deviation of the corresponding probability distribution. The importance of a topic which comprises several items can then be defined analogously as the standard deviation of the probability distribution which results from adding the distributions for the individual items (i.e. as the square root of the sum of the squares of the importances of the individual items).

Net desirability of a comment C on an item I,
after a question focusing on the topic QT =

Benefits of C x Benefit magnification
- Costs of C x Cost magnification

(Benefits and Costs are defined as in Figure 3.)

Benefit magnification =

$$\sqrt{\frac{\text{Importance of entire evaluation form}}{\text{Importance of common topic of I and QT}}}$$

Cost magnification =

$$\sqrt{\frac{\text{Importance of common topic of I and QT}}{\text{Importance of QT}}}$$

Figure 6. Formulas used by Imp in Situations 5 through 9 to take question focus into account when determining how worthwhile it is to make a given comment.

Using these definitions, the two principles stated above for taking question focus into account can be satisfied if Imp computes the net desirability of each comment according to the formulas given in Figure 6, which generalize the formulas of Figure 3. In particular, the use of the square-root function in computing the cost- and benefit-magnification factors ensures that differences in the basic desirability of comments cannot be canceled entirely by question focus.

To Respond to a Question Q:

1. If Q is a yes/no question, give the correct answer of **yes** or **no**.
2. [Only applicable in Situations 6 - 9:]
   If the projected bias is either indeterminate or different from the desired projected bias, then
   Attempt to Establish your Desired Projected Bias [cf. Figure 9].
3. For each item in the evaluation form, starting with those within the question topic and continuing with the others in ascending order of the associated cost-magnification factor [cf. Figure 6],
   Volunteer a Comment on that item [cf. Figure 5], taking into account its relation to the question topic [cf. Figure 6].
   (On the first of these items, exclude silence as a possible comment if Q was formulated in such a way as to require this.)

Figure 7. Imp's procedure for responding to questions in Situations 5 through 9.

Figure 7 shows how the general strategy of meeting the obligations imposed by the question and then offering a focused monolog is realized in the simulation model. In addition to simple attention-focusing questions, Imp handles yes/no questions (cf. Step 1 of the procedure) and questions which require some comment on a particular item, e.g. "How often have you changed jobs up to now?" (cf. the last sentence in Step 3 of the procedure).

Since Imp is not intended to address the problems of analyzing natural language input, it does not include a parser for questions. When asking the system a question, one must specify, in a special notation, the question type, the question topic, and (with yes/no questions) the possibilities that are consistent with an affirmative answer. Natural language formulations of questions, such as those shown in the example dialogs, serve only as glosses of the questions for the benefit of the reader.

The simulation model uses a limited number of the most frequently occurring preparatory expressions, though the principles used are applicable to many others. For the ones which announce that an impression is about to become less or more favorable, the critical quantity is the change in the expected value of the impression (as defined in Situation 1.4), multiplied by the benefit-magnification factor (as defined in Figure 7). **Unfortunately** or **I'm afraid** is chosen if this shift is in the negative direction and its size exceeds a certain threshold. If two successive impression shifts

are in the same direction, the comment which produces the second one is prefaced by **and** or **in fact**, the latter being chosen only if the second comment concerns the same item as the first one. **But** is used if the second of two successive changes is in a different direction than the first one. In addition, **at least** is inserted after **but** if the second shift is in the positive direction, its size exceeds a certain threshold, and the two comments concern the same item.

If the system's obligatory answer to a question does not result in any impression change at all, the answer is prefaced with **of course**. The expressions **by the way** and **I should mention that ...** are used if the difference between the cost-magnification factor (as defined in Figure 7) and 1.0 (or the cost-magnification factor of the preceding comment, if this was greater than 1.0) exceeds a certain threshold.

6.5. Resulting Behavior

Many of the consequences of the principles stated in Section 6.3 follow so directly that they needn't be listed explicitly; illustrations will be found in the example dialog of Figure 8. The following consequences are less direct.

1. S will often (though not always) follow up on an obligatory answer concerning an item with an additional comment on the same item.

This happens in Figure 8 after the yes/no questions 2 and 4. Even after S has supplied the obligatory information, L's impression concerning the item may be fairly indefinite and/or inaccurate. There are thus potential benefits for S to gain by changing the impression, and these benefits are temporarily magnified by the question focus. This last consideration can be formulated more generally:

2. S may volunteer a comment which will have only minor impact on L's impressions if it is related to a sufficiently unimportant topic which has been focused on by a question.

This follows from the principle concerning the magnification of the benefits of comments on question-related items. In Figure 8, Question 4 provides the clearest example, since S assumes that the item "Variety of previous work" is of low importance; but the responses to the first two questions also illustrate the effects of focusing on topics of differing importance.

3. Comments which are only indirectly related to the question topic tend to become less frequent in the course of the dialog.

There are typically a few items about which S would particularly like to say something, e.g. the ones she would comment on in an unconstrained monolog as in Figure 4. Applying the strategies introduced in this section, S will comment on such an item as soon as a question is asked to which the item is reasonably closely related (see e.g. the responses to Questions 2 and 3). But as more and more comments of this sort are made, there are fewer and fewer cases left where it seems worthwhile to depart from the question topic. Thus after Question 5, S would surely have mentioned her preparatory school background had she not already slipped in a comment on this after Question 3.

4. If S's projected bias is different from her actual one, she will now more frequently find it necessary to volunteer some comment other than the one she would most like to make, in order to maintain her image.

Natural Language Formulation and Type of Question	Formulation of Answer	Commentary on Answer
1. Tell me something about your personal situation. (focusing on general topic "Personal situation")	I LIVE HERE IN AMSTERDAM	No mention of family situation; anticipated benefits not sufficient to justify effort required.
2. Do you have children? (yes/no question on item "Family situation")	NO IN FACT I LIVE ALONE ... AND BY THE WAY THE REASON I WANT TO LEAVE MY PRESENT JOB IS THAT I'M LOOKING FOR MORE CHALLENGING WORK	Additional information volunteered because of the focus on the item - but still not detailed enough to rule out all possibilities. Comment on item within broader topic of "Motivation for working at this firm".
3. How challenging has the sort of work been that you've done up to now? (request for comment on item "Complexity of previous work")	UNFORTUNATELY MY WORK HAS BEEN PURELY ROUTINE ... BUT BY THE WAY I'VE BEEN WORKING FOR ABOUT A YEAR AND A HALF ... AND I WENT TO UNIVERSITY PREP SCHOOL	Some comment other than silence obligatory here; vague, more positive comment would be inconsistent with projected objectivity. Optional comments on items within broader topic of "Competence at this sort of work".
4. Did you get a good deal of experience in different departments? (yes-no question on item "Variety of previous work")	NO BUT I SUBSTITUTED OCCASIONALLY IN OTHER DEPARTMENTS	A minor point - but mentioned because of focus on very specific topic.
5. Did you graduate from the MTS? [= higher-level graphic training]? (yes/no question on item "Vocational training")	NO	An objective speaker would here definitely add that she had received no vocational training at all.
6. But of course you did get some vocational training, didn't you? (unexpected yes/no question on same item)	NO I GUESS I'D BETTER BE GOING - GOODBYE	Canned formulation for case where discrepancy between actual and projected bias has been exposed.

Figure 8. Dialog conducted with Imp in Situation 5. (The actual bias is positive and the projected bias is negative.)

In Situation 4 it was seldom necessary for S to suppress a desired comment in order not to reveal her bias, since the primary consequence of her bias was silence on items about which the truth was unfavorable, and silence is usually consistent with all possible biases. But when S has to respond to specific questions, silence is less often acceptable, even where S has some freedom in her choice of comments. On the one hand, some questions (such as Question 3 in Figure 8) obligate S to produce some comment other than silence on a given item.

If S would like to remain vague in such a case, she may find that the only available vague formulations require about the same amount of effort as more precise ones and would therefore be inconsistent with objectivity.

On the other hand, even when a question does not explicitly exclude silence as a comment on a given item, the extent to which the item is focused upon may magnify the benefits of comments so strongly that an objective speaker would say something no matter what the truth was. (This is almost the case with Question 5, which is why it may strike the reader as implausible that S should add nothing to her obligatory yes/no answer; but it happens that one of the remaining possibilities for the item, "lower-level graphic training", is so much more commonly realized than the other ones that even an objective speaker would not bother mentioning it.)

5. It is now more dangerous for the speaker to allow the listener's pragmatic interpretation to yield an inaccurate impression about a given item.

If the listener for some reason later asks a question about the topic concerned which forces the speaker to disclose the truth, the speaker's behavior will be revealed to have been inconsistent with her projected bias. This pattern can be seen at the end of Figure 8's dialog. (Since it is difficult to specify a generally appropriate recovery strategy, the simulation model simply gives up and terminates the dialog.)

6.6. General Issues Raised

The above discussion of preparatory expressions is not intended as a balanced contribution to the study of their semantics. In particular, the criteria given for the use of connectives take into account only the evaluative implications of statements (as in "She is young [positive] but inexperienced [negative]"), whereas other implications can also influence the use of such connectives (as in "She is young and [as one might therefore expect] inexperienced"). (A more comprehensive account of the semantics of connectives is given, e.g. by Carlson [CARO83], whereas Van Dijk [VAND81] discusses their pragmatic uses.)

A particularly complex question concerns what a speaker should do if her projected bias has been exposed (cf. [HARR63, chap. 13] for a discussion of related issues). In some contexts the revelation that S does not in fact have the bias she was projecting has consequences which make it pointless for her to continue the interaction. But even where continuation is possible, S must try to guess how L will handle the difficult task of retracting the largely unjustified pragmatic interpretations that he has already made. (Nisbett and Ross [NIRO80, chap. 8] discuss psychological theories and empirical results relevant to the problem of retracting the inferences that one has made on the basis of an assumption that later turns out to have been incorrect.)

7. Situation 6: Establishing a Desired Projected Bias

7.1. Definition of the Situation

In the preceding situations it was assumed that the speaker had, from the beginning of the dialog, some belief about her projected bias and that she wanted to behave consistently with this bias. In the present situation this assumption is loosened: L may begin the dialog with no definite idea about S's bias and/or S may not be sure what L thinks about her bias. More importantly, S now wants to behave in such a way that L will ascribe to her some particular bias. Speakers sometimes try to accomplish this by making an explicit claim about their bias (e.g. "I'm going to be perfectly frank"), but it is more interesting to stipulate that S can only accomplish her goal by selecting a single factual comment which, because of the nature of the information conveyed about the Object, convinces the listener that S has the desired bias.

7.2. How the Situation Can Arise

As noted in Section 5.2, social norms and expectations can make it desirable for a speaker to project a certain bias. In addition, each possible projected bias has consequences for the way L interprets S's individual utterances, as was seen in the two previous situations. These consequences can make a particular projected bias seem particularly attractive to S.

First, if S wants communication to be maximally efficient, she should try to ensure that her projected bias matches her actual bias, since otherwise some justified pragmatic interpretations may not be made and some incorrect ones may be.

A discrepancy between the actual bias and a particular projected bias also has the drawback that it may be exposed at some point (as illustrated in Situation 5), no matter how hard S tries to avoid this. Since this sort of unintended exposure can be undesirable in various ways, S may prefer to eliminate any discrepancy at the outset, unless she has something to gain which justifies taking such a risk.

How S can benefit from such a discrepancy has been illustrated for the combination of positive actual and objective projected bias: L can be expected to rule out some negative possibilities via pragmatic interpretation, including some which are in fact realized. This pattern therefore leads to generally more favorable impressions; it is especially attractive if S is worried that certain important negative facts might come to light, e.g. if S is a job applicant who has a "skeleton in her closet" that the employer would be likely to uncover with one of his questions if he suspected that S was positively biased.

No examples have been given yet of the extreme combination of positive actual bias and negative projected bias, and one may wonder whether it offers the speaker any advantages beyond those just mentioned for the positive/objective combination. The main differences when the projected bias is negative are that (a) positive possibilities can hardly ever be mentioned, but (b) they are also hardly ever ruled out by pragmatic interpretation. Thus the positive/negative combination is more

attractive than the positive/objective one mainly when there are no (or hardly any) strongly positive facts that S could mention; for it ensures that L will at least continue to reckon with all positive possibilities that haven't been explicitly ruled out. For example, a job applicant who has very little to offer may try to come across as being extraordinarily modest or self-critical. If she succeeds in this, the fact that she volunteers very little information about herself will be interpreted as implying that she has very few negative attributes and perhaps some awe-inspiring virtues.

The situation of the speaker who actually has a negative bias needn't be discussed explicitly, since it is the mirror image of the situation of a positively biased speaker.

7.3. Appropriate Strategies

S should try to establish her desired projected bias as soon as possible in the dialog, especially insofar as she is concerned about the pragmatic interpretations that L will make on the basis of it. (In experimental research on attitude change, for example, there is some evidence that information about a communicator's credibility has more effect when it is presented before the persuasive message than after it – see e.g. Ward & McGinnies [WARD74]). As noted at the end of the previous section, if L changes his hypothesis about S's bias at a late stage he will have difficulty in revising all of his previous pragmatic interpretations accordingly. In any case, whatever the reasons, everyday experience tells us that speakers and writers typically make any efforts to convey information about their bias (or any other general parameters which guide their behavior) as early as possible.

Within the situation as just defined, the only way that S can establish a given projected bias is to produce a comment which is uniquely characteristic of that bias. That is, the comment should be consistent with that bias and inconsistent with any other bias that L thinks that S might have. This strategy is appropriate in view of a general (and rational) tendency often noted in social-psychological research (see e.g. [JONE65, AJZE71]) which is applicable to the way a listener interprets the behavior of a speaker: people explain an act by attributing to the actor a particular trait (e.g., a particular bias), if the act has consequences which would be desirable only to a person with that trait.

S should try to find a comment which not only will satisfy the above constraints but will also be fairly desirable (or at least not extremely undesirable) to herself in terms of the general criteria introduced in the previous situations, i.e., the desirability of the resulting impression change and the amount of effort required. As an extreme example, a job applicant will not want to volunteer information which implies that she is completely unqualified for the job simply in order to appear objective.

Since these constraints will often not be satisfied by any comment which pertains to the topic of L's first question, S should also consider commenting first on some other topic, as long as such a digression would not seem completely unnatural.

7.4. Realization in the Simulation Model

Instead of beginning the dialog with a given projected bias, Imp now begins with a **desired projected bias** and one or more **undesired projected biases**. In the examples in the middle column of Figure 10, the desired projected bias is **objective** and the only undesired one is **positive**; these values are reasonable if negative bias is a priori so unlikely in the given context that L will not reckon with it as long as S's behavior is interpretable in terms of one of the other biases. In the examples in the third column, S's desired projected bias is in fact negative, so it is necessary to treat as undesired both of the more likely biases **positive** and **objective**.

To establish a desired projected bias as opposed to certain undesired projected biases:

1. For each item in the entire evaluation form,
   For each comment C on that item, in descending order of net desirability [cf. Figures 3 and 6], given your actual bias and taking into account the item's relation to the question topic,
   anticipate the pragmatic interpretation of C [cf. Figure 5] using, in succession, your desired projected bias and each of the undesired projected biases.
   If the result is not empty in the former case and it is empty in the latter cases, then
   Note that C is a suitable comment and record its net desirability.
   Do not consider further possible comments on this item, but proceed with the next item.
2. If you have found some suitable comment on at least one of the items, then
   Verbalize the comment with the highest net desirability.
   Assume that your projected bias is now as desired.

Figure 9. Procedure used by Imp to select a comment which will establish a given projected bias.

The procedure the system uses to search for appropriate comments is shown in Figure 9. Figure 7 shows the point within the general question-answering procedure at which it can be invoked, namely after L's first question (after S has given any obligatory yes/no answer). Invocation will be unnecessary after subsequent questions unless S fails to come up with a suitable comment while responding to the first question. Briefly, the procedure involves a search through the entire evaluation form for the most desirable comment which fulfills the requirements given above. Only after the highest-rating candidate has been verbalized does S proceed to volunteer comments in the usual manner, assuming that all subsequent comments will be interpreted in the light of the newly established projected bias.

7.5. Resulting Behavior

The four comments in Figure 10 which are marked with an asterisk are only generated by S under a condition which will be introduced in Situation 8, namely that L is quite uncertain about the evaluation standards that S ascribes to him. The consequences that this has for S's attempts to establish a projected bias will be discussed in Section 9.5; for now it is sufficient to note that these comments are at best marginally consistent with S's desired projected bias.

When one looks at the procedure in Figure 9 or the discussion in the text, it is not immediately obvious what sort of comments the strategy outlined will produce; and in fact a rather surprising variety of results emerge. Since they are quite different for the cases where an objective and where a negative projected bias is desired, these cases will be discussed separately.

7.5.1. Comments Intended to Establish Objectivity

1. The most typical comment is one which is just negative enough to seem reasonably informative.

This type of comment is exemplified by the responses to Questions 1 through 3 in the middle column of Figure 10. The first response shows S going to considerable pains to avoid saying anything more negative than necessary: the comment is only vaguely germane to the question asked, but S nonetheless prefers it to a directly relevant comment (on her lack of vocational training) that would be much more negative.

2. How negative a comment of this type must be depends on (a) how strongly the item it concerns is focused on by the question and (b) how much freedom S has in choosing a comment on that item.

The comment volunteered after the yes/no answer to Question 3 results in only a slight downward impression shift, but according to the principles introduced in Situation 5 it is plausible because L himself has focused attention on the unimportant topic of S's hobbies.

By contrast, when S has to shift attention to a distantly related item, the resulting comment must in general be strongly negative in order to seem worthwhile at all. This is why S has to be more negative after Question 2 than after Question 3, although the two questions both concern S's hobbies. Since Question 3 is a yes/no question, the bit of negative information S adds is optional and thus appears to reflect objectivity. Question 2, on the other hand, obliges S to make some comment on her hobbies, so the same remark in this context does not prove that she is not positively biased. Consequently, S must slip in the less relevant but more negative comment about the routine nature of her work.

3. A comment need not be negative at all to establish objectivity, if it is clear that a positively biased speaker would have made an even more positive comment.

Negative comments as in the first three examples were clearly inconsistent with positive bias because a positively biased speaker would rather say nothing at all than make them. But as we saw in Situation 2, the behavior characteristic of positive bias includes not only silence on matters where the truth is unfavorable but also vague formulations (e.g. "working for more than a year") which leave open one or

Formulation of Listener's first question	(Undesired) → Desired Projected Biases: (Positive) → Objective	(Positive, Objective) → Negative
1. Tell me something about your education. (request for comments on topic "Education")	* I SHOULD MENTION THAT UNFORTUNATELY MY WORK HAS BEEN PURELY ROUTINE ... BUT I WENT TO UNIVERSITY PREP SCHOOL Normal amount of vocational training. [!]	I'M AFRAID I DIDN'T GET FULL-TIME VOCATIONAL TRAINING Has taken some evening courses. [!]
2. I spend a lot of free time pursuing my hobbies. What about you? (request for comment on item "Hobbies")	* I SHOULD MENTION THAT UNFORTUNATELY MY WORK HAS BEEN PURELY ROUTINE ... AND MY HOBBIES DON'T HAVE ANYTHING TO DO WITH THIS TYPE OF WORK	I HAVE SOME HOBBIES Hobbies are relevant to this type of work. [!]
3. Do you have any hobbies? (Yes/no question on item "Hobbies")	YES BUT I'M AFRAID MY HOBBIES DON'T HAVE ANYTHING TO DO WITH THIS TYPE OF WORK	YES * BUT BY THE WAY I'M AFRAID I DIDN'T GET FULL-TIME VOCATIONAL TRAINING Has taken some evening courses. [!]
4. Has the work you've done up to now been challenging? (yes-no question on item "Level of previous work")	NO BUT I SHOULD MENTION THAT I'VE BEEN WORKING FOR ABOUT A YEAR AND A HALF Work has been purely routine.	NO AND BY THE WAY I'M AFRAID I DIDN'T GET FULL-TIME VOCATIONAL TRAINING Has taken some evening courses. [!]
5. Why did you decide to leave your present job? (request for comment on item "Reason for changing jobs")	* I SHOULD MENTION THAT I'VE BEEN WORKING FOR ABOUT A YEAR AND A HALF ... AND THE REASON I WANT TO LEAVE MY PRESENT JOB IS THAT I'M LOOKING FOR MORE CHALLENGING WORK	I DIDN'T GET FIRED FROM MY LAST JOB Is looking for more challenging work.
6. How much variety was there in your last job? (request for comment on item "Variety of previous work")	I SUBSTITUTED OCCASIONALLY IN DIFFERENT DEPARTMENTS	I DIDN'T GET A LOT OF EXPERIENCE IN DIFFERENT DEPARTMENTS Got some experience in different roles.

Figure 10. Comments made by Imp after various opening questions with the aim of establishing a particular projected bias. (Any subsequent comments volunteered on items within the question topic are also listed, as are any pragmatic interpretations that L is expected to make with respect to such items. Pragmatic interpretations are summarized in lower case. '*' indicates that a comment is at best marginally acceptable, as discussed in connection with Situation 8, and '[!]' indicates that a pragmatic interpretation is incorrect.)

more possibilities which are more favorable than the one actually realized. S can thus appear objective simply by resisting the temptation to generate such a vague comment and substituting a more precise one (e.g. "working for about a year and a half") – which may still result in a somewhat positive impression shift. Comments of this sort are made after Questions 4 through 6 in Figure 10.

4. Even when the speaker's actual bias is objective, she will sometimes be able to establish it, if at all, only by making a comment that she would normally not consider worthwhile.

One would intuitively expect the task of establishing a projected bias to be easier for a speaker who was simply trying to make clear her true dialog motivation. And indeed, after Questions 1 and 4 in Figure 10, S would produce more straightforward responses than those listed in the second column if its actual bias were objective, simply beginning with negative comments on the topic of the question.

But sometimes, as with Questions 2 and 5, none of the things that an objective S would normally say in response to the question happens to differ from what a biased speaker would say. S may then find no suitable comment at all, or – as in the two examples just mentioned – have to produce a comment which she doesn't actually consider worthwhile in its own right, though it is not blatantly implausible. The reader can probably recall from his or her own experience how difficult it is, for example, to avoid appearing negatively biased when commenting on a piece of scientific work which is almost totally without merit.

7.5.2. Comments Intended to Project Negative Bias

5. The comment most suitable for establishing a negative projected bias is typically a vague one which rules out only the most positive or the most negative possibilities.

In everyday situations negative bias is often associated with emotionally charged expressions and implausible exaggerations; but within the constraints of the present situation the available ways of convincing a listener that one is negatively biased are more subtle. Extremely negative factual comments will not work, since they would be just as likely to be made by an objective speaker as by a negatively biased one. In order to be inconsistent with objectivity as well as with positive bias, a comment must be vague and allow only the following interpretation by L: "The possibility which is realized must be the most favorable one that is allowed by S's comment; S is being vague so that I will not rule out the less favorable possibilities."

In the examples in the right-hand column of Figure 10, L in fact always makes the most favorable possible pragmatic interpretation – which happens to be incorrect in the first two cases (the next two involve the same comment as the first one). In these cases S, with the very comment that she uses to establish her negative projected bias, draws her first benefit from it in the form of an unjustifiably favorable pragmatic interpretation.

Generally speaking, vague comments are characteristic of positive as well as negative bias, as we have already seen in several examples. One may therefore wonder under what circumstances a given vague comment is suitable for projecting one of these types of bias, as opposed to being compatible with both of them. This can only occur when the impression left by the comment on a straightforward interpretation is asymmetric with respect to the favorableness and probability of

the possibilities that must still be reckoned with. For example, the comment "I have some hobbies" after Question 2 only works because the most favorable possibility (i.e. that S has some hobby related to her work) is quite improbable. It would thus not be worthwhile for a positively biased S to remain vague in order to leave this favorable possibility open, whereas it is plausible that a negatively biased speaker might take pains to avoid mentioning it specifically.

6. Even if the speaker's actual bias is negative, she may be able to project this bias, if at all, only by saying something different from what she would like to say.

This consequence is largely similar to the fourth one listed above for the case of objective bias; e.g. after Question 3 even a speaker who was actually negatively biased would have to resort to an almost irrelevant comment in order to get her bias established. But a new phenomenon is the bias-projecting comment which is more vague than the one S would normally make. For example, after Questions 1 and 2 a negatively biased S would normally be inclined simply to describe the possibility actually realized, since it is the most negative one in each case. But if she wants to make it clear that she is not objective, she has to remain vague, leaving open at least one possibility which is more favorable. Paradoxically, then, she can only make clear how negatively biased she is by making a comment which is misleadingly positive.

7.6. General Issues Raised

It is instructive to compare the analysis in this section to social-psychological research which shows that listeners are more strongly influenced by a persuasive message if they think that it was produced by a speaker whom they would normally expect to be biased against the position advocated. For example, in an experiment by Walster, Aronson, and Abrams [WALS66], a passage arguing that prosecutors should be granted increased power was more persuasive when the speaker was said to be a criminal than when he was said to be a prosecutor; the reverse was true of a passage which argued for the opposite conclusion. Eagly, Chaiken, and Wood [EAGL81] showed that when an argument runs against the speaker's own interests, the listener perceives the speaker as being less biased, as one would expect in the light of the considerations discussed in this section: such a speaker makes statements which are inconsistent with the expected bias; some of these are consistent with either the opposite bias or objectivity, but since it is implausible that a speaker should actually be biased against her own interests, the listener can only conclude that the speaker is attempting to convey an accurate picture of the facts.

Eagly et al. argue that in this case the listener is more willing to accept the speaker's conclusion and, at least in some cases, is less inclined to scrutinize the speaker's arguments critically. This account is not applicable to evaluation-oriented dialog as defined in this paper, since here the speaker does not explicitly argue for any conclusion. The analysis in this section, however, suggests that the effect can also be explained in terms of the pragmatic interpretations that the listener makes of the speaker's individual statements. These interpretations will often be less favorable to a given position if the speaker is seen as being biased in favor of that position. (For example, if an apparently biased prosecutor reports that the crime rate in a

given area is "rising", the listener may suspect that the extent of the increase is in fact negligible, whereas an apparently objective speaker would be expected to make such a statement only if the increase was large enough to be worth mentioning.)

Although these studies confirm that listeners do make inferences about a speaker's bias in the general way hypothesized here, they say nothing about how easy it is to establish a projected bias with a single comment, as the speaker attempts to do in the present situation. Indeed, the reader may well feel that he or she as a listener would not be convinced about the speaker's bias by some of the individual comments in Figure 10. The definition of this situation would be more realistic if it took into account the fact that sometimes the cumulative evidence of several comments is required to establish a projected bias. An analysis of this sort would have to take into account the fact that there are various degrees of prior probability of given sort of bias in a given situation and also various degrees to which a comment can be (in)consistent with a given bias. A study by Ajzen [AJZE71] shows that people take such considerations into account in a way which can be described fairly well in Bayesian terms.

A further complication is that, since attempts to project a bias that one does not actually have are so familiar, listeners will often recognize the pattern and therefore not be as gullible as S assumed in the above analysis. But then again, if L does suspect that S is attempting to project a bias that she doesn't have, it is difficult to specify what alternative strategy S could use to overcome this suspicion – even if the suspicion is not justified.

Since the assumptions underlying the present analysis exclude these complications, its main contribution is to clarify the nature of the individual comments which can contribute to a successful attempt to project a given bias.

8. Situation 7: Discrepancy Between Actual and Projected Ascriptions

8.0. Introduction: Embedded Beliefs About Evaluation Standards

Up to now it has implicitly been assumed that the listener had very definite values and expectations and that these were mutually known to S and L (cf. the remarks at the end of Situation 4 concerning embedded beliefs about S's bias). Since this is a very strong assumption, the remaining three situations will be defined in terms of more realistic alternatives to it. But first it will be helpful to introduce some convenient terminology and to mention two types of complication that will not be examined in detail.

8.0.1. Uncertainty in L's Evaluation Standards

First, L may be uncertain as to what the appropriate evaluation standards are and therefore sometimes be unsure just how pleased or disappointed he should be at the information S provides. In extreme cases, it might in fact be inappropriate to speak of evaluation standards at all – e.g. if L was unable to choose between two

contrasting approaches to evaluating Objects of the type in question. We will continue to restrict our attention to cases where it is reasonable for S to view L as confidently applying a single set of evaluation standards.

8.0.2. Uncertainty in S's Ascription of Evaluation Standards to L

S can in general estimate L's evaluation standards only approximately. It is very difficult to predict a person's values and expectations precisely unless one is provided with quite specifically relevant clues.[9]

This sort of uncertainty in S would tend to make her unsure as to what comments would have the most desirable effects on L's impressions of the Object; this uncertainty could in turn be reflected in the use of expressions such as **You may be interested to hear that ....** But the consequences are less serious than they would be if S had the responsibility of making a decision on behalf of L (or a recommendation, as in [MORI*]), i.e., if L's evaluation standards were **decision properties** in Sparck Jones' [SPAR*] terms. We will therefore continue to restrict our attention to cases where S can be viewed as ascribing certain evaluation standards to L, and we will ignore the complications that can arise because of the fact that these ascriptions represent only S's best estimates of L's values and expectations.

To make it easier to talk about embedded beliefs, we will henceforth refer to the evaluation standards that the speaker ascribes to L as the speaker's **ascriptions.** In analogy with the concept of a **projected bias** we can now define S's **projected ascriptions** as the standards that L thinks that S ascribes to him. A discrepancy between S's actual and projected ascriptions is the defining characteristic of the situation to be discussed within this section; but we will assume that no more deeply embedded discrepancies exist; i.e., S thinks that L believes S's ascriptions to be mutually known (this is analogous to the assumption made at the end of Section 5 with respect to beliefs about bias).

8.1. Definition of the Situation

S's projected ascriptions (as just defined) are now different from those that she actually ascribes to L. As in Situation 4, S wants to behave in such a way that this discrepancy will not be noticed by L.

8.2. How the Situation Can Arise

There are many ways in which such a discrepancy can arise; the following case is the one that will be assumed in the specific examples below.

Employers are often inclined to apply different evaluation standards to applicants from different categories. For example, an employer might evaluate a male

---

[9] Brief discussions of some of the highly fallible heuristics that people use to make such predictions in everyday situations can be found in [JAME84, NIKU85and – in more general terms – HOLL86, chap. 8]. The techniques employed to construct user models also tend to yield speculative conclusions, as can be seen in several chapters in the present volume (especially [FINI*, RICH*, SPAR*]).

candidate with special attention to his suitability for eventual promotion into a management role; faced with a female candidate for the same job, the same employer might be more concerned about the possibility that she might leave within a short time because of family commitments. When the use of different standards can be seen as unfair, the employer may want (or be legally obligated) to appear to be applying the more socially acceptable standards to all applicants.

Consider now the applicant who is sophisticated enough to recognize that she is being judged according to the less desirable standards. She may consider it undiplomatic to reveal her awareness of this fact and thus she may behave as if she thought she was being judged according to the more acceptable standards.

8.3. Appropriate Strategies

The speaker in this case should use her actual ascriptions in deciding what is worth saying, since these reflect L's actual concerns. But when anticipating L's pragmatic interpretations, she should bear in mind that L will be basing them on the assumption that she was generating her utterances on the basis of the projected ascriptions. This is analogous to the role of actual and projected biases in Situation 4.

A new problem concerns the preparatory expressions discussed in Situation 5: since these presuppose certain evaluation standards, their applicability will in some cases be different according to the actual ascriptions than according to the projected ones. In order to appear to be generating her utterances on the basis of the projected ascriptions, S should of course refer to these in selecting such preparatory expressions.

The most subtle problem concerns the effects of question focus, which also depends in part on evaluation standards. For example, the topic of a question might be quite important according to the actual ascriptions yet trivial according to the projected ascriptions. Here again, the actual ascriptions are relevant for deciding what to say and the projected ones for anticipating pragmatic interpretations.

8.4. Realization in the Simulation Model

8.4.1. Representation of Projected Ascriptions

In principle the actual and projected ascriptions could be represented with two separate evaluation forms. These could differ with respect to the items that were defined, the possibilities that were distinguished within each item, and/or the values and probabilities associated with each possibility. But the major phenomena of interest can be most clearly illustrated using a very restricted form of discrepancy; we need to allow each pair of corresponding items in the two forms to differ only with respect to their importance (as defined in Situation 5). That is, each item in the projected evaluation form is identical to one in the actual evaluation form except that the values (i.e. numbers of points) assigned to the possibilities are all multiplied by a certain nonnegative **weight**. Thus, for example, if this weight is 3.0 for a given item, all impression changes involving that item will be three times as large

according to the projected evaluation form as according to the actual evaluation form. With this restriction, it is of course unnecessary for the simulation model to store two separate evaluation forms. One need only specify for each item the weight which expresses the ratio between its projected and actual importance.

The implementation of the strategies just sketched involves simply using the relevant weights when rating the desirability of alternative comments during the anticipation of pragmatic interpretations and when computing the impression shifts that are relevant for the selection of preparatory expressions.

8.4.2. New Items for the Examples

In the specific interview which provided the context of the examples discussed above, our Informant did not suspect any significant discrepancy between her actual and the projected ascriptions, but the relevant evaluation standards were somewhat different in a later interview with a smaller firm which had plans for expansion. On the one hand, she expected this firm to be interested in employees with the ability and motivation to move up eventually to a management-level position, as expressed in the first two of the five new items referred to in the left-hand column of Figure 11. The Informant suspected that these items would receive only about one third as much weight for her as they would for a male candidate. On the other hand, in this interview it was known in advance that she lived nearby and had a boyfriend. The two previous items about her personal situation were therefore in effect replaced by the remaining new items referred to in Figure 11, which all concern the possibility that she might leave her job because of commitments involving her boyfriend; she guessed that these items were about three times as important in her case as the employer would have liked to admit.

The example questions and answers in Figure 11 concern only these five new items, although the remaining ten old items were still part of the evaluation form and could in principle have been commented on. To facilitate comparison across Situations 7 through 9, the same four questions were posed to the system in all three situations, although the questions would presumably not be equally likely to be asked in all of them.

8.5. Resulting Behavior

The first column of responses in Figure 11 (i.e. Column 1), provides a baseline for understanding the other responses: as in Situation 5, there is no discrepancy here between S's actual and projected ascriptions, i.e., S is willing to let it be known that she has **realistic** expectations about L's evaluation standards. Her minimal answers to the first two questions reflect her assumption that L is not really very interested in the items focused on. When responding to the third question, by contrast, she makes it clear that she thinks L has finally asked about the matters that really concern him; in fact since her second comment anticipates L's fourth question, she expresses surprise when he asks it.

Column 2 illustrates an attempt by S to act as if her ascriptions were **naive**, i.e. as if she thought she were being evaluated according to socially acceptable standards.

Formulation of question	Assumed Nature of Speaker's Projected Ascriptions			
	1	2	3	4
	Situation 5	**Situation 7**	**Situation 8**	**Situation 9**
	Definitely Realistic	**Definitely Naive**	**Definitely Naive; Listener Uncertain About Specific Values**	**Unclear Whether Realistic or Naive**
Unimportant items:				
1. How much experience have you had in leadership positions? (item "Leadership Experience")	I NEVER GOT MUCH LEADERSHIP EXPERIENCE Has a bit of leadership experience.	I'M AFRAID I NEVER GOT MUCH LEADERSHIP EXPERIENCE Has a bit of leadership experience. Has moderate interest in management position. [!]	I'M AFRAID I NEVER GOT MUCH LEADERSHIP EXPERIENCE Has a bit of leadership experience.	I NEVER GOT MUCH LEADERSHIP EXPERIENCE Has a bit of leadership experience. [Only if projected ascriptions are naive:] Has moderate interest in management position. [!]
2. Do you have any interest in moving into a management role? (item "Interest in management positions")	YES	YES OF COURSE	YES	YES
Important items:				
3. By the way, does your boyfriend have a steady job around here? (item "Boyfriend's career plans")	UNFORTUNATELY NOT ... BUT AT LEAST TOM DOESN'T HAVE DEFINITE PLANS TO MOVE ... AND BY THE WAY TOM WANTS ME TO KEEP WORKING INDEFINITELY ... AND I DONT EVER INTEND TO HAVE CHILDREN	UNFORTUNATELY NOT No definite plans to move.	UNFORTUNATELY NOT ... BUT BY THE WAY TOM WANTS ME TO KEEP WORKING INDEFINITELY ... AND I DON'T EVER INTEND TO HAVE CHILDREN	UNFORTUNATELY NOT No definite plans to move. [Only if projected ascriptions are realistic:] Boyfriend won't insist she stop working soon. Expects to have children some time [!] but doesn't plan to quit soon to do so.
4. Tom isn't going to ask you to quit working and keep house, is he? (item "Boyfriend's attitude")	OF COURSE NOT	NO	OF COURSE NOT	NO [No longer possible to behave consistently with both realistic and naive projected ascriptions:] IN FACT TOM WANTS ME TO KEEP WORKING INDEFINITELY ... AND I SHOULD MENTION THAT I'M NOT PLANNING TO HAVE CHILDREN IN THE NEXT FEW YEARS
5. [No question asked.] (item "Plans for children")				

Figure 11. Responses of Imp to questions under four different assumptions about the speaker's projected ascriptions. (The actual bias is positive and the projected bias is negative. Pragmatic interpretations are summarized in lower case and '[!]' indicates that a pragmatic interpretation is incorrect.)

The items were thus given higher or lower weights in the simulation model, as noted above, in computations involving the projected ascriptions. This column illustrates the main consequences of the strategies which were claimed to be appropriate in this situation:

1. The listener sometimes makes incorrect pragmatic interpretations.

The reason for this (and also for the second consequence to be mentioned below) is the same as for the corresponding phenomenon noted in Situation 4 in connection with projected bias: L's pragmatic interpretations are based in part on false assumptions – in this situation, the projected ascriptions. An example is L's interpretation, after S's first answer, that S can't have either especially strong or especially weak interest in a management job, since a naive and objective applicant would have mentioned either of these facts. In reality S's interest in management is strong, but she didn't consider this to be worth mentioning. (Note that the second question, if formulated a bit differently, could have brought this misinterpretation to light and thus exposed the discrepancy between S's actual and projected ascriptions.)

A difference with the case of projected bias is that incorrect pragmatic interpretations may now be somewhat unfavorable to S – e.g. in the present example a favorable possibility is ruled out. But such misinterpretations arise because S did not consider it worth the effort to mention a given favorable fact, so they can never have especially grave consequences for L's actual impressions.

2. The speaker will sometimes be unable to say what she would like to say.

After Question 3, for example, S feels it necessary to suppress the additional comments produced in Column 1, since each of them would be inconsistent with her projected ascriptions. Since L doesn't expect much information to be volunteered on such items, he will not make unjustified pragmatic interpretations of the resulting silence (though he may correctly rule out some particularly extreme possibilities, especially if the item involved is strongly focused on, as with Question 3).

8.6. General Issues Raised

Even more than in the previous situations involving discrepancies between actual and projected biases, the present situation calls to mind the metaphor of dialog participants as actors playing roles which may bear little relation to the way they actually are. This notion raises a great variety of questions, many of which have been examined by sociologists (e.g. [GOFF59]) and social psychologists (e.g. [SNYD79, TETL85]). The present analysis addresses only a limited subset of these issues: the specific strategies used by speakers to maintain a given image in a certain sort of dialog situation and the consequences of such behavior in terms of the information conveyed and the (sometimes incorrect) conclusions drawn by listeners.

There are of course circumstances in which S would want to eliminate rather than maintain a discrepancy between her actual and projected ascriptions; and even if she does consider discrepant projected ascriptions desirable, she may not be so fortunate as to find herself in a context which causes the listener to expect these ascriptions from her at the beginning of the dialog. The basic strategy discussed in Situation 6 for establishing a projected bias can in principle be adopted in such cases. For example, S could presumably establish realistic projected ascriptions by

spontaneously mentioning her plans with respect to children (as in Column 1); or she could come across as naive by mentioning how interested she was in a management job (instead of ignoring the issue, as in Column 2). It would be interesting to examine, much as in Situation 6, what sorts of image-establishing comments would emerge given various relationships between the standards S actually ascribed to L and the ascriptions she wanted to project. A fundamental limitation is that this strategy is only useful for establishing a reasonably coherent image such as the realistic or naive ones in these examples. That is, if L draws a conclusion about S's ascriptions with respect to a particular item, L must feel that he is able to infer S's ascriptions for other items. Where this is not the case, there is no easy way for S to establish projected ascriptions for a number of items at once.

9. Situation 8: Uncertainty in the Listener About the Speaker's Ascriptions

9.1. Definition of the Situation

In all previous situations it has implicitly been assumed that the listener thought he knew precisely what evaluation standards the speaker ascribed to him. We now postulate, more realistically, that L has a rather vague idea of S's ascriptions. These vague beliefs may be systematically discrepant with S's actual ascriptions, as in the previous situation; but to avoid certain complications we must assume that L does not contemplate the possibility of such a systematic discrepancy. That is, L thinks that any error he may make in guessing S's ascriptions with respect to one item is independent of errors with respect to other items.

9.2. How the Situation Can Arise

It is not difficult to imagine how L might experience this sort of uncertainty, but a single illustration with respect to the examples in Figure 11 may be useful. Suppose that L believes that S has based her ascription of evaluation standards to him on her previous experience with other potential employers who (like L) gave the impression of evaluating women and men in the same way. Then L will feel sure that S is naive with respect to items like those in Figure 11. Still, unless L is thoroughly familiar with the behavior of such employers and thinks that S is as well, L will not have a very definite idea of what ascriptions S has arrived at for the individual items – e.g. what S believes to be the relative importance of the various items that concern personal commitments.

9.3. Appropriate Strategies

S must now take into account the following considerations when anticipating L's pragmatic interpretations: Since L is uncertain about S's ascriptions, he will often

be uncertain as to what comment S would make on a particular item if a particular possibility were realized. Yet it is only sensible for L to rule out a given possibility by pragmatic interpretation if he is quite certain that S would have said something different if that possibility were realized. The speaker should therefore only anticipate that a possibility will be ruled out by pragmatic interpretation if some comment other than the one she made is so obviously more desirable for a speaker with her projected ascriptions that L will be convinced of this in spite of his uncertainty about S's ascriptions.

As noted in the previous situation, S should use preparatory expressions in a way which is consistent with her projected ascriptions. Since these are now indefinite, S can take into account the fact that a particular expression will seem inconsistent to L only if L would have felt quite certain that S would not use that expression in that context.

9.4. Realization in the Simulation Model

In the previous situation, the representation in the system of discrepancies between actual and projected ascriptions allowed these to differ only in terms of the weight attached to each item. Similarly, it is sufficient here to represent uncertainty in L only with respect to the importance that S attaches to each item. The degrees of L's uncertainty can in principle be different for each item, but the examples are easiest to interpret if we assume that it is a constant percentage (50% in the present examples) for each item. Specifically, each item now has associated with it, instead of a single weight reflecting its projected importance, an interval which reflects all of the degrees of importance that L thinks the item is likely to have. Thus for the items whose projected importance is greater than their actual importance, the weight of 3.0 is replaced by the interval (1.5 4.5), and for the items where the reverse is true, the previous weight of .33 is replaced by the interval (.17 .5).

This replacement of weights by intervals requires the following modification in the implementation of the procedure for anticipating pragmatic interpretations: When a comment under consideration (C) is being compared with an alternative comment (C'), each one can no longer be viewed as having a single rating of desirability according to the projected ascriptions, but rather a range of possible ratings. C' then only rates higher than C if **each** of its possible ratings is greater than the corresponding rating of C (it is sufficient to make the comparisons only for the corresponding endpoints of the intervals). It is this use of intervals that implements the specification in Step 2 of the procedure at the bottom of Figure 5 that it be "clear to L" that the alternative comment would rate higher.[10]

During the generation of the examples for the first seven situations a slight amount of uncertainty on the part of L was actually assumed in the simulation

[10] The situation is actually more complicated when the focus of the question is a single item and a comment is being selected on some other item. Then it is not only L's uncertainty about the latter item that must be taken into account, but also his uncertainty about the focal item, which affects the relevant cost- and benefit-magnification factors. The condition is then that C' must rate higher than C according to all combinations of weights for the two items.

model, since otherwise the model would have anticipated some pragmatic interpretations that are so implausible as to be distracting.

When deciding whether to use a preparatory expression, S now does so if the criteria for its use are satisfied given at least one of the weights in the interval considered by L.

9.5. Resulting Behavior

1. S will now less often expect L to derive a pragmatic interpretation of a comment which is narrower than its straightforward implication.

Comparing Column 3 of Figure 11 with Column 2, we see that L no longer makes the second pragmatic interpretation after Question 1 or the first one after Question 3. These pragmatic interpretations would require quite precise assumptions about S's ascriptions.

2. Some types of pragmatic interpretation can be expected even when L has only a vague idea of S's ascriptions.

Note that S expects her first comment (that she never got much leadership experience) to give rise to the same pragmatic interpretation in all of the conditions in Figure 11. The derivation of this pragmatic interpretation involves a comparison between two comments which require approximately the same amount of effort: S could just as easily say that she had "no leadership experience" if this were the case. Since this more precise statement would yield more benefits for an objective speaker virtually no matter what specific evaluation standards she ascribed to the listener, L need not have a definite idea of S's ascriptions in order to interpret the fact that the statement was not made.

More generally, whenever one possible comment conveys more information than an alternative one and requires the same or a smaller amount of effort, one needs to make only weak assumptions about the value of the information conveyed in order to predict that the more informative one will be preferred by a cooperative speaker if both are true. Cases like this are often discussed in analyses of conversational implicature (e.g. [GAZD79, chap. 3, GRIC75]). By contrast, where the more informative of two comments involves substantially more effort – in particular, when the less informative one is silence – L can only be sure that the more informative one would be preferred if he can assess the additional benefits it would bring for the speaker and weigh these against the additional effort required.

3. S will now less frequently have to reject a comment because of a fear that it might be inconsistent with her image.

This consequence is a corollary of the first one: Since L generally makes fewer pragmatic interpretations, he is less likely to find a given comment inconsistent with his (vague) image of S. Thus the two additional comments after Question 3 can now be volunteered just as they were in Column 1: even though L wouldn't actually predict that a naive applicant would make such comments, he now thinks that this deviation from his expectations might be due to his own ignorance of the standards that this particular naive applicant ascribes to employers.

4. There are some desirable comments that S cannot make even if L's uncertainty is very substantial.

This is a corollary of the second consequence. After Question 1, for instance, S is inclined to say that she has "some leadership experience"; but since a more informative comment would be available, at no extra effort, no matter what amount of experience she had, she would see, that this comment would betray her positive bias.

The first and third consequences can be summarized as follows: Uncertainty on the part of L about S's ascriptions tends to cancel the effects of a discrepancy between the actual and the projected ascriptions. This is illustrated by the fact that Imp's behavior in the present situation (Column 3) is almost identical to its behavior in Column 1, where there was no discrepancy. In general, of course, behavior in these two conditions can differ more sharply, if L has less uncertainty or the discrepancy between the actual and the projected ascriptions is greater.

The following consequence of L's uncertainty concerns S's attempts to establish a projected bias, as in Situation 6. It is illustrated by the examples presented earlier, in Figure 10, which were run assuming the same degree of uncertainty on the part of L as in the present situation.

5. It is generally easier for a speaker to establish a given projected bias if the listener is uncertain about the speaker's ascriptions.

We have just seen that S generally has more freedom in choosing comments when L is uncertain about her ascriptions. This applies as well to S's attempts to find bias-establishing comments. For example, the four comments marked with asterisks in Figure 10 are only acceptable because L is so uncertain as to how S assesses the importance of the various items that he will hesitate to judge the comments to be peculiar and uninterpretable. If L had more definite ideas about S's ascriptions, either S would have to make a less desirable (but more plausible) comment or she would find no way at all to establish her desired projected bias immediately.

But if L's uncertainty makes it easier to appear consistent with her desired projected bias, does it not also make it more difficult for her to appear **in**consistent with her **un**desired projected bias(es) – an equally essential condition? This is true in some cases. For example, the fact that S's first comment in response to Question 5 in the middle column is inconsistent with positive bias is not at all obvious; if L were a bit more uncertain about S's ascriptions than the simulation model was in these examples, this comment would not convince him of S's objectivity.

But very often bias-establishing comments are inconsistent with the undesired projected bias(es) in ways which can be recognized without particularly definite beliefs about S's ascriptions. First, vague comments are almost always inconsistent with objectivity if a more precise comment is available at no extra effort. This is why the third comment in the right-hand column of Figure 10 can be easily recognized to be inconsistent with objectivity even though it is not obvious that it is consistent with negative bias.

Second, nonobligatory comments which cause downward impression shifts are almost always inconsistent with positive bias, as noted in Section 3.5. In order to recognize that the first and second responses in the middle column and the third one in the right-hand column are inconsistent with positive bias, therefore, L essentially only has to know enough about S's ascriptions to recognize that S expects these comments to shift his impression downward (in these examples, this expectation of S happens to be expressed anyway in her use of a preparatory expression).

In short, it is generally less of a problem for a speaker to appear inconsistent with an undesired image than it is for her to appear consistent with the desired one. This is why it is easier to establish a desired image if the listener is tolerant in his judgments of what behavior is consistent with a given image.

9.6. General Issues Raised

These consequences suggest that it may often be sensible for S to try to ensure that L has only a vague idea of S's ascriptions. Thus in the present examples, the applicant might try to appear generally unfamiliar with the usual evaluation standards for applicants for this type of work. This would cause L to be uncertain about S's specific ascriptions and thus make it easier for S to appear naive with respect to the interviewer's application of a double standard.

The definition of this situation stipulated that L's uncertainty did not involve coherent patterns involving several items, such as the question whether S was realistic or naive. This sort of uncertainty can of course occur, but it introduces considerable additional complications for S which cannot be explored in detail here: sooner or later, S is likely to say something that is inconsistent with at least one of the images L considers possible; this will narrow down L's expectations as to what she will say on other items and maybe even cause L to reinterpret what she has already said.

We hypothesized that L takes into account his uncertainty about S's ascriptions by only making a pragmatic interpretation if he is quite sure that it is justified. Listeners presumably also make probabilistic pragmatic interpretations of the following sort: "I'm pretty sure she would have said something like C' instead of C if the possibility P were realized, though C is quite a likely comment if P is not realized; so probably P is not realized, even though is reasonably likely a priori." A formalization of this sort of reasoning, using the principles of Bayesian inference, can shed further light on the basic logic of pragmatic interpretation. But such an analysis requires much stronger assumptions about the nature of L's uncertainty than the ones made here, and it cannot be integrated easily into the present simulation model.

10. Situation 9: Uncertainty in the Speaker About Her Projected Ascriptions

10.1. Definition of the Situation

S once again assumes that L has definite beliefs about her ascriptions, but now S is uncertain as to what these beliefs are. In other words, S suspects that there is a discrepancy between the actual and the projected ascriptions, as in Situation 7, but she is now unsure about the nature of the projected ascriptions. As before, S wants to behave consistently with the projected ascriptions – whatever these may be.

10.2. How the Situation Can Arise

One of the many ways in which this situation can arise is a variant of the case used in the examples for the previous two situations: S has realistic ideas about the standards that L is applying, and she thinks that L may know this, but she also thinks that he might consider her ascriptions to be naive. In other words, she does not know whether her projected ascriptions are realistic or naive.

A different sort of pattern is worth mentioning because of its frequency of occurrence: the case where a speaker wants to appear more familiar with the accepted norms in a given community than she actually is. For example, if for a given type of work there are certain generally accepted criteria for evaluating candidates, a job applicant who gives the (false) impression of being thoroughly familiar with this type of work will be expected to know these standards. Thus the projected ascriptions will be the generally accepted criteria, but the applicant will not know what they are. All of us have presumably been caught up in this sort of situation when asked to comment, for example, on a work of art of a sort that we are not accustomed to talking about.

10.3. Appropriate Strategies

It will be convenient to refer to any set of evaluation standards that – as far as S knows – L might think that S ascribes to him, as a possible **set of projected ascriptions**. Strictly speaking, S must in general reckon with an infinite number of these sets. A reasonable general strategy is to consider some small number of them which are sufficiently representative, in the sense that behavior which is consistent with each of this small number of sets is likely to be consistent with all of the other possible sets. What (and how many) sets of ascriptions should be used as representatives depends on the structure of the space of possibilities that S must reckon with; it is therefore difficult to give any general rules for choosing them. (For the examples to be discussed below, a pair of extreme sets of ascriptions is sufficiently representative.)

When considering a possible comment, S should anticipate its pragmatic interpretation according to each of the representative sets of projected ascriptions. Since the pragmatic interpretation may be different according to each set, S will sometimes be unsure as to how a given comment will be interpreted and will therefore have to keep track of more than one possible interpretation.

If the comment would be inconsistent with even one of the representative sets of projected ascriptions, it should not be made, since it is then possible (though not in general certain) that it would be inconsistent with S's image.

For the same reason, the use of a preparatory expression should be avoided if there is some set of possible projected ascriptions according to which it would be inappropriate.

10.4. Realization in the Simulation Model

As in the previous two situations, the implementation is restricted to sets of projected ascriptions which differ from the actual ascriptions only in terms of the weight attached to each item. The system handles this situation as if it were addressing two listeners at the same time, for whom the projected ascriptions were as different as possible – e.g. one listener who thought that S's ascriptions were naive and one who thought they were realistic. The projected ascriptions for each item are no longer represented by a single weight interval, as in Situation 8, but by an ordered pair of intervals, one interval for each possible listener. For example, each of the "unimportant" items in Figure 11 is associated with weight intervals of (.9 1.1) and (2.7 3.3) for listeners to whom realistic and naive ascriptions, respectively, are projected. (A small amount of uncertainty in L – 10% as compared with the 50% of Situation 8 – is assumed for the examples in Column 4, since the situation is even more difficult for S to manage if no such uncertainty at all is assumed.)

The system implements the strategies mentioned above by anticipating pragmatic interpretations, and checking the appropriateness of preparatory expressions, using each weight interval separately.

10.5. Resulting Behavior

1. The speaker will sometimes be unsure as to what pragmatic interpretation, if any, will be made of a given comment.

This happens after Questions 1 and 3 in Column 4 of Figure 11. In each case L is expected to make a pragmatic interpretation only if the item involved is important according the projected ascriptions.

2. The speaker must sometimes suppress desirable comments even while realizing that they might be perfectly acceptable to L and that their suppression might in fact give rise to undesirable pragmatic interpretations.

The reason why S gives a minimal answer to Question 3 is that the comments she would like to make would sound strange if her projected ascriptions were naive, as we saw in Situation 7 (Column 2). But now, S must also reckon with the possibility that her projected ascriptions are realistic, in which case her silence will be interpreted to mean that there is nothing worth mentioning about the three items which are most relevant to the question. In other words, S's attempt to maintain more than one image at a time tends to make silence the most attractive option (even more so than in Situation 7), since it is the response which is generally the most compatible with conflicting images. But since L is not aware that this is the reason for S's silence, he will sometimes draw incorrect and unfavorable substantive conclusions.

This general pattern is especially familiar in the case of a speaker who is uncertain about the accepted standards of a community: the natural coping strategy is to say as little as possible, except on matters where even a vague notion of the prevailing standards is sufficient to verify that a given comment is acceptable.

3. In some cases there will be no behavior available with which the speaker can be sure of maintaining her image.

Sometimes even silence is not consistent with all possible sets of projected ascriptions. For example, after the obligatory yes/no answer to Question 4 in Figure 11, a realistic speaker would be expected to eliminate the remaining uncertainty on the same item by saying some more about her boyfriend's attitude, whereas a naive speaker would be expected to consider further commentary superfluous. This predicament is not quite as serious as the one illustrated at the end of the example dialog for Situation 5 (Figure 8), in which S could not avoid behaving inconsistently with her image: S now has a chance to escape this fate by guessing the projected ascriptions for the critical item correctly. Exactly how S should handle this situation is hard to say in general (it depends, among other things, on whether the projected ascriptions for the different items are systematically related). Just one of many possible strategies is implemented in the Imp system and illustrated in Figure 11: the system chooses the most desirable comment which is consistent with at least one image and gives up on trying to maintain the images with which that comment is not consistent. In these examples, this means opting for the "realistic" image and producing not only a comment on the item asked about but also a second one which S would actually have liked to volunteer while responding to the previous question. It is interesting that this second comment has to be kept more vague and less positive than S would like, since S is constrained by the pragmatic interpretations that L made while S was still being cautious: if S stated that she never intended to have children, this would surprise even a listener who expected her ascriptions to be realistic, since he would wonder why she hadn't volunteered this important information in response to the previous question.

10.6. General Issues Raised

This last example dialog calls to mind an attempt to navigate in densely mined waters, and it may have struck the reader as rather implausible that a speaker should experience such difficulty in maintaining her image while responding to four simple questions. But recall that the situation as defined has an asymmetry which makes it exceptionally challenging for S: she has considerable uncertainty about L's expectations, yet she thinks they are quite definite and wants to conform to them whatever they might be. In a more symmetric situation, presumably more frequent in everyday dialogs, S's uncertainty about her projected ascriptions is similar in extent to L's uncertainty about her ascriptions. We need not discuss this situation at length, since the results are predictable on the basis of the discussions of Situations 7 and 8. Just as L's uncertainty in Situation 8 largely eliminated the effects introduced in Situation 7 by the **discrepancy** between S's actual and projected ascriptions, in this new situation L's uncertainty tends to eliminate most of the complications caused by S's **uncertainty** about her projected ascriptions. In fact, the simulation model's responses to the questions in Figure 11 differ only slightly in this new situation from those seen in Columns 1 and 3; in particular, the predicament at the end of Column 4 does not arise. In sum, the lessons to be learned from Situation 9 concern not only the nature of the behavior it tends to elicit but also the conditions which are critical to the situation's definition.

A situation analogous to the present one can arise if S is initially uncertain about her projected bias (as opposed to her projected ascriptions) and wants to behave consistently with it whatever it is (as opposed to trying to establish a particular projected bias, as in Situation 6). This can happen, for example, when S is trying to fit into some community which has certain generally accepted biases which are unknown to S, e.g. toward particular political figures. It would be interesting to investigate systematically what sort of comments a speaker produces when asked her opinion of some controversial figure without knowing whether she was expected to display positive or negative bias. Presumably the difficulty of maintaining her image would incline S to say as little as possible; there would be few opportunities for S to attempt to create a favorable or unfavorable impression of the person in question in accordance with her actual bias.

11. Conclusions

We can look at the issues discussed in this chapter from another perspective by returning to the more practical problems of human-computer communication raised in the introduction. The above analysis of human dialog raises a number of questions about the design and use of user-modeling systems. Some tentative answers to these questions are also suggested by the analysis, and by the Imp simulation model; but these suggestions are obviously subject to revision on the basis of future research which examines interaction with such systems directly.

Whereas the above analysis has in effect taken the point of view of the system in human-computer dialog, the discussion to follow is written with the interests of the user in mind. In particular, it presupposes that discrepancies between the system's actual motivation or user model and the image it projects of them are undesirable and should be minimized.

The following two sets of questions and suggestions are directed at two categories of persons who can contribute to the reduction of such discrepancies: those who design such systems and those who use them (or who educate or inform the users.)[11]

11.1. Questions for Designers of User-Modeling Systems

The most obvious way of ensuring that users have an appropriate understanding of a system's user model and motivation is to describe the system clearly in documentation (either printed or on-line). But there are severe limits to what can be accomplished in this way. On the one hand, users often lack access to a system's documentation or don't bother to consult it (especially with systems like those of [CHIN*, QUIL*]). On the other hand, even the most skillfully written documentation cannot convey a clear picture of a system whose structure and behavior are

[11] Since the roles of the system and the user correspond to those of the speaker and the listener in the main analysis, masculine pronouns will be used in this section to refer to the user.

intrinsically difficult to understand. Attention will thus have to be devoted to the following questions:

1. What types of user model and dialog motivation are in general easiest for users to perceive accurately?

A generally accepted principle in the literature on human-computer interaction is that even quite a complex system can be comprehensible if it corresponds to some familiar metaphor. As can be seen from the chapters in this book, most user-modeling dialog systems are based on the metaphor of the human dialog participant. In fact, it is difficult to think of an alternative metaphor for systems which include user models, since human beings are the only familiar objects which construct and exploit complex models of people. Moreover, the metaphor of the human dialog participant, to the extent to which it is reasonably accurate in a given context, has an advantage not shared by metaphors involving inanimate objects. The user can not only make use of his extensive experience with other dialog participants; he can also imagine what he himself would do in a given situation in order to predict and interpret the system's behavior. In empirical studies concerning the everyday prediction of the behavior and beliefs of other persons, the predictor's self-knowledge is typically found to be a major determinant of the predictions, often more important than all of the predictor's independent beliefs about other persons (see e.g. [NIKU85] and the further references given there).

As with any metaphor, it is important that there be an easily recognizable boundary between the respects in which the system behaves like a human dialog participant and those in which it does not. Otherwise overapplication of this metaphor will lead to incorrect expectations, misinterpretations, and puzzlement (cf. [SHNE87, pp. 322-325]).

In particular, user modeling techniques or forms of dialog motivation which have no analog in everyday human dialog are problematic, even if these features seem especially effective or otherwise appropriate in their own right. The designer should consider whether such a deviation from the basic metaphor can lead to incorrect expectations and interpretations by the user.

Although the Imp system was not designed for practical use, it can serve as an example of a user-modeling system which is basically easy to understand and predict (except when it is attempting to project an inaccurate image). The criteria according to which it selects factual comments approximate human behavior quite well, and the many respects in which its behavior deviates from that of human dialog participants (e.g. its restriction to the production of purely factual comments) are easy to recognize and to describe.

Whether or not a system's basic design makes it intrinsically relatively easy to understand, there will often be cases where the user begins the dialog with specific false assumptions about its user model or motivation, or an amount of uncertainty about it which constitutes an obstacle to effective interaction. The considerations mentioned in Section 7.3 suggest why it is desirable to make the user's perceptions more accurate as soon as possible within the dialog. One general approach to achieving this is addressed by the following question:

2. How should the system behave so as to project an accurate image of its motivation and user model?

In Situation 6, one general strategy for establishing a given image was examined, namely that of behaving consistently with that image and inconsistently with any likely alternative image. This strategy can in principle be applied by user-modeling systems, but its limitations must be borne in mind: depending largely on contextual factors beyond the system's control, the strategy can often only be applied, if at all, at the expense of other dialog goals – even if the image to be established is an accurate one.

An alternative approach is to take advantage of the extremely broad bandwidth of computer-human communication to convey to the user explicit information about the state of the system. As a rough initial example of what can be accomplished in this way, the Imp system uses a dynamic display on a relatively simple semigraphic terminal. For a number of items in the evaluation form there is a continuous graphic representation of the impression that the user is expected to have, the changes in the impression that the system would like to bring about, and the degree of relevance of the item to the topic of the most recent question. With techniques such as windows and high-resolution screens, it should be possible to make such displays available to the user in a way that is informative yet unobtrusive.

A further question calls attention to a different approach to the same problem:

3. Under what circumstances should a system adapt to its user's expectations about itself instead of trying to influence them?

Consider, for example, a system which is usually objective and which has quite an extensive model of its user, but which somehow infers that the user expects it to be positively biased and to lack certain types of information about him. The simplest way for the system to resolve these discrepancies might be to conform to the user's expectations by adopting a positive bias and refraining from using the information that it was not expected to have. To be sure, this could result in behavior which was less helpful to the user than it might be; but on the other hand it would minimize the chance of incorrect pragmatic interpretations. In the terms used in our analysis of evaluation-oriented dialog, this amounts to using the projected bias and the projected ascriptions not only to check pragmatic interpretations but also to evaluate the desirability of possible comments in the first place.

Regardless of the extent to which it proves possible to ensure that the user's image of the system is appropriate, the following question should be considered:

4. In what ways should the system try to anticipate specific pragmatic interpretations on the part of the user?

If the user's image of the system is basically accurate, the main purpose of anticipating pragmatic interpretations is to keep track of the user's specific beliefs, so that, for example, the system will not volunteer information that the user has already inferred. To be sure, as noted in Section 5.6.3, this strategy is unproblematic only if the user is aware that his pragmatic interpretations are being anticipated in this way, since otherwise he will have an inaccurate idea of the specific beliefs that the system ascribes to him. On the other hand, if the system has not managed to ensure that the user has an accurate image of its motivation and its user model, it may be important to anticipate possible misinterpretations which can have serious consequences. What the system should do when it thinks that its behavior would lead to such a misinterpretation depends on the approach taken to the three questions just discussed.

11.2. Questions for Users of User-Modeling Systems

It is unrealistic to expect that user-modeling systems will always embody satisfactory answers to the four questions just discussed. Aside from the difficulty of the questions, there is the problem that designers will not necessarily share the basic goal of ensuring that the user has an appropriate model of the system. Users will therefore have to accept part of the responsibility for pursuing this goal.

5. How can the first-time user of a given system form realistic prior expectations about the system's user model and dialog motivation and about the sort of image that it is likely to try to project?

As far as the user model itself is concerned, the user should have a general understanding of the basic difficulties involved in user modeling and of the state of the art. If the user is to have realistic expectations about the system's dialog motivation and about the image it will try to project, it is useful for him to have some understanding of the reasons why certain combinations of actual and projected parameters may seem more attractive than others to the designer (or to the system itself). Unfortunately, the considerations involved here can be quite complex, as was shown by the discussions above with respect to everyday dialog. For instance, in the examples in Situation 7, the speaker's goal of projecting certain ascriptions with the aim of being diplomatic conflicted with her goal of making a favorable impression; and it was pointed out in Section 7.2 that a speaker's decision as to what sort of bias to project can depend on the specific properties that the Object in question has – which are of course not known in advance to the listener. A realistic user, who is aware of these complexities and of the fundamental difficulty of user modeling, will therefore often not have very definite prior expectations about the system. A consequence of this open-mindedness is that he will initially make relatively few pragmatic interpretations of the system's behavior and thus will tend to avoid misinterpretations (cf. Situations 8 and 9).

6. How can the user form a more definite and realistic image of the system on the basis of its behavior?

Even if the system does not provide explicit information about its user model and motivation (cf. the second question above), the user can in principle draw useful conclusions about these aspects of the system on the basis of the system's dialog behavior. To be sure, this is only possible if the user is sufficiently sophisticated about any image-maintenance strategies that the system might employ. In the analysis of evaluation-oriented dialog, the speaker's attempts to project a bias different from its actual bias presupposed that the listener was not aware of the possibility that she might do this or of the way she would go about it. If the user's degree of sophistication matches or exceeds that of the system, he may be able to avoid being misled; but if he is to be able to draw definite conclusions, he may need to devote some effort to active exploration of the system. This leads us to the final question:

7. How can the user experiment with a user-modeling system in order to determine its motivation and the nature of its user model?

One often sees novice users of systems such as text editors engaging in behavior which is aimed primarily at revealing the properties of the system, rather than performing some task with the help of the system. This sort of exploratory behavior

can require considerable insight and skill with user-modeling systems, especially if the system (unlike the Imp model) tries to interpret the user's behavior in terms of the user's plans and goals or (like Imp) sometimes attempts to project an inaccurate image. Experience in interacting with such systems can increase a user's skill at this sort of exploration. For example, with the Imp model it is an instructive exercise for a user to take the role of the listener and to try to pose a sequence of questions which will lure the system into a situation in which the discrepancy between its actual and projected dialog motivation will be exposed (as at the end of Figure 8).

11.3. Concluding Remarks

The reader may be reluctant to accept the idea that the practical use of user-modeling systems makes it necessary to think about the questions listed in this section. On the other hand, we should note that all of these questions have analogs that must at times be dealt with by speakers and listeners in everyday dialog; and it is unrealistic to expect that we can endow dialog systems with a sort of understanding which is characteristic of human dialog without having to deal in some way with the associated complications.

An encouraging thought is that, although these questions cannot be ignored entirely, the answers to them will sometimes be simpler than they are in connection with human dialog. Unlike human dialog participants, user-modeling systems can be designed from the start so as to be as comprehensible as possible (cf. the first question above); and if this is accomplished, the remaining questions tend to become either less important or easier to answer satisfactorily. In addition, computer technology offers some specific solutions that are not available in human dialog (as noted in connection with the second question). But note that these potential advantages of user-modeling systems over human dialog participants will not follow automatically from the fact that the systems are not human beings (as was presupposed by some of the views discussed critically in the Introduction). The advantages can only be realized if those who design user-modeling systems (a) accept the goal of ensuring an accurate perception of their system by its users and (b) understand the complexities of human dialog sufficiently well to be able to achieve this goal.

Acknowledgements. Thanks are due to H. Bego, E. Hovy and R. Meertens for helpful comments on earlier versions of this paper and to the informant who supplied the information required by the simulation model to produce the examples. The participants in the Maria Laach Workshop also provided useful feedback, and the volume editors created exceptionally supportive conditions for the authors.

A brief, early description of the simulation model discussed in this paper was given in [JAME83], and a shorter version of the material in Sections 2 through 6 appeared as [JAME87].

12. References

[AJZE71], [ANDN81], [CARB*], [CARO83], [CECI84], [CHIN*], [CLHH81], [EAGL81], [FINI*], [GAZD79], [GOFF59], [GREW83], [GRIC75], [HARR63], [HINT75], [HOLL86],

[HOVY86], [JAME82], [JAME83], [JAME84], [JAME87], [JONE65], [JOSH82], [JOSH84a], [KOBS85a], [KOBS*], [MACL82], [MCCL78], [MCCO*], [MCKE85b], [MORI*], [NIKU85], [NIRO80], [NORM86a], [PARI*], [QUIL*], [REIC78], [RICH*], [ROWE84], [SAMP82], [SCHW79], [SHNE87], [SIKL78], [SNYD79], [SPAR*], [TETL85], [THMA83], [VAND81], [WALS66], [WARD74], [WEYD79], [WINO77], [WINO86], [WYER74]

11 Incorporating User Models into Expert Systems for Educational Diagnosis

Robin Cohen and Marlene Jones

Abstract

In this chapter we study a particular real-world domain, that of educational diagnosis. We argue that expert systems for educational diagnosis require user models, and that these user models should include several components, including the user's background knowledge of both the student and the domain as well as the user's goals. Our proposal is directed at enhancing the particular expert system of the CGD project. We then propose an architecture for this expert system that separates the knowledge base into relevant components and includes a user model. We further demonstrate that this divided model for the system facilitates providing the best response for a particular user, according to his background knowledge of the domain and of the student and his goals. Finally we argue that the techniques outlined here will be useful in general in expert systems, to vary the response to the user at hand.

1. Introduction

There are many reasons for including a model of the system's user within an expert system. The foremost is to allow the system to tailor its explanations to meet the particular user's requirements. These explanations may concern the gathering and entering of data, the use of the system itself, or the system's findings, recommendations and reasoning process. One may argue that it is possible to alter the system's explanations without going to the effort of including an elaborate user model or any user model at all. For example, one can simply store predetermined explanations with varying amounts of detail and allow the user to request further (i.e. more detailed) explanations, if desired. This is unsatisfactory in situations where one would like to vary several aspects of the explanations simultaneously, which is the case in educational diagnosis, the application domain with which we are concerned.

We provide examples to defend the need for user models in our expert system, and then address the issues of what information to incorporate into a user model, how the model can be derived and updated, and how the model can be used to provide explanations. In short, we argue for the feasibility and usefulness of modeling (not just the student but the expert system user) for educational diagnosis.

2. Application Domain

This chapter studies a particular real-world domain – educational diagnosis. When a student is experiencing learning difficulties, it is important to ascertain the exact nature of the problems as quickly as possible. This may involve an in-depth educational diagnosis with the ultimate aim of developing a remedial program which is tailored to that particular student. An in-depth diagnosis/assessment generally includes examining the student's developmental history including the home environment and early school history; assessment of psychoeducational correlates such as intellect, visual and auditory acuity; language skills; achievement within fundamental areas such as reading, arithmetic and spelling; medical and behavioral information as well as a detailed assessment of any achievement areas in which the student's performance is deemed to be inappropriate. Usually such an assessment is carried out by a psychologist and/or resource room teacher. The resultant diagnostic report is generally made available to the regular classroom teacher and any other participants in the remedial program, as well as to the necessary administrators such as the principal. Of course, the diagnostic findings and educational decisions should be discussed with both the student and parents.

There are well-accepted guidelines and procedures regarding educational diagnosis [COMM71] as well as appropriate models; for example, see [ADEL82, MCLE82] and the references therein. It is rare, however, to find educational diagnosis carried out as thoroughly and rapidly as the guidelines suggest. This is largely due to insufficient resources, particularly personnel such as highly experienced resource room teachers.

One means of facilitating educational diagnosis within the regular school environment is via an expert system to guide a teacher (such as the resource room teacher) through the various stages of diagnosing learning difficulties. This is the long-term goal of the CGD (Computer-Guided Diagnosis) Project based at the University of Saskatchewan [MCLE85]. Preliminary systems have already been developed in the area of reading [COLB82, COLB83, COLB84] and arithmetic [JONE86, TUBM86], with a more elaborate, comprehensive system under development [JONE87, MCLE86]. It is this latter expert system that we consider in this chapter.

The advantages of such a system are many. It will serve as an information source to experienced diagnosticians, an assistant to less experienced resource room teachers, and a tool for training teachers. Even during the handling of one particular case, there may be a variety of users at each of the several stages of the diagnostic process – in-depth assessment, development of a remedial program, and monitoring the progress during remediation, which generally includes further assessment and may include placement and funding decisions. Furthermore, one may want to make some portions of the system's findings and suggestions available to either the parent or student. Hence, such an expert system may be used by or required to supply explanations to any of the following individuals: psychologist, resource room teacher, educational consultant, regular classroom teacher, special education teacher, principal, outside administrator such as the head of special education ser-

vices or superintendent, researcher, parent, the student him/herself, or an education student or training teacher.

Obviously these users vary in a variety of factors, most notably in their *background regarding the field* of special education. However, even personnel with a strong background in special education will vary immensely in regard to their familiarity with testing procedures, knowledge of individual tests (some of which require special training in order to administer), knowledge of areas of assessment such as reading, arithmetic, language etc., administrative procedures and guidelines, laws and mandates, outside resources, funding criteria and availability etc.

Individuals will also vary in their *familiarity with the individual case*, such as knowledge about the medical history, the home environment or the current school behavior. For example, the student, parent and teacher are familiar with the situation on a day-to-day basis whereas the psychologist may have a one-page referral form and the principal may have a short synopsis of the recent observable problems. Similarly, the "official" opinion on the case may be available to a select few. Some individuals, such as the psychologist and resource room teacher, may be familiar with a variety of similar cases which the teacher and principal are not.

Also the *goals* of the users may vary. Although hopefully all users' main goal is the implementation of the most appropriate remedial program, the subgoals may vary. For example, the teacher's most immediate concern may be techniques for decreasing disruptive classroom behavior. The principal may be concerned as to whether the student qualifies for special funding. The parent may be searching for assurance of the student's normalcy.

How does this myriad of variations affect the expert system's explanations? First of all, if the user has a particular goal in mind, this should be addressed. Even for a particular question or subgoal, the explanation itself should vary depending on the user's familiarity with the case and background in special education. Ideally, both the amount and detail of the explanation should be altered. Most parents will not benefit from a detailed description of the student's performance on a variety of tests, nor is an in-depth description of each remedial activity appropriate. This information is, however, critical for the resource room teacher who is implementing the program in conjunction with the regular classroom teacher (often with the assistance of a consultant). The language in which the information is expressed is also critical. Most parents are overwhelmed by the use of technical terms. Straightforward, honest explanations are desired, which must include realistic encouragement for both parent and student. It is important to remember that both parent and student may have been aware of problems for several years without understanding them. In other words, one may be contending with several years of frustration, fear and maybe even mistrust. Hence, the tone of the discussion is important.

We focus in this chapter on the problem of how to set up the expert system so that different users can get different explanations. We then provide a framework for varying the content of the explanation. The topic of varying the surface form with appropriate natural language generation is left as future work.

3. Designing the System

In this section we propose an architecture for our expert system and a control structure for searching the knowledge bases of the system when producing explanations, which will provide different responses to particular users, thus making the current system available to a variety of individuals. Although this model for making the system user-specific has not yet been implemented, we provide an analysis of some sample queries to illustrate the usefulness of the approach.

One contribution of the research is thus specifically for educational diagnosis, a method for making a system more effective. We suggest extensions to the current implementations of the CGD project, as illustration. Moreover, the proposal outlined here can be viewed as a general strategy for incorporating user models into expert systems to provide clearer explanations, primarily by avoiding the presentation of material already known to the user.

3.1. Motivation for System Design

The educational diagnosis expert system of the CGD project is designed to interact with a school psychologist or resource room teacher to discuss the case of a particular student. As the diagnosis proceeds, information regarding the student's background, behavior, abilities, skills, *etc.* is collected from a variety of sources. At any point in the diagnostic process, the system analyzes the available information and employs its knowledge of educational diagnosis to draw conclusions which may include recommendations for further assessment or particular hypotheses regarding the student's difficulties. The main types of questions being posed by the user are thus: (i) what is the diagnosis for this student? (ii) why is this the diagnosis? and (iii) why is this diagnostic procedure (such as a particular test or type of test) being recommended? The system should be able to respond to similar questions posed by a classroom teacher, parent, or school administrator. Furthermore, there may be a variety of backgrounds among members of any of these "types" of users – e.g. two psychologists with different favored approaches to diagnosis.

In Example I we illustrate the need for different responses to the same question for different types of users.

Example I:

User: Could the student's mispronunciation "errors" be due to dialect?

Response to Parent: Yes, non-standard pronunciations may be due to dialect rather than poor decoding skills.

Response to Psychologist: Yes, the student's background indicates the possibility of a dialectal difference.

Here, the parent is asking for more information regarding the domain of educational diagnosis (lacks the domain knowledge). The psychologist knows about the tests employed and has the necessary domain expertise, but is unaware of the background of the student.

In Example II, we emphasize the possible difference in goals of users employing the expert system. In particular, we indicate session-wide preferences a user may bring to bear.

Example II:

System: Administer an individual intelligence test. For this particular case, the Peabody Picture Vocabulary Test (PPVT) is recommended.

User: Why is this test being recommended?

For each user, the stated goal is the same – i.e. to find out why this test is being recommended. However, the inferred or intended goal may be different. This may be due to a system-wide preference. For example:

Psychologist Preference: administer intelligence tests with which s/he is familiar (e.g. the psychologist may be more familiar with the Stanford Binet or the WISC-R (Weschler Intelligence Scale for Children – Revised)).

Response: The PPVT is being recommended because it does not require verbal responses, and the student appears to have language problems including difficulties with expressive language. Hence, the PPVT is more appropriate than more common tests such as the WISC-R or Stanford Binet.

If the user is a teacher, principal or parent, the system would respond by explaining why an intelligence test is being suggested, as well as why this particular test is appropriate. The user may have additional goals which also must be addressed, as illustrated below.

Principal Preference: employ tests which can be used in funding formulae

Response: There is no evidence to suggest that the student will require or will qualify for additional funding.

Parent Preference: minimize testing due to stress involved

Response: The student should not find this test stressful, as no verbal responses are required; answers are indicated through pointing.

The above discussion may lead the reader to believe that it is sufficient to stereotype our users and exploit such defaults when answering the user's questions. Defaults alone are, however, insufficient. For example, a parent may have substantial knowledge of educational diagnosis because of involvement in organizations such as ACLD (Association for Children with Learning Disabilities) or CEC (Council for Exceptional Children) or because of professional training within a related field. And yet, the defaults of a fully trained psychologist may be too much.

Furthermore, there are several possible reasons why a psychologist may not fit a given stereotype. For example, s/he may have additional information about the student which the system is not assuming. The psychologist may have been involved in some previous diagnostic situation with this student or perhaps knows the student from other circumstances. A second possibility is that the psychologist lacks experience within a particular area of exceptionality or assessment, such as the diagnosis of emotional difficulties. Having determined this, the system knows that additional explanations must be supplied for all aspects of assessment within

this area, not just for the user's first query. Similarly, consider the case where the student is visually impaired. Although the psychologist may be well aware as to why certain skills are being assessed, s/he may be unaware of the appropriate tests to do so, given this additional constraint.

A third type of variation from the initial stereotype is based on differing philosophies. For example, the user may adhere to the belief that visual-motor training is effective in the remediation of reading difficulties; although previously a popular educational practice, current evidence indicates otherwise. Or the philosophical difference may stem from a disagreement regarding the usefulness of certain types of information during the diagnostic process or the conclusions that can be drawn from the student's performance on a particular test. The following example illustrates this point.

Example III:

System: Administer an individual intelligence test such as the WISC-R or Stanford Binet. User instead administers the ITPA (Illinois Test of Psycholinguistic Abilities) and the student's performance is recorded.

User: What are the conclusions based on the results of the ITPA?

System: The student's intellectual ability appears to be within the normal range; no further assessment of intellectual ability is required.

User: What other conclusions were drawn? For example, the student's poor scores on the Auditory Association and the Auditory Closure subtests indicate that the student's auditory channel is weak.

System: Although the test results raise some suspicions regarding the student's processing of auditory information, such conclusions cannot be drawn from the ITPA alone because the validity of the ITPA's subtests is questionable.

This type of question from the user (drawn from a real example case) would alert the system that the user does not adhere to certain current philosophies. The system does not consider the individual subtests of the ITPA to be valid. Moreover, the user does adhere to the ITPA model and related philosophies. Because the psychologist has drawn certain conclusions which the system has not, this will affect what s/he concludes in regard to reading assessment or even which tests s/he thinks are appropriate.

3.2. Details of the System Design

In essence, we are requiring our expert system to accomplish two modeling tasks simultaneously. The system must model the student in order to produce an appropriate diagnosis. In order to provide appropriate responses to the user, it must also model the user. The system must, therefore, contain the following three knowledge bases:

(1) domain knowledge which includes knowledge regarding diagnostic procedures, practices and educational theories, information regarding standardized tests and remedial activities etc.,

(2) a model of the student,
(3) a model of the user.

We show how it is possible to present a finer gradation, useful in generating responses. For each of these knowledge bases, different types of knowledge can be identified to partition the knowledge base further. For example, domain knowledge can be partitioned into rules of educational diagnosis versus specific data about given tests. Even within the category of domain rules, there exist different types of rules.

A more detailed description of the required knowledge bases is provided below.

Domain knowledge

Rules
- diagnostic procedures, including principles and terminology
  - e.g. if possible, employ individually administered tests because individually administered tests tend to have higher reliability than group tests
- test interpretation rules
  - general
    - e.g. IF the student's score is on standard deviation below the mean on an individually administered achievment test, THEN further assessment is required
  - specific
    - e.g. IF the student's reading quotient is less than 80 on the Schonell Reading Test, THEN further assessment is required

Databases
- test database
  - e.g. information regarding purpose, subtests, types of scores, reliability, *etc.* (The system employs this information in conjunction with the above rules, particularly the general test interpretation rules, to draw appropriate conclusions.)
- remedial activities and methodologies

Student knowledge

- background
  - background facts acquired through testing (e.g. age)
  - other background: family facts, etc. (e.g. medical history)
- test results
- hypotheses
  theories of the system as to diagnosis of the student

User model

- knowledge of the domain (same division as above)
- knowledge of the student (same division as above)

- goals
  - for the session
    - domain goal (find "best" diagnosis for student)
    - preferences
      e.g. administer tests that can be used to get funding
- for the particular question
  - stated goal
  - intended goal

As mentioned earlier, during the course of the diagnosis the expert system is developing hypotheses (i.e. a diagnosis) based on the information at hand, as well as suggesting further testing in order to gather information to improve the accuracy of the diagnosis. The techniques employed in answering the user's questions regarding either the diagnosis or the recommended diagnostic procedure are essentially the same. Therefore, for the next section of this chapter, we concentrate on one type of question: "What is the diagnosis?" and illustrate our methodology for providing an appropriate answer to this query. Figure 1 highlights the system's knowledge bases and introduces a corresponding notation.

In the next section we show a control structure of the system to generate a response for a user, mobilizing appropriate knowledge bases. The main features of this strategy are: (i) default assumptions about an entire class of users (psychologists, teachers, parents, etc.) are used as a starting point for the model of the user's knowledge (ii) the system is aware that its model of the user (as well as its knowledge of domain and student) is updatable – i.e. that new information revealed during a session with a user may cause the models to be revised. The defaults provide a starting point for the modeling, and the dynamic maintenance of the knowledge bases allows for additional exceptions to the assumed model of the user.

4. The User Model Structure

In this section we advance a proposal for the acquisition and updating of a model of the user to produce quality explanations. What we want, ideally, is a system that can respond to all the examples offered in the first part of 3.1. We indicate here a proposal to isolate the parts of the system required for responses. But we do not deal with lower level generation issues, such as choosing the most appropriate terminology for a user. We will at times indicate what should ideally result; in these cases we also make clear to what extent the current framework also provides a solution to these harder questions.

4.1. Characterizing the User

We now show how to use the diagram of the system, outlined in Section 3.2, to generate appropriate responses for different users. We illustrate the procedure for the case of a user who is a psychologist. Consider, in fact, a user who is the psychologist who tested the student whose case is being evaluated by the system.

System (Sy)
- Knows About Domain (KAD)
  - diagnostic principles (d)
  - test rules (t)
- Knows About Student (KAS)
  - background (b)
    - acquired through testing (t)
    - other background (o)
  - test results (r)
  - hypotheses (h)
- Believes that the User (B)(U..)
  - Knows About Domain (KAD)
    - diagnostic procedures (d)
    - test rules (t)
  - Knows About Student (KAS)
    - background (b)
      - acquired through testing (t)
      - other background facts (o)
    - test results (r)
    - hypotheses (h)
  - Wants (W)
    - domain goal
    - domain preference
    - query goal (stated)
    - query goal (intended)

Figure 1. Labeling of system knowledge bases

The general principle used to produce a desirable response in an efficient amount of time is basically: "Don't explain what's already known, and keep track of what's already known, to avoid providing extraneous detail". This extension of Grice's maxim of quality [GRIC75] allows for some division of knowledge and tracking of shared knowledge bases.

At the start of the session, the system would evaluate the knowledge of the user, in terms of its knowledge bases, as follows:

1. SyB (UKASr) = SyKASr
   [read as: "system believes user knows about student ..."]

The system believes that what the user knows about the test results of the student is equivalent to what it knows. This assumption is reasonable as the test results have been input by the user.

2. SyB (UKASbt) = SyKASbt

The system believes that the background knowledge it has acquired about the student, specifically through testing, is equivalent to what the user knows. This

background information would be data like the student's age, public school, etc. that would be part of the information drawn during the testing stage. Note that the system may have gathered additional information about the student's background from other sources in the course of its search for a diagnosis, so there may be some discrepancy in the overall knowledge of the student's background between system and user (included in (KASbo)).

3. SyB (UKADt) = SyKADt

The system starts off with the assumption that the user is familiar with the standard interpretation of tests used in diagnosis.

4. SyB (UKADd) = SyKADd

The system also assumes that the user shares knowledge about basic diagnostic procedures.

The main features of the design are thus: (i) a coarse-grained division of the system into knowledge bases, to identify sources of shared knowledge and to thus focus on elaborating unknown information (ii) a tracking of specific information on the user's current beliefs to produce more co-operative responses within the areas where the knowledge is not shared. We illustrate these in a tracing of the system's algorithm below.

In order to describe how the system mobilizes certain components of the proposed architecture to generate better responses, it is necessary to present some model for the system's reasoning. The computational framework employed in the CGD expert system is THEORIST, a logic programming system based on default reasoning, developed at the University of Waterloo [POOL86]. For details and examples of the implementation of the CGD project using THEORIST, see [JONE87].

To clarify the discussion below, we refer to a more conventional model of reasoning in expert systems. Consider a production rule base, where the system encodes a series of "rules" of the form: "If X then Y", and may then record various "facts" of the form "X1", "X2", etc. By some deduction mechanism such as forward chaining, a diagnosis may be drawn by proceeding from the facts and rules to the most likely conclusions. The main framework which we draw for discussion is: facts + rules lead to diagnosis (F + R → D). (Note: which of the system's defined knowledge bases are invoked for the system's control structure should remain the same, regardless of the reasoning model employed.)

In our domain of educational diagnosis, the facts are items like scores on tests for student and rules are items like standard interpretations of test results for diagnosis. The model for reasoning is thus:

Facts + Rules → Diagnosis

This is expanded to:

Background Facts + Test Results + Domain Rules → *Hypotheses*

Further, Domain Rules can be divided into Test Interpretation Rules and Diagnostic Procedures, yielding the formula:

Background Facts + Test Results + Test Rules + Diagnostic Procedures
→ *Hypotheses*

where Background Facts may be further partitioned as:

Facts acquired through testing + Other facts

There are in fact two ways to use Test Rules to produce a hypothesis: a) use a specific test interpretation rule or b) use a general test interpretation rule, together with information from the test database regarding the test at hand. We put these two cases together under the label "Test Rules". Note that extensive use of the test database in fact occurs in conjunction with responses that suggest administering new tests. For future work, we can examine more closely possible sources of difference between the system and psychologist users regarding testing procedures. One strategy worth studying is segmenting the test rules portion of the knowledge base as well in order to more appropriately respond to the user.

Then, a system should respond to a question posed by a user by activating those parts of its reasoning towards a hypothesis that the user may not already share. We illustrate this focusing mechanism below.

Dialog I:

For a user who is a psychologist:

Question: What is the diagnosis for this student?

Response: *Hypothesis*

Reasoning: UKASr (test results)

UKADt (test rules)

UKADd (diagnostic procedures)

– if all Background has been acquired through testing, then UKASb (background)

– since *Hypothesis* is the only component not already shared by the user, this is included in the response

Note that after this exchange, SyB (UKASh) is updated to include the hypothesis just presented. Once more, the separation into knowledge bases allows for a current view of the user's knowledge.

Dialog II:

Question: What is the diagnosis for this student?

Response: *Hypothesis*, because background fact X is true for this student.

e.g. The system suspects that the student can be classified as "learning disabled"; there is a family history of reading problems.

Reasoning: as above, but in this case SyKASb≠ SyB (UKASb) because there is some background on the student not directly acquired through testing

Guideline: The system is compelled to add in the response relevant information that the user may not yet know.

This example indicates that for our domain the knowledge base referred to as background on the student may be further subdivided into background acquired

through testing and background acquired from other sources. Again, the principle is to divide to facilitate labeling of shared knowledge with the user.

The above dialog can also be employed to demonstrate how the system would handle Example I, as illustrated in Dialog IIb below. Recall that the user is asking about a possible hypothesis:

Dialog IIb:

User: Could the student's mispronunciation "errors" be due to dialect?

Response to Psychologist: Yes, the student's background indicates the possibility of a dialectal difference.

Reasoning: (as in Dialog II above). In addition to responding with *Hypothesis*, provide the relevant background fact not acquired through testing.

A similar approach is employed when answering the parent's query:

Response to Parent: Yes, non-standard pronunciations "errors" as measured by the Neale Analysis of Reading Test may be due to dialect rather than poor decoding skills.

Reasoning: In addition to responding with *Hypothesis*, provide the relevant Domain Rules (in this case a Test Rule).

A more detailed description of the system's evaluation of users in the parent class is now in order. Basically, the parent is assumed to know about background information on the student. So:

SyB (UKASb) = SyKASb (this covers both bt and bo)

Otherwise, the parent is assumed to be unaware of how to interpret tests and unaware of the test results, yielding the conclusion that for this user, KAD is all unknown, and KASr is also unknown.

In order to actually generate different responses for different users, it is useful to outline particular schema, which can be instantiated by the specifics of the response to the query, as follows:

for the psychologist:
(affirm (response)
state (BACKGROUND FACT not acquired through testing))
for the parent:
(affirm (response
state (TEST RULE)
state (DIAGNOSTIC PROCEDURE)
state (TEST RESULT))

Consider Dialog IIb. For the psychologist the only part of the information extracted to produce the response that is unknown is dialect, an example of background not acquired through the testing stage. For the parent, the fragment is assumed from a dialog where the parent already knows the test result, so this is known and not part of the response content which must match the schema. Likewise, diagnostic procedure does not enter into the response.

In effect, our approach for generating user specific responses is similar to the idea of "user perspective", as used by [MCCO86] – except it is the user's perspective of the system's knowledge bases, and not of particular objects, which is used to introduce the variability. Our reading of a schema combines features of McKeown's [MCKE85b] and McCoy's [MCCO86] use of the same term. In [MCKE85b], there are schema to generate responses which contain predicates, some of which are optional, which then get instantiated according to the relevant knowledge in the knowledge base to produce the actual response. In [MCCO86], the concept of schema is carried further, to also expand the predicates of the schema according to the user at hand. Instead of focusing on certain objects in a knowledge base, we focus on the content required to produce a response to a query. But then the user at hand also determines the appropriate schema (or parts of a general schema) and the actual selection of predicates. Just as perspective in [MCCO86] allows for a user model which varies according to context, so our consideration of the user is tempered by the query at hand and the highlighting of the knowledge base for that user.

This approach can be contrasted with other efforts to model users for response generation. We feel it is more effective than the double stereotyping scheme of [CHIN86] because we can classify our users at several levels (vs. just expert or novice), and we are not tied to labeling all the content of the knowledge base according to its "difficulty". Rather, whole portions of the knowledge base are labeled according to whether they are known by the user, according to his class. For expert systems environments such as CGD this classification is quite natural. For Dialog IIb, it is unclear how a labeling of "level" of knowledge base information could afford the variation in response required. It is not that information about the student's background is "less difficult to understand" for some users; rather, it is a kind of knowledge more likely to be known to a particular class of user.

Our work is also in contrast to the work done in refining explanations from expert systems (e.g. [BUCH84, SWAR83]). These researchers focus on varying the chain of reasoning displayed to a user. We are concerned with varying part of an already selected response to produce the ultimate response to the user.

4.2. Allowing for Updates to the User Model

Note that when we paint with such a broad brush, as for the case of the psychologist, it is not unreasonable to have the system simply mark those parts of its knowledge base used for the calculation of the response which are not shared by (known to) the user. This is because the majority of the knowledge is shared, so that the process is still computationally feasible. The suggestion for the general case (any user, any type of user) is to track the specific user more closely to focus on possible arenas for misunderstandings or lack of information.

For the psychologist, tracking the user becomes an issue in questions of the type below.

Dialog III:

Question: What is the diagnosis for the student?

Response: *Hypothesis*
Question: But why is *hypothesis* appropriate for this student?
Analysis: There is now a conflict between what the system has recorded as known by the user and what the user should thus be able to conclude, and the lack of belief in the conclusion, as implied by the question asked.

One description of the problem is:

UK (A) and UK (A → B) and yet ~ UK (B)

This description is in fact the model for Dialog I above – the *Hypothesis* is explained simply because the user may not explicitly believe the desired conclusion. (See [LEVE84] for further discussion of the notions of implicit and explicit belief). In Dialog III the problem is that, additionally, part of what the user is recorded as believing to use in drawing the conclusion, has been incorrectly assessed as a current belief of the user. We will not include here a discussion of the possible reasons for the incorrect assessment, and which reason is most probable. (For example: Is it most likely that psychologists disagree about diagnostic procedure vs. test interpretation?). Instead, we emphasize the need to track the user to draw a more accurate picture of possible differences with the system's "default model" of shared knowledge.

One possibility is to draw from previous dialog other misunderstandings that have been recorded in order to come up with a generalization about differences with this particular user. One example of this situation is a psychologist who still believes that certain tests now considered outdated are effective measures of student's problems. Once an indication of the user's preference for a certain test is noted, this information can be noted as differing form the system's knowledge about diagnostic procedure – i.e. SyKAD≠ SyB (UKAD). (See Example III).

Another effective strategy is to track the user's goals and preferences, and to postulate a plan for the user. This plan may then be compared to an ideal system plan, to note discrepancies and misconceptions. An outline of this work is provided in [VANB86a]. The general idea of performing some pragmatic processing on the input is not new; see [COHE85] for a review of some of the key efforts in natural language pragmatics. But the particular emphasis of van Beek's work [VANB86b] is to define a mechanism for recording user specific information, regardless of domain, and to suggest some methods for judging the best alternative available to a user, given his goals. The model is thus able to indicate a plan which also satisfies a user's preferences (see Example II). Thus, in conjunction with ongoing natural language research on plan-based approaches to inferring users' goals, it is possible to correct possible defaults that need to be retracted.

Note that the plan of the user is currently not included in the model of the user; what the system will be given, it is assumed, is some indication of the user's goals. The plan constructed by the system to help determine a response for the user, is not necessarily a plan that the user constructed, but a reasoning device for determining better alternatives for a particular identified goal (see [VANB86b]). It is possible to explore for future work the inclusion of procedures for additionally tracking the plan of the user, during discourse, to influence the response. But we have not advanced this in our proposal to date.

So far we have presented a basic strategy for answering a user's query. The general idea is that the response must contain the relevant aspects of the knowledge base which are not known to the user. In order to determine what knowledge is or is not shared by both the system and the user, the system employs initial stereotypes for the various user types, as well as updating this information through tracking the system-user dialog throughout the course of the diagnosis. In this manner, the system can determine what information the user is unaware of and hence, must be supplied as part of the response. However, this may not suffice. As a final step in the algorithm, the system must check that the user's goals and preferences have been addressed; if not, the answer must be modified to do so. Ultimately, one may wish to modify the interaction between the phase of the algorithm which detects which components of the knowledge base represent unshared knowledge and the portion of the algorithm which addresses the user's preferences and goals. The interplay between these two components also currently remains unaddressed.

The previously presented examples (Examples I-III and Dialogs I-III) illustrate various aspects of our general proposal for user specific responses. In the first example, the stereotypic models of parent and psychologist alert the system to what portions of the knowledge base must be supplied in the response (e.g. rule vs. background or testing facts). In Example III, we see the need to track the user-system dialog in order to update the initial stereotypic models, and in Example II, the need to address the user's preferences.

4.3. Dynamic User Modeling Through a Discourse

Before leaving this discussion of the system's general algorithm for answering queries (as illustrated primarily by the case of handling the psychologist's questions), we present examples to discuss the application of this procedure within a more general framework of the system's reasoning.

Dialog IV:

System: Administer an individual intelligence test such as the WISC-R.

Psychologist: Why is the WISC-R being recommended?

Analysis of Problem:

> Because the user is a psychologist, the system initially assumes that the user is familiar with the rules regarding diagnostic procedure etc., and hence these need not be supplied as part of the response. The information which is not shared by the user and system is some of the background data regarding the case at hand which has lead the system to believe that the student may qualify for high-cost funding. Therefore this information will be supplied in the response. However, before answering this query, the system needs also to determine whether the response adequately addresses any relevant user preferences. If the user prefers to administer a more familiar test, this response should help to counter the preference as well.

System: There is some weak evidence (such as a family history of learning problems, left-right confusion, slow language development) to indicate that the student may

qualify for "high-cost" funding; therefore, the WISC-R is being recommended because it is appropriate for use within the provincial funding formulae.

User: I would prefer to administer the McCarthy scales. Is this test appropriate?

System: No, only the WISC-R and the Stanford Binet are appropriate for use within the provincial funding formulae.

System Reasoning: The system now realizes that, contrary to its initial stereotype, the user is unfamiliar with the specifics of the provincial funding formulae and hence, marks within its knowledge base that this type of information is unknown to this user. This updated model can then be successfully employed later in the diagnosis. For example, the system now wants the user to administer an individual achievement test. Again the system would like the user to administer a test that can be employed within the funding formulae. Because the system knows that the user is unfamiliar with the these formulae, additional information will be supplied by the system.

System: Administer an individual achievement test. Either the WRAT or PIAT should be used because these can be employed within the provincial funding formulae.

For our second example, recall Example III from Section 3.1 in which the system and user have drawn different conclusions based upon the results of the ITPA.

Dialog V:

System: Administer an individual intelligence test such as the WISC-R or Stanford Binet.

User instead administers the ITPA (Illinois Test of Psycholinguistic Abilities) and the student's performance is recorded.

User: What are the conclusions based on the results of the ITPA? (i.e. What is the diagnosis?)

System: The student's intellectual ability appears to be within the normal range; no further assessment of intellectual ability is required.

User: What other conclusions were drawn? For example, the student's poor scores on the Auditory Association and the Auditory Closure subtests indicate that the student's auditory channel is weak.

System: Although the test results raise some suspicions regarding the student's processing of auditory information, such conclusions cannot be drawn from the ITPA alone because the validity of the ITPA's subtests is questionable.

System Reasoning: After this interaction, the system can conclude that the user does not adhere to certain current philosophies. In particular, the user *does* believe in the ITPA Model and consequently has drawn conclusions which differ from those of the system. The system's model of the user is updated to indicate both the difference in philosophy (adherence to the ITPA model) and candidate hypotheses (the student's auditory processing skills are weak). Then later in the diagnosis this information can be exploited when addressing the user's questions. For example, in the response below, the system supplies additional information regarding the student's auditory processing skills.

Scenario: The student has just completed an informal test of social maturity in which brief stories of moral dilemma are read to the student and he is asked to describe a solution.
User: What is the diagnosis?
System: The student's better than average score indicates the following:
- good social judgement
- good comprehension skills
- average ability to express himself verbally
- no difficulty with material presented via the auditory channel.

The student's performance on this test indicates that there is no reason to advocate further testing of the auditory channel. The student is *not* experiencing any difficulty with processing auditory information.

Recall the three examples of Section 3.1 which motivated the research discussed herein. The framework we have proposed is in fact appropriate for handling these examples. Dialog V demonstrates that our algorithm can be used to handle Example III, and Dialog IIb illustrates Example I. Recall that Example II concerns user preferences. We have addressed the handling of user preferences (as illustrated in Dialog IV), although we have left for future research the full integration of user preferences within our algorithm.

5. Related Work

There has been a wide range of previous work on the general topic of user models, but none of this work focuses on the particular needs of an expert system for educational diagnosis. One domain for which user models have been studied is intelligent tutoring systems (e.g. [SLEE82c]). The suggestion is to construct a model of the student interacting with the system, to better determine the learning problems of the student.

As well, there has been a study of producing effective responses from expert systems, but little accomplished to deal with a range of possible users. Mycin [SHOR76] at first had a very limited explanation capability. [WALL82] tried to accommodate different users by having the user supply his level of expertise. A limited amount of control on the level of detail in the prepared response then resulted. [SWAR83] examined how a more elaborate storage of knowledge (descriptive facts and principles) could produce better responses from medical expert systems.

There has also been work on constructing stereotypes for classes of users, to be employed in fine tuning the actual model for a user (e.g. [RICH79a]). In [RICH*], stereotypes are described as general predictions, not necessarily accurate, drawn for the user until specific knowledge is available. This is essentially our interpretation of the use of stereotypes as well.

Finally, there are the kind of user models constructed for natural language understanding systems which perform pragmatic processing. The work of [PERR78] considers the plan underlying the production of a discourse and the concept of mutual belief to help a listener to comprehend the likely aims of a speaker. [JOSH84a]

emphasize the need for a system to predict the false inferences that a user may be drawing, by carefully studying the goals underlying utterances.

In a sense, the user models we are advocating for our educational expert system draw from each of the above approaches. We essentially allow for the more robust kind of user modeling practiced in natural language pragmatics (tracking the beliefs of system and user, and including goals in the model), but employ default assumptions on users as a starting point and also accommodate correcting misconceptions.

There are a growing number of researchers studying the topic of "user modeling" (e.g. the recent Invitational International workshop on user modeling, Maria Laach, Germany). Our work is quite distinct in its scope and approach, but relates to some of these other current efforts. For instance, the work of [MCCO86] addresses the different perspectives on objects in a knowledge base which a user may have. Our framework suggests different perspectives for whole classes of users (psychologists vs. teachers, for example), not with respect to attributes of an object, but instead with respect to whole segments of the knowledge base which are more likely to be understood to a user.

The research of [FINI86] is also relevant, since this work includes the technique of assigning default values to the knowledge of users, and updating these models as discrepancies arise. We make the case for default models stronger for our particular domain where we have classes of users that can be identified, rather than the more general problem faced by Finin of determining the right classification for a user. As well, Finin offers more specific strategies on correcting the possible assumptions on a user.

The discussion of the distinction between agent models and user models in user modeling [WAHL*] is relevant to our work as well. Our system includes both a student model and a user model, where the student is in effect the subject of the queries to the system by an interested third party.

There are also common concerns with the work of [CHIN86, PARI85a]. Chin suggests a system of double stereotyping, assigning a user a level of expertise and then also labeling all the knowledge of the system according to a "difficulty level" (e.g. complex, simple, etc.). This calls for a segmentation of knowledge according to complexity. We instead advocate a segmentation into knowledge bases which are not labeled with one level for all users, but are determined as known or unknown to each user type. We thus avoid the problem of globally determining the classification of each individual rule.

Paris addresses a different dimension of accommodating different classes of users – varying the form of the response, rather than the content. So, expert users get a "process" description, while naive ones have "parts-oriented" explanations. We instead examine required content changes, and leave aside some of the lower level generation issues. Note that for both Chin and Paris, the user types are on an axis of expert to novice. For our domain, we work with a "user class" identification which is both appropriate for our given task and helpful in isolating the required user specific content in responses.

Some of the differences in our approach to user modeling, outlined above, allows us to advance a general framework for user modeling, summarized in the section below.

6. A General Proposal for User Modeling in Expert Systems

This chapter makes some recommendations for the next version of our expert system, but the principles advocated are intended to be useful as general guidelines for allowing expert systems to vary their responses to users. We summarize the general procedure recommended:

1. There should be defaults for the user types, to initiate a user model.
2. The system should make use of some partitioning of its knowledge bases, to focus on possible differences between user and system knowledge.
3. Differences should be accounted for in responses; so differences may be more usefully recognized by tracking the user's goals and ideally the previous dialog.

In connection with the third point above, it is possible to specify more clearly the potential avenues for disagreement between the system and a particular user class. For example, for the user as psychologist, we have already indicated in the Example III of Section 3.1 that the user may well disagree on the applicability of certain tests in the test database. We are aware that deeper specification of these dimensions of difference is an area for future research.

7. Current Status and Future Work

In our discussion of particular users of the expert system we focused on the psychologist and the parent. A third important class is the teacher, who knows the general purpose of diagnosis but does not know most of the details of specific tests. A teacher is not expected to be aware of the test results for the student, but may know a good deal of other background on the student, especially characteristics exhibited in the classroom environment (e.g. behavioral problems). For future work, we plan to develop a classification of this type of user in more detail.

In fact, we are currently refining our proposed specification of the knowledge bases within the part of the system referred to as "student knowledge". We have identified different sources of knowledge about the student, which directly associate to our user types (home, school and testing procedure), to expand on our current division of the background information into "acquired through testing" and "other". We are also considering the inclusion of both historical background information on the student (e.g. results of previous testing) and current. In this broader framework for the operation of the expert system, the additional divisions discussed above should facilitate user specific responses as well. We are encouraged by the application of our general strategy of subdividing the knowledge bases to identify unknown knowledge to classes of users, as demonstrated in our simulations of the system in the examples.

It is also worthwhile to comment on some current implementations into which we may integrate our proposal for variation of responses to users. The CGD project now has a useful small prototype system for educational diagnosis of students' errors

in arithmetic [TUBM86]. The basis for reasoning in this system is THEORIST, an approach of formulating theories about the possible diagnosis, using rules of default reasoning. So, for example:

default (problem (addition-of-wholes), Student, EWRPT-A test)
if:
is-in-grade (Student,Grade)
ge(Grade,2); le(Grade,7)
has-taken-test (Student,EWRPT-A test)
error-in-ewrpt-a-questions-1-or-2

This integrates well with providing a response to a user as follows:

User: Why is the hypothesis trouble with addition of wholes?
Response: (to teacher) Because he had an error in questions 1 or 2 of the EWRPT-A test.

In addition, work for employing goal information in the formulation of responses has been implemented [VANB86b], and a group is currently studying the integration of goals with the user's knowledge of the domain and student, for the selection of the response.

8. Conclusions

Expert systems for educational diagnosis require user models, to provide variability in the form and content of explanations generated for the users. This chapter describes a framework for incorporating user models into expert systems, which is specifically designed to

(1) identify and separate the knowledge bases which must be consulted in the preparation of an explanation
(2) allow for default assumptions of user's background knowledge in the absence of further information on the user, and
(3) continuously maintain and refine the user models, both as dialog from the user reveals knowledge and as previous output from the system adds to the shared knowledge between system and user.

In short, there is a prescription for the design of expert systems to incorporate models in order to serve a broad base of users. (There is as well a discussion of designing schemas for user classes, to guide the generation of responses). Moreover, there is a detailed argument for why this framework is particularly effective in the area of diagnosing learning disabilities. In sum, we have the means for implementing a more powerful version of our current system and a general framework of use to designers of other systems, to focus on beliefs of users in producing quality responses.

Acknowledgements. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

9. References

[ADEL82], [BUCH84], [CHIN86], [COHE85], [COLB82], [COLB83], [COLB84], [COMM71], [FINI86], [GRIC75], [JONE86], [JONE87], [JOSH84a], [LEVE84], [MCCO86], [MCKE85b], [MCLE82], [MCLE85], [MCLE86], [PARI85a], [PERR78], [POOL86], [RICH79a], [RICH*], [SHOR76], [SLEE82c], [SWAR83], [TUBM86], [VANB86a], [VANB86b], [WALL82], [WAHL*]

Part IV

Shortcomings of Current Models, Prospects for the Future

Introduction

The aim of Part IV is to point out a number of shortcomings that apply to the user modeling components developed to date, and to outline the research directions which should be given greatest priority in order to obtain a better understanding of the function of a user model in human-computer interaction, and eventually to build better models.

In this vein, **Karen Sparck Jones** identifies three major aspects which bear on user modeling: The characteristics of the person(s) involved in the interaction with the dialog system, the function of the model(s) of this/these person(s) in the system's dialog planning, and the means by which the information for the models can be obtained. The first two aspects can themselves be classified along several dimensions. With respect to the persons involved, Sparck Jones distinguishes between a person who interacts directly with the system (the so-called *agent**), and a *patient* who is the subject of the system's decision-making process.[1] In many cases, both roles will coincide with the same person. However, if a separate agent* exists who mediates information about the patient (e.g. a social insurance clerk consulting an expert system on behalf of an applicant), a model of both persons must be constructed, Sparck Jones argues, since assumptions about the agent*'s (Sparck Jones) properties (e.g. his/her level of experience, his/her prejudices, etc.) may also influence the system's dialog behavior (see Cohen & Jones' [COHE*] educational advisory system).

Other distinctions drawn by Sparck Jones with respect to properties of both agents* and patients are those between objective vs. subjective (mental) properties, dynamic vs. static properties (which do and do not change over time, respectively) and between decision vs. non-decision properties (which do and do not directly influence the system's primary task decisions, respectively). Sparck Jones claims that systems should, both for the agent* and the patient, in fact possess different models – one for each relevant type of property, i.e. up to 8 altogether – since they differ in the degree of scrutiny necessary for accepting them, in the treatment of inconsistencies, etc.

The second important aspect that bears on user modeling is the function of agent* and patient models in the dialog planning of the system. Agent models (necessarily non-decision ones) are relevant to the acquisition of information about the

---

[1] Sparck Jones' notion of 'agent' is more specific than that used by the other contributors (see e.g. [KASS*, MORI*, WAHL*]), hence the discriminating '*'. Morik's [MORI*] notion of 'protagonist' is closely related to Sparck Jones' agent*.

patient as well as to the agent's own interaction with and reaction to the system. Patient models can support the effectiveness, efficiency and acceptability of systems, i.e. contribute to the correctness of system decisions, reduce the time necessary for finding them, and improve their presentation to agent* and patient. By definition, non-decision properties of patients do not directly influence system decisions about him/her. When modeled, however, they may sometimes influence its effectiveness or acceptability. In some cases, they may even indirectly contribute to a judgement of the plausibility of assumed decision properties, or help to judge the credibility and expertise of the patient and thus the correctness of the information supplied by him/her, etc. The question arises now as to the extent to which non-decision properties (not only of the patient, but also of the agent*) should be sought and modeled by the system. Sparck Jones argues that these properties are too loosely connected and the implications yielded mostly too weak to justify taking great pains to determine and model them. Beyond decision property modeling for effectiveness, user modeling should therefore be very conservative, i.e. should not be driven too deep for content or stretched too far in use.

Katharina Morik's central thesis is that a number of research issues have not yet been sufficiently explored in the field of user modeling, particularly concerning the acquisition and revision of user models and – most of all – the conversational settings in which dialog systems equipped with a user model have been employed.

After some preparatory clarifications of what constitutes user models, agent models and dialog memories, the author identifies three factors that characterize conversational settings: the kinds of things that can be affected by system actions, the types of actions that can be employed for this purpose, and whether or not the user him-/herself plays a role in the domain of discourse. System actions may affect the user's domain model, the system's domain model, or the system's user model. Action types which can achieve these effects include communicative actions, observation of the domain of discourse, and "physical" (i.e. non-mental, non-communicative) actions in the domain. Action types may have more than one effect, and intended effects can often be achieved by more than one action.

Morik illustrates these factors by analyzing their influence in three types of conversational settings (namely information-seeking dialogs, reporting, and action-achieving dialogs), and by describing how the HAM-ANS system behaved in these settings. She claims that research on user modeling has so far concentrated mostly on information-seeking, and neglected action-achieving, dialogs (which, in addition to communication actions, also include "physical" actions). In this latter setting, a much wider variety of speech acts would occur both on the part of the user and of the system, and a much greater range of possible reactions to the user's input would be at the disposal of the system. Secondly, moreover, Morik claims that current systems rely too much on user model acquisition which is separate from, and prior to, the "normal" dialog in which, for example, the achievement of the user's goals is attempted. Instead, these processes should be much more intertwined. And thirdly, the revision processes for the user models investigated and implemented to date are claimed to be by far too simplistic, since they consist mainly of additions, but exclude replacements, modifications, dependency-directed backtracking, and the

recording of changes in user properties. To substantiate her theses, Morik reviews several existing user models and analyzes the extent to which these issues have been taken into account.

In a final section, Morik surveys the kind of user modeling that is performed by the HAM-ANS system in its role as a hotel manager. From facts entered about the prospective hotel guest, HAM-ANS makes assumptions about criteria along which the customer is presumed to evaluate rooms being offered and takes these assumptions into account in its responses to the user. A reasoning method based on the text-linguistic concept of *isotopy* (semantic cohesion) is presented as one step in this inference process.

Though user modeling and student modeling are related in many respects, they have generally been pursued separately for the past decade. The aim of **Robert Kass'** contribution is to explore the role of student modeling in intelligent tutoring systems as seen from the point of view of user modeling, and to find out what lessons can be learned for the field of user modeling. This is all the more justified since the area of student modeling has a somewhat longer tradition than user modeling.

Kass first differentiates intelligent tutoring systems (ITS's) from other types of computer-aided instruction systems and outlines their basic architecture. He then presents the three forms of relationships between the student model and the system's model about the domain of discourse (or "expert model") that have been investigated to date, and discusses their implications for the representation and acquisition of student models. Seven intelligent tutoring systems are surveyed in detail, and their contributions and limitations are discussed.

Against this background, the author discusses a number of research issues in student modeling that might also be relevant for user modeling, particularly for handling user misconceptions. His proposals include:

- The ITS methods for explaining why students performed a task in a particular manner (which usually require a deep, "psychological" student model) may be exploited for detecting and recognizing user misconceptions, e.g. in advisory systems (see [MCCO*, QUIL*]).
- The three above-mentioned forms of relationships between student and expert models are not limited to ITS's only, but are also appropriate for user modeling in applications where a body of domain knowledge is the main focus of interaction.
- Although systems containing a user modeling component generally cover a much wider range of interaction with the user than do intelligent tutoring systems (see [MORI*]), the admissible range is sometimes strongly restricted (e.g. when user misconceptions have been recognized). Methods for driving the interaction by means of questions or problems to which the student must respond have been developed in ITS research, the application of which may be considered in such cases.
- Attempts are being made in ITS research to isolate a small number of very general principles which can account for students' errors in a domain (the goal being domain-independent principles). Kass expects that similar rules could be used to

explain the ways in which users' beliefs, goals or plans are likely to differ from those in the domain model.
- Unlike user modeling, student modeling has already addressed the problems imposed by occasional non-regular user errors (the "noise problem"), although no general method has as yet been achieved there either.

Tim Finin is concerned with the development of a general tool system for user modeling. As this volume demonstrates, the development of a full-scale user modeling component is rather expensive when one has to build it right from scratch. Two research strategies have therefore been pursued to alleviate this problem:

1. to restrict user models in their scopes and aims (for instance, only some of the beliefs and goals listed in [KOBS*] are modeled, or the user model is exploited for specific purposes only);
2. to develop application-independent tool systems (so-called *user modeling shells*) which are intended to form the backbone of user modeling components in application systems. These shell systems would, for example, comprise a representation system for storing assumptions about the user, an inference system for deducing the consequences of current assumptions, a reason maintenance system for detecting inconsistencies, etc.

Tim Finin's chapter is concerned with the second approach (which is rather novel – the only other system with similar aims seems to be BGP-MS, briefly outlined in KOBS88b). After a short survey of the possible kinds, contents, acquisition and maintenance methods and application domains of user models, Finin classifies his General User Modeling System (GUMS) along these dimensions: GUMS allows for the representation of any kind of assumption about the knowledge (beliefs) of stereotypical and individual users, as long as they can be expressed in the PROLOG-like language on which the system is based (certain predicates have a system-defined interpretation). The acquisition of assumptions about the user from the ongoing dialog is left to the application system. The emphasis of the GUMS system lies on elaborate methods of default reasoning over the assumptions which are directly or indirectly supplied by the application system. No restriction whatsoever is made with respect to the uses to which the assumptions contained in the user model can be put; this is also left to the application system.

GUMS supports three kinds of default reasoning, which are described in detail in Finin's chapter: stereotype handling, explicit default rules, and negation as failure. These types of reasoning differ with respect to increasing granularity and decreasing strength of conclusion: stereotypes are used to capture generalizations of large classes of users. Within a stereotype, explicit default rules are used to express the stereotypic norms which are allowed to vary across individuals. Negation as failure is used as a general technique for gathering weak evidence for beliefs about a user when no stronger evidence exists.

Finin's approach seems to be a promising start in a direction which has been strongly pursued in other fields of artificial intelligence research in the last few years, namely the development of flexible and portable tool systems which the designer of

application systems can fill with application-dependent knowledge. Expert system shells, grammar workbenches, constraint propagation systems and other tools have proven to be very valuable for the development of better application systems. The same will probably also be true for general user modeling shells.

12 Realism About User Modeling

Karen Sparck Jones

1. Introduction

The object of this chapter is to explore the issues raised by user modeling from an analytic point of view. It is generally assumed that systems cooperating with human beings, and especially systems with natural language interfaces, need, and can get, user models. There is also a widespread assumption that the more user model the better and, correspondingly that, while we may not yet see quite how to do it, there is no problem in principle about being able to construct ever richer models.

The user models of current working systems, with some specialised exceptions like teaching systems, are very simple. This chapter does not describe any implementation. It considers examples that are much more complex than those that could be supported by the present state of the art, in order to lay out the elements and implications of modeling and so provide a framework for future work on actual implementations. This framework distinguishes and characterizes the major factors in modeling, namely the nature of the information in a model, the function of a model, and the means by which the information for a model can be obtained. In any specific modeling enterprise it is necessary to be clear about the relevant values of these application parameters.

The direct conclusion to be drawn from the chapter's analysis is a pessimistic one: even assuming very powerful system resources, far beyond the scope of today's systems, there are generally early limits to the modeling that can be achieved. Thus though the examples illustrating the argument imply modeling aims far more ambitious than those on which current system building is based, the conclusion is relevant to the long term research goals of the field.

But there is also a more positive and constructive message in the chapter. This is that the useful, if limited, modeling it is reasonable to seek would be promoted by adopting the framework the chapter presents. The distinctions it draws between the main components of modeling are important and should be applied.

This chapter attempts a comprehensive treatment of modeling considering first, what is being modeled, second, what it is being modeled for, and third, from what the modeling information may be obtained. The implications to be drawn from this discussion, and the relations between the account of user modeling given here and those presented elsewhere, are examined in the fourth, final, section on rational

principles for user modeling. The points made are illustrated through extensive examples. These examples, which are designed to show what is involved in modeling when it is taken seriously, are necessarily drawn from hypothetical systems. The chapter further takes specific modeling mechanisms for granted and is not concerned with what the technical procedures involved might be. The focus is on such questions as the aims and value of modeling, taking it for granted that it can, in a strictly technical sense, be done: thus I am assuming, for example, that some piece of information can be extracted from some linguistic expression, that a suitable data structure representing this and the other information constituting a model can be created and manipulated, and that the results of this processing can in turn drive further system operations.

For the purposes of the chapter I am also making some further assumptions. First, somewhat obviously, that I am modeling a human user. Second, I assume that the object is to model users as individuals, not simply to assign them to previous, explicitly defined classes. I am further assuming modeling in the context of complex computational systems of the kind represented by substantial expert systems. That is to say, it is taken for granted that the system may, for its primary purpose, exploit large bodies of knowledge and engage in extensive reasoning, with no direct connection with user modeling. I am also assuming, naturally, that there is some means of interaction between user and system: the implications of natural language interaction will be examined in Section 4 of the chapter. In addition, I assume throughout that we are concerned with the modeling that is done in a single interactive session, taking longer-term modeling as dependent on, and not essentially different from, this.

Finally I am assuming, without attempting to define rigorously, some notion of what a model is, as constituting something more than a simple aggregate of pieces of data. Individual items of information, or a collection of these, do not constitute a model. The presumption in talking of user models is that items of information about a user may be related to one another, or to other (typically general) knowledge stored in the system, in a manner which supports predictions that can stimulate further system actions. This is not necessarily a very strong definition, since much depends on the nature and extent of the relationships implicitly or explicitly holding between pieces of information. For example in a medical system there may be a tacit assumption of causal relations between symptoms. The kind of categorization applied in GRUNDY [RICH79b] to predict book choices, on the other hand, does not make any strong assumptions about causal relations between the different user characteristics.

2. What Is Being Modeled

This section considers aspects of users bearing on modeling, or modeling *factors*, drawing a number of distinctions relative to these factors. Some of the distinctions may seem over-fine. The justification for them is to be found in Section 3, where the uses to which models may be put are examined.

2.1. User Roles

It is necessary, first, to distinguish the different roles users can have. In a given session, we may have different users, with distinct roles, or just one user with different roles. The two roles I distinguish are the *patient* role and the *agent* role. I define the patient as the subject of the system's decision-making processes, confining the system's decision-making processes here to those the system was primarily constructed to carry out. I assume the system was designed to reach decisions in a specific domain, in the sense in which classical expert systems have been designed to reach decisions about the diseases people have, the financial investments that might be appropriate for individuals in given circumstances, and so forth. The domain knowledge is instantiated in relation to the user, to reach a domain conclusion about him. (This therefore views personal financial decisions as ones about a human money owner and not as ones relative to abstract financial situations.) This is not of course to imply that a system reaches a decision only at the end of a session: it may take decisions continuously throughout a session, as in a teaching system.

As will be evident, a system can make decisions, in the general sense, about other matters. But I am specifically restricting "decision-making" here to activities aimed at satisfying the system's primary task goal. In the medical case "patient" has an obvious interpretation, but I am using it as a role label in a more general sense.

The agent, by contrast, is the person who conducts the interaction with the system. The agent, as such, is thus not the subject of the system's decision-making processes.

The implications of this distinction between patient and agent will become apparent as the other components of modeling are considered. Here it is necessary to point out that different person and role situations can arise as shown in Figure 1. Thus for a given system we may have a single human whose only role is as patient, as with a surveillance system where the interaction between the system and the external world, and hence with the patient, is through automated monitoring. Or we may have a situation where the single human being is only the agent, as in industrial design, where the system's decisions are about inanimate machine structures. We may have two human beings, one as patient and one as agent, so the latter is acting as the medium for information about the patient, as in the medical case with a nurse as system driver. Finally, we may have a system with a single human who is both patient and agent, as in a home computer diet advice system.

It is clear that both patients and agents, as humans, can have beliefs, goals, plans, etc. All users, whether patients or agents, are thus agents in the general sense of purposive beings. However I shall specifically reserve "agent" in this chapter for the role definition I have just given, taking it for granted that any user can have beliefs, goals etc; (so if it is necessary to refer to the general meaning of "agent", this will be clearly marked). My use of "agent" is therefore more particular than that generally adopted in discussions of user modeling. My distinction between agent and patient is indeed intended to allow an analysis of aspects of modeling not covered by the normal unitary view of the user (though Morik [MORI*] makes a similar distinction between 'user' and 'protagonist').

I shall use "user" where the distinction between patient and agent is immaterial because the points made apply to either role. This makes "user model" comparable

1. PATIENT ONLY
   e.g. automatic surveillance of a human being
2. AGENT ONLY
   e.g. human operater of an industrial design system
3. PATIENT AND AGENT, DIFFERENT
   e.g. medical diagnosis system with nurse agent and sick person patient
4. PATIENT AND AGENT, SAME
   e.g. diet advice system with customer agent and patient

Figure 1. User role possibilities

to "agent model" as the latter is generally understood, but here I am using "agent model" more restrictively to refer specifically to a model of a human in the agent as opposed to patient role.

2.2. User Properties

It is further necessary to distinguish types of user property. Some of the distinctions to be made here are not theoretically watertight, but they are operationally important because they are reflected in material differences in the reliability and accessibility of modeling information.

The first distinction, see Figure 2a, is between *objective* and *subjective* (or mental) properties, for example between clinical thermometer readings and sensations of worry. (Note that subjective properties are defined as subjective in the possessor, not the observer.) This is an intuitive, but real enough, distinction. In general we behave, and therefore a system could similarly be expected to behave, as if it is easier to establish, for example, that a user is female than that they believe democracy is under threat. (There is no space here for the more detailed discussion this distinction requires: for example what is the status of reported properties, as in "I felt terrible pain"? A good case can nevertheless be made for the basic distinction as one with consequences that have to be taken into account in modeling. For some further points see [SPAR84].)

A second distinction, cutting across the first, and applying most obviously but not exclusively to subjective properties, is that between *static* and *dynamic* properties, for example between political beliefs that hold throughout the session and preferences for dresses that change in interaction with a mail order system. This is not to imply that political beliefs cannot change. It is rather that, in relation to a system's purposes and activities, there are properties which the system can treat as static because they are, if not necessarily static, at least relatively permanent and are unlikely to be changed by the system's activities, which are not directed towards decisions concerning them.

The distinction between static and dynamic does not refer to the effect of time on the *system's* knowledge of a user property, where a system may suppose X until

is a decision one or not is wholly contingent, as it depends on the specific system application. Any particular user property, e.g. political belief, may be a decision property in one system (e.g. a voting advisor) and a non-decision one for another (e.g. an arithmetic tutor).

If the system uses both non-decision properties and decision ones, the distinction between them may appear unnecessary. But it may nevertheless be important operationally. Property information about users can be exploited for different purposes, described in the next section. Clearly, that for which decision property information is sought is the most important. The permanent system model of the domain phenomena to which these properties refer may therefore be much more fully developed and characterized than that of other entities or activities; and it may also be the case that the importance of decision property information may mean that it has to be handled more carefully, for example be more rigorously tested. There are therefore good practical reasons for distinguishing these two types of property, as will be more clear from the examples considered later, though there may be no intrinsic difference in the nature of the properties labeled decision and non-decision for a given system.

Though for precision it might be desirable, in relation to to the three distinctions just drawn, to refer respectively to, say, the type, mode and status of a property, I shall regard it as sufficient to refer simply to the (six) types of property, as listed in Figure 2a.

It will be evident that in general these distinctions of type are relevant to both patient and agent roles. Thus as Figure 2b shows, the same property, like salary, or age, can hold of patient or agent, indeed hold of both where there are two people. However there are some restrictions. Thus by definition decision properties can only be properties of the patient: they do not apply to the agent role.

3. What Modeling Is For

The previous section characterized the nature of the inputs about users with which a system could be concerned. The next question is why it should be so concerned. Specifically, if the function of a system is to take decisions, why should the agent role be considered? Why, indeed, should non-decision properties even of the patient be considered?

Both roles, and both decision and non-decision properties, are relevant because there are quite different system functions user models can serve, so non-decision properties of either patient or agent can be helpful in supporting or enhancing system performance. The possible functions models can serve are illustrated in Figure 3, for a medical system with distinct patient and agent. Thus the first, and most important function models can serve is system *effectiveness*. The prime object of a user model is to ensure that the system, as a decision-making system, reaches the correct decision. For example, if we have a medical diagnosis system, we need a model of the patient which is adequate to support what, if independently evaluated, would be shown to be the correct decision (assuming that, given adequate evidence, the system's processing is itself sensible.) This clearly implies that the only relevant

model for effectiveness is a patient model. Effectiveness is clearly also a necessary system function.

However, effectiveness is not the only system function user models can serve. Models can serve system *efficiency*, i.e. reaching the correct decision in an economical way. A patient model is obviously relevant here too: for example, the available knowledge about a patient may be used to order the tests to be made on the patient for further information so that the most potentially useful tests in the specific circumstances are carried out first. Moreover, if the agent is seen as a transmitter of information about the patient, the system can also exploit an agent model to gather information in one way rather than another, as likely to be more efficient. For example, if the agent is medically experienced, this fact can be used to promote efficiency through quick answers because well-defined medical terms can be used in questions the system puts to the agent.

There is also a third function a user model can serve. This is system *acceptability*, i.e. supporting (or expressing) its decision making in a comprehensible, perhaps also agreeable, way. (This refers primarily to the system's operations in a session, not to the extent to which its performance over many sessions carries conviction about the validity of its decisions.) Here again both agent and patient models can be used, for instance the fact that the patient is in pain can be exploited to emphasize one aspect of the proposed regime (as likely to reduce pain) rather than another (which may be nasty), and the fact that the agent is experienced to augment the decision information with literature references to comparable case histories. Acceptability, like efficiency, is not a necessary function; but both may be very important in practice.

EFFECTIVENESS	
e.g. disease diagnosis	P
EFFICIENCY	
e.g. diagnostic test ordering	P
terminology of system questions	A
ACCEPTABILITY	
e.g. medical regime presentation	P
case citation	A

Figure 3. System functions and relevant models. A medical diagnosis system. P = patient model, A = agent model

As the examples of Figure 3 show, the different role models may be used in different ways to serve distinct system functions. But this general statement disguises some relevant complexities. Thus in relating models to functions it is necessary to separate the source of the modeling information from its functional destination. In the example of Figure 3, *source* and *destination* are in fact the same, e.g. information about the agent is applied to the agent, as in the exploitation of his experience to use technical terminology. But patient data could also, for example, be applied to

OBJECTIVE / SUBJECTIVE
e.g. temperature reading / feeling worried
STATIC / DYNAMIC (for session)
e.g. political belief / dress purchase preference
DECISION / NON-DECISION
e.g. temperature reading / age ; dress purchase preference / political belief

(a) Property type distinctions

OBJECTIVE	P / A	salary
SUBJECTIVE	P / A	anxious
STATIC	P / A	sex
DYNAMIC	P / A	belief about relative costs of brick and concrete
DECISION	P	foreign language words known
NON-DECISION	P / A	age

(b) Property role possibilities. P = patient, A = agent, assumed different people

Figure 2. Property types

the passage of time brings evidence that not-X, but the user is throughout in state not-X. It refers to changes in the *user* over time, i.e. from state X to state not-X. In practice it may be difficult for a system to know whether its knowledge or the user's state is changed, and in some cases the consequences may be the same. But I have introduced the distinction to allow for the situation where changes in the user are important because modeling is intended not merely to recognise them but (possibly) to effect them. Clearly, though, there need be no close correspondence between when a change in a user is observed by the system and when it is experienced by the user. It is important to emphasize that objective properties can change as well as subjective ones, though this may be less common: for example instructions to sell financial stocks.

There is, finally, a third property distinction which also cuts across the others. This distinction, unlike the others, is an absolute one because it is explicitly defined in the construction of the system for its primary task; and as it is fundamental to the system's purpose, it is also much more important than the other two. This distinction is one between *decision* and *non-decision* properties. As noted earlier, an expert or other complex system is intended to take decisions in a certain domain. Those user properties that pertain to this domain are a user's decision properties. According to the design purpose of the system these may be, for example, medical symptoms, or preferences for sorts of clothes. They can clearly be a mixture of objective and subjective, or of static and dynamic, as illustrated in Figure 2a, which shows a static objective property, temperature reading, and a dynamic subjective property, sartorial preference, as possible decision properties. Whether a property

system functions directed at the agent (i.e. in his own right, not as an information transmitter), for example to explain the diagnosis.

There are indeed further points to note, which also have implications for the detailed way in which models are exploited. In using models to serve functions, it is necessary to separate (logical) *addressee* from *beneficiary* (as well as actual from logical addressee: the agent may be actually addressed to transmit a question to the patient). Thus the agent may be addressed, using an agent model, to obtain information exploited for the benefit of the patient, or for the agent's own benefit; or the patient addressed to obtain information which may also, as just suggested, be exploited for the benefit of the agent. One may also have first and second order beneficiaries, so for example, benefitting the agent indirectly benefits the patient. In other words, we may have both patient and agent models serving system functions themselves directed towards both patient and agent. Either patient and agent model may naturally, moreover, be exploited differently both to gain inputs, and to direct outputs, in supporting the system's various functions.

It must further be emphasized, as has been implied and will be more fully illustrated below, that both decision and non-decision properties can be of value in relation to the system's functions. In the example of Figure 3 this is shown most particularly by the use of the agent model in connection with efficiency, since the agent's properties are necessarily non-decision ones; but if we assume, for example, that pain is not a decision property for the disease in question, a patient non-decision property is being used for acceptability (to the patient).

The set of distinctions drawn in this section seems very elaborate. But the distinctions and their implications are relevant if we want to serve system purposes by modeling, given that modeling situations range from those where it is necessary to recognize user properties to those where it is only desirable, and from those where models are used to promote positive system behaviour to those where we want to inhibit negative behaviour. This discussion of course assumes, in general, a system of sufficient complexity to suggest a need for modeling beyond that involved in constructing and manipulating the basic patient decision model. But we still have to relate the validity of the modeling that can in practice be done to its actual utility.

Example 1

The need, assuming modeling is feasible, to recognize all the distinctions discussed, relating functions and roles and also property types, can be illustrated by a fuller system example, summarized in Figure 4. Suppose we have a (notional) social security expert system with distinct patient and agent. The patient is an elderly and disabled woman who has to use a wheelchair; she also has poor eyesight, is a Catholic, honest but suspicious of officialdom, poorly informed about benefits, and believes that age is the main determiner of benefit (these are illustrative properties, and are not intended to constitute a complete patient description). The agent is an experienced clerk, but one who is a male chauvinist who believes women are unduly favored by the benefits system. Figure 4a lists the patient's properties by type. For clarity, but in some sense artificially, user properties of different types are treated here as pertaining to separate models. Thus for the patient we have

PATIENT

		Decision	Non-decision
P1	**Static, Objective**	aged, disabled, female	
P2	**Static, Objective**		poor eyesight, Catholic
P3	**Static, Subjective**		
P4	**Static, Subjective**		honest, suspicious
P5	**Dynamic, Objective**		
P6	**Dynamic, Objective**		
P7	**Dynamic, Subjective**		
P8	**Dynamic, Subjective**		poorly informed, age matters

(a) Patient properties

P1 **Static, Objective, Decision**
disabled → EFFECTIVENESS e.g. determinant of benefit
→ EFFICIENCY e.g. drives search establishing benefit
→ ACCEPTABILITY e.g. referred to in explanation of benefit

P8 **Dynamic, Subjective, Non-decision**
age matters → EFFICIENCY e.g. guides data gathering
→ ACCEPTABILITY e.g. referred to in explanation of benefit

(b) Exploitation of patient properties for system functions

AGENT

A2 **Static, Objective, Non-decision**
experienced → EFFICIENCY e.g. data gathering
→ ACCEPTABILITY e.g. practice manual references

male ...

A4 **Static, Subjective, Non-decision**
chauvinist → EFFICIENCY e.g. data checking for bias
→ ACCEPTABILITY e.g. explanation of benefit basis

(c) Agent properties and uses

Figure 4. User models and their uses. A social security benefits system (patient and agent distinct)

a P1 model referring to her static, objective, decision properties, in this case age etc., another model P2 referring to her static, objective, non-decision properties, poor vision etc., in fact eight models altogether, though not all are instantiated

in the example. Agent models are correspondingly numbered, but as an agent cannot have decision properties, the only relevant models are the even-numbered ones.

The separation of models for a single role is somewhat artificial, but does serve to mark differences in the relative importance or stability of the information involved which, even in a single union model, could be reflected in the way information in the various classes was managed and applied. Decision property information may be more carefully vetted, for example, and inconsistencies require different treatment in static and dynamic cases, since static property clashes may have to be resolved where dynamic inconsistencies may legitimately reflect changes over time.

The kind of uses to which the example modeling information might be put are illustrated in Figure 4b and c. The patient's disability is directly used to reach a decision about the amount of benefit due, but may also be exploited in the interests of efficiency to organize the search for possible benefits (I assume a suitable rule structure allowing this). It may also be used for acceptability to the patient to explain the makeup of the benefit payment, for example to indicate that disability precludes some other apparently relevant factors like sex. The patient's dynamic belief that age is the determiner of benefit could be used to support data gathering efficiently through the forms of questions, or might be corrected as part of an explanation for the nature of the questions being asked.

Turning to the agent's illustrative properties, his experience could be exploited for efficient data gathering and to make the decision reached acceptable by reference to the relevant sections of the regulations. His unfortunate chauvinism, on the other hand, might have to be counteracted, for efficiency, by careful question formulation in data seeking and, for acceptability, by an indication that sex is not the basis of the calculated benefit. (Notice the need to maintain a broad definition of acceptability, and the need for the system to discourage as well as encourage user behaviour.)

This example is developed in more detail in [SPAR86], and is compared there and in [SPAR84] with other examples illustrating systems with different purposes and other patient and agent role combinations and property descriptions. The various examples taken together are designed to provide a fuller picture of the relevance and utility of the factor (i.e. role and property type) and function distinctions I have made. The situation would be quite different, for example, in a system with the rather different factor combination represented by a learning system with patient as agent and decision properties that were all or mainly dynamic. The social security system would of course also be very different with different properties or property values.

These different examples, however, simply reinforce the important general points made here by the social security illustration. These are that when one looks at systems as wholes and not only at interactive interfaces, and at systems engaged in non-trivial tasks, it is necessary to allow on the one hand for distinct user roles and property types, and on the other for distinct system functions. These distinctions then imply multiple relationships between pieces of information about the user and system purposes. Thus an individual user property may be exploited not only to support different functions; it may be used in different ways to serve a single function. A particular function may, conversely, exploit not merely different

properties of the user, but different types of property. It is particularly important to recognize the need, in more complex systems and system environments, for the distinction between decision and non-decision properties and the value of nondecision information in supporting the efficiency and acceptability functions which, if they are not primary in the way effectiveness is, are nevertheless highly significant.

Thus if we consider a single user property, namely sex, this may be a decision property in some systems. But in other systems where it is not a decision property, it can still be very useful. For example (and making some arbitrary medical assumptions), if sex is not a decision property for a smallpox diagnosis system, it might still be exploited for efficiency to order diagnostic tests (supposedly justifiable by possible pregnancy implications), and/or it might be used in relation to acceptability (to the patient) to couch verdicts in different terms (assuming women are more sensitive to potential disfigurement than men); sex could indeed be used in more than one way in relation to acceptability, for instance to draw attention to the implication of verdicts for contact with children (generally rather different for men and women). Of course, in choosing information to present the system is making a decision in the ordinary sense, though not in the strict sense defined earlier: properties which are not decision properties for the central task become decision properties for the subsidiary response generation task. The distinction between central decision properties and non-decision properties is nevertheless a useful one.

		PATIENT	AGENT
EFFECTIVENESS			
Decision	(Static / Dynamic) (Objective / Subjective)	X	
Non-decision	(Static / Dynamic) (Objective / Subjective)	(X)	(X)
EFFICIENCY			
Decision	(Static / Dynamic) (Objective / Subjective)	X	
Non-decision	(Static / Dynamic) (Objective / Subjective)	X	X
ACCEPTABILITY			
Decision	(Static / Dynamic) (Objective / Subjective)	X	
Non-decision	(Static / Dynamic) (Objective / Subjective)	X	X

Figure 5. Factor-function relation possibilities

The rich pattern of relationships between kinds of modeling information and their functional uses that follows from going outside the decision core of a system is summarized in Figure 5. This applies the convention introduced earlier and isolates each factor combination as a submodel, so looking at the different functions models can serve gives us the distribution shown in Figure 5.

As Figure 5 indicates, the contribution of some models to some functions is indirect rather than direct. The agent or patient's non-decision properties can bear

on system effectiveness only indirectly through, for example, the transmission of incorrect decision information. It is of course the case that quite apart from the general question of whether a system can in a straightforward sense obtain reliable information (it may simply not be clear, for instance, how severe a sick person's pain is), transmitting patient information through an independent agent can lead to all kinds of distortions (the doctor may think the patient is malingering, for example).

As a further potential complexity, it should also be noted that it is possible that each submodel may really be a set of models, either because different and incompatible interpretations may be made of the available evidence, or because the different uses made of the same information impose different types of structural organization on it. This sort of complexity arises, moreover, even when the factor distinctions made earlier are suppressed so all the information about one human is amalgamated in a single model. (This of course assumes the system can recognize, if it is not set up to assume, when there are two people as opposed to one.)

These points are illustrated more fully in Example 2, presented in Section 4 below.

4. What Modeling Is From

I have not so far considered the means of communication between the user (strictly agent) and the system. If we can see the point of modeling, there is still the question of how easy it is to do. This clearly depends essentially on whether the user information needed for modeling can actually be obtained, and this in turn depends on the way user and system communicate.

The context in which this chapter is being written assumes that user and system (and especially the former) communicate in natural language, and ideally in full and free natural language. Natural language is of course not the only possible means of communication. Setting aside situations where user, and specifically patient, communication is like the autonomous and involuntary monitoring of the VM expert system [BUCH84], where the patient is not a decision-making agent in the ordinary sense of "agent", there may be applications where the user may communicate, as a purposive agent, with the system through nonlinguistic means. This could be the case, for instance, in a design system where drawing was the means of communication, or in a mathematics teaching system where arithmetic terms and expressions are used. Formal language, and quasi- and restricted natural language are also obvious possibilities, which may be implemented to allow more or less initiative and direction to the user: with the sort of restricted natural language interface often found with menus, for instance, the user's freedom is typically very limited.

It is important to recognize that these other communication means can perfectly well support user modeling (as further discussed in [SPAR84]): intelligent computer-aided instruction in school mathematics is an obvious example, where the instruction is specifically driven by a model derived from the user's problem-solving behaviour. Restricted communication, e.g. through menus, may also be perfectly

satisfactory for a particular application. I shall nevertheless concentrate on interaction through unrestricted natural language (assuming therefore a natural language processing component in the system which is far beyond out present capabilities), primarily because natural language offers the richest communicative resources and so the best opportunity to gather information indirectly if, for good reasons, it is not supplied directly. In language use information may be conveyed both directly, as in the answer to a question, and indirectly, as in the form of words chosen. As in human dialog, it may not be feasible to gather all the modeling information directly, either because the necessary explicit interchange becomes too tedious, and therefore oppressive, or because it may be too offensive to the user. Natural language concentrates many different items of information simultaneously in its expressions through the concurrent deployment of, for example, lexical choice, syntactic structure and form of reference, so natural language interaction offers the best chance of maximizing user information. This is independent of the fact that natural language dialog may be required or appropriate for other reasons, for example speech in telephone interaction.

It is also important to emphasize here that exploiting already-interpreted natural language communications to gain modeling information is logically distinct from having to have a model to interpret these language communications in the first place. Thus modeling may be logically parasitic on natural language communication adopted as an effective means of obtaining the system's decision-making information, as is most clearly shown where there is no human patient, but also holds for models other than decision-property ones where there is a human patient. Or it may be deemed to be a prerequisite for the utterance interpretation that supplies that decision-making (or other functionally-useful) information. There may in any case be situations, for example where the human patient is also the agent, where this distinction seems immaterial. The point of making it here is nevertheless to draw attention to the fact that in this chapter the relation between language and model is being approached from one end rather than the other, so the concern is with what the linguistic expression contributes to the user model, and not with what the model tells us about the expression. This is not unreasonable, given the varied modeling purposes considered here, since, though some model elements like subjective goals will clearly work both ways and play a part in both enterprises, others may have little part to play in assisting linguistic interpretation.

Example 2

The problem of extracting useful modeling information can be illustrated by an extended example. This is a hypothetical dialog, representing the first part of an interaction, for the social security benefits system of Example 1. The users, the elderly disabled woman as patient and the experienced benefits clerk as agent, are assumed to have the properties given earlier in Figure 4, with some additional ones needed to make modeling fruitful and interesting (these will emerge in the dialog).

The example assumes that the system has the necessary technical linguistic interpretation and generation capabilities, and also the necessary internal modeling component. Thus I am assuming that the system is in principle able to recognize instances of the kinds of properties it has available to describe patient and agent,

and may indeed be capable of constructing new, particular properties out of other knowledge it already has. The example is designed to show what support natural language dialog, and more specifically the user inputs, can offer for modeling, and hence to provide a basis for a realistic assessment of the opportunities for, and value of, user modeling in enhancing system performance.

In the dialog, the modeling conclusions drawn by the system from its inputs (by whatever means it has for doing this) are associated with the various submodels described earlier and illustrated in Figure 4, i.e. P1 the static, objective decision model for the patient, P2 and A2 the analogous non-decision patient and agent models, and so forth. Thus the presumption is that the system starts with empty models, and seeks to construct reliable and coherent submodels, with its dialog responses seeking further modeling information as indicated. The use of the submodels is primarily, as before, to make the different modeling operations clearer. The symbol "+" below indicates the addition of information to a model, or its reinforcement if already present; "?" indicates the model proposition is hypothetical rather than certain (of course in practice there would not be an uncertain/certain dichotomy, but different degrees of certainty: however, the simple dichotomy is convenient here). The comments, *c*, on input and output, summarize what the system is doing.

As this is the beginning of a dialog, the system's responses are primarily directed towards obtaining more modeling information rather than applying a built model e.g. in explaining something; but as the example shows, the information seeking is frequently guided by the (tentative) models already constructed, though in some cases the system engages in quite basic information gathering. Moreover it is possible to show, even this early in the dialog, the system applying the information it has to different system functions. (Note that I am not concerned here with the plausibility of the assumed benefits regulations and its expert system implementation.) "O" is system output, "I" is user input. (I assume the initial logon sequence has been gone through.)

O: Please give me details of the applicant.

I: The applicant is female, aged 89, widowed, grade 3 disabled, and I imagine will get a pretty lavish benefit.

c: P1 + female, 89, widow, disabled 3
P2 + nil
P4 + nil
P8 + nil
A2 + experience?
(use of "grade 3" jargon)
A4 + hostile to patient?
(use of word "lavish", "and", "will" – compare "but", "should")
A8 + nil

O: What are the applicant's sources of income?

c: response motivated simply by need for further decision property information, primarily to promote effectiveness (i.e. no model application yet).

I: She says she's getting widow's pension of 17 pounds and 3 pence and otherwise the priest sometimes gives her 2 pounds out of his poor box. Her clothes seem rather worn.

c: P1 + pension 17.03, charity occasional 2.00
P2 + catholic?
(reference to "priest")
P4 + honest?
(pence details; mention of odd gift)
P4 + ignorant?
(does not realise housing status relevant)
P4 alternative + dishonest?
(concealing housing status)
P4 + suspicious (of clerk)?
(only replying to direct questions; note not necessarily an alternative)
P8 + no obvious beliefs, expectations about system?
A2 + nil
A4 + hostile to patient?
(reinforced by use of "she says", "seems")
A4 alternative + cautious observer?
(not experienced in evaluating state of old women's clothes)
A8 + exploring influence of modifications on basic income?
(apparent interest in whether she gets a clothing allowance)

O: Find out whether she buys clothes frequently. But has she been receiving anything for her disability? And ask her whether she lives with anybody.

c: response motivated by need to elaborate on P1, for effectiveness and possible efficiency; also to check on P4s of honesty and ignorance, attempting to select from alternative hypotheses, for the same reasons; by use of homely language attempting to bypass hypothesised suspicion, with additional motive of enhancing acceptability to patient; further motivation to gather information sufficient to start a useful P8. With respect to agent response motivated by attempt to distinguish alternative A4s and sidestep hostility, primarily for effectiveness but also for acceptability to patient; also using A8 suggesting focus on ancillary income to redirect attention to main sources of income. (Here the system is beginning to apply its models.)

I: She says the hospital gave her a wheelchair and that she lives by herself but her daughter sometimes visits and the District Nurse attends at basic frequency. She finds it very difficult to afford clothes.

c: P1 + no disability allowance, home owner?
P2 + nil
P4 + honest? i.e. + +
P4 + ignorant?
(apparently does not know of disability allowance, clothing allowance)
P8 + no notion of system and its capabilities?

A2 + experience?
("basic frequency" jargon)
A4 + nil
A8 + nil

O: Ask her whether she owns her home, and if she is still paying anyone for it (check for mortgage freedom). Tell her we may be able to help even if she has a house.

c: response pushing for more P1 information, for effectiveness and, we will suppose, efficiency and acceptability; also reacting to and testing P4s and P8, promoting acceptability and perhaps changing the patient attitudes embodied in the P8. The system is exploiting the agent's experience in A2 (reference to "mortgage freedom") and trying to reduce his hostility and render the system more acceptable by exhibiting care and exhaustivity in its operations.

This example illustrates many specific points about modeling, for example the possibility of agent interference in the transmission of information about the patient, and also the need to distinguish user beliefs about the nature of the domain as covered by the system from beliefs about the domain within which the system operates. For instance the patient may have beliefs about the scope of the system based on her beliefs about the scope of social security, although the scope of the system is narrower: for example in relation to housing, she may suppose the computer system covers rates rebates because social security as a whole does, though the system is in fact only concerned with housing in relation to asset ownership. The patient's mistaken beliefs could motivate replies to questions about what she is paying which were quite misconceived, but in a way very difficult to detect.

But the major, critical points made by the example are first, how heterogeneous the evidence for models is, and second, how weak it can be. This is particularly in relation to properties other than decision ones, i.e. the non-decision properties of both patient and agent. Thus there are three data of different sorts, the three pence, the cash gifts, and the daughter visits, all suggesting the user is honest; however any one of these, like mentioning the three pence, is really only very weak evidence for honesty: it could rather indicate a passion for accuracy, for instance.

The essential problem about modeling is that property indicators may be unreliable or undiscriminating. Thus to take an independent example, a man being a Catholic does not imply he is a priest, so being a Catholic is unreliable evidence for being a priest. But being a priest does not imply being a Catholic priest: as there are priests in other churches, being a priest is not discriminating. Building and applying models is clearly a very tricky business, even for decision models, but it is much more so for non-decision models. It should be noted, moreover, that these points also apply to modeling initiated by stereotypes: for example the stereotypic old woman may be religious, but it may not be at all easy to verify that a given old woman is religious (in the extreme case, even by asking) or, more importantly, to establish that she is in fact not religious.

5. Rational Principles for Modeling

Given the manifest difficulty of obtaining reliable modeling information, what are the implications for system design? Should the system devote effort to trying to confirm the evidence? Information about the user is needed to guide action bearing on the user, but if the system does not have much, or good information, should it seek more, especially if drawing strong conclusions from weak evidence can lead to poor performance?

There is no doubt that if the decision properties the system uses are the only proper ones for the nature of the decision to be made, then every effort has to be made to get the best information about them, even if this is very difficult. The problem is much more serious with non-decision properties because it may be much harder to get useful non-decision modeling information.

This is primarily because there are potentially very many, diverse non-decision properties, not connected with one another or with decision properties. The presumption about decision properties, as mentioned earlier, is that they form a related set, which is typically also a select set, especially if the system's task is a well-defined and constrained one. But the number and range of properties a user can possess outside the decision set is large and wide, even considering only properties that might be validly exploited to serve any of the system's functions. Thus for the example system, user properties as disconnected as religion and poor sight could be usefully exploited in the interests of system acceptability, the one to stimulate a visit to the priest to encourage a discussion of a low benefit, the other to direct the patient to specially-printed large-type explanatory leaflets. The problem is that there may well be no motivation or leverage for a system search in one non-decision direction rather than another (unless a random pursuit is adopted on the basis that anything is better than nothing, and one thing is as good as another).

There is a particular further problem with non-decision property information, assuming it is not explicitly sought, namely that it is more likely to be conveyed indirectly than directly by the user, and thus be more likely to be subject to uncertainty. In general, the more indirect clues are the less helpful they are. Again, the deeper models are, as those concerned with subjective properties generally are, the harder the properties involved are to ascertain.

All this suggests that the realistic conclusion to draw is that rather than diverting systems into costly and probably still unprofitable searches for evidence, given that even quite powerful systems may yet lack extensive general world knowledge, we should restrict modeling to the user properties we have a chance of getting good information about, i.e. primarily, if not exclusively, decision properties, for which data, if they are not sought directly, may be more obviously supplied or, where decision properties are systematically related, inferred. The further conclusion is that modeling, especially modeling beyond decision property modeling for effectiveness, should be very conservative, i.e. it should not be driven too deep for content or stretched too far in use.

It is relatively easy to imagine how this strategy could be applied to the social security example. The system would not bother, for instance, with the patient's religion as this is a high cost, low payoff property, i.e. one difficult to establish and

of rather marginal utility through not having a predictable acceptability value; the system could probably do better tailoring output for acceptability by using suitable decision properties like age. Honesty and system expectations would, on the other hand, be available as non-decision patient properties, but would not be pursued unless the decision data on hand was blatantly incongruous or inadequate.

The illustration below is designed to emphasize these points, through a second and rather different application system.

Example 3

This example is summarized in Figure 6. The system here is a simple travel agent dealing with transport, so the system's decisions are recommendations on the means of getting from A to B. The decision properties are thus those setting constraints on the choice of mode of travel, e.g. date, cost etc. The user is assumed to be both patient and agent. Suppose, then, that we have the initial interchange between the travel agent T, and the client C, represented by the C1, T1, C2 in Figure 6a. The client's reference to trains can suggest three different hypotheses, (a), (b) or (c), about the user properties motivating this, namely that the user is interested in economy (trains are cheap), thinks travel by train is fun, or dislikes flying. Different choices by the system here would naturally suggest quite different responses, for example T2a for hypothesis (a) or T2b for (b). Clearly, if the system's hypothesis is wrong, its response could look very unmotivated to the user. A less risky strategy, therefore, than adopting a hypothesis in the absence of good (i.e. reliable and selective) evidence, would be to seek further evidence, allowing a motivated choice, via the response T2x which explicitly asks the user about his underlying interests. But the user may in fact have no such interests, and may find the further questioning unnecessary and hence irritating. Thus a third, even more conservative strategy would be simply to take the user's input at its face value, and to respond, as in T2y, just by listing the Istanbul train data. (Note that even here, the hypothesis that the user wants trains has properly some uncertainty attached to it, though it is reasonable to regard this hypothesis as much less uncertain than any of (a), (b) or (c).)

What this example suggests is that even where the system is dealing with decision properties, it may be better to be less active in constructing and using models, in an attempt to provide a tailored response, and simply to offer the user sufficient information for him to apply his own model of himself.

The hazards of overcommitting on evidence about non-decision properties, which may be even less adequate than that for decision properties, are illustrated in Figure 6b. Thus suppose that after C1 and T1 as before, we have C2, leading the system to hypothesize that the user has the non-decision property of being religious (being religious is clearly a non-decision property for characterising modes of travel.) This non-decision model may be exploited, with the best of intention, to produce T2p rather than T2q. But in fact, what sufficient reason is there for supposing that religion is the user's motive for not wanting to travel on a Sunday? Attempting to gather more evidence for the hypothesis, for example by questioning the user, would, moreover, be unnecessarily elaborate. The safest, and also the optimal strategy, is for the agent simply to produce the straightforward response T2r.

C1 I want to ask about Istanbul.
T1 So you want to go to Turkey?
C2 Yes. Can I go by train?

train → (a) economy
(b) fun
(c) no flying

T2a Even the train costs $100.
T2b How about the Orient Express?

train → ?
?
?

T2x Do you want economy or fun ...?

train → wants trains

T2y The trains are ... (times, types, costs).

(a) Dialog using decision properties

C1 I want to ask about Istanbul.
T1 So you want to go to Turkey?
C2 Yes. I want to go by train, but not on Sunday.

not Sunday → religious

T2p Unfortunately the only train is a so-called Fun Special on Sundays.

NOT

T2q There's a super Fun Special on Sundays.

not Sunday → not Sunday

T2r The only available train is on Sundays.

(b) Dialog using non-decision properties

Figure 6. Model-using strategies. A travel agent system

6. Conclusion

But is it really necessary to draw such a pessimistic conclusion as this chapter would appear to suggest: namely that modeling is so difficult that we are unlikely to be able to provide systems with any material modeling capacity, and should therefore not even attempt to do this? After all, areas like intelligent computer-aided instruction demonstrate that successful modeling can be done [SLEE82c, CLAN86b].

But it is important to recognize that modeling here is primarily decision-property modeling, made easier by the constrained nature of typical applications, namely teaching a relatively restricted skill like solving certain types of algebraic equation or basic linguistic knowledge. This is indeed not to say that building such systems is easy; but it can be claimed that user modeling here is less difficult than in many other cases because of the narrow focus such systems tend to have and the well-structured domain they normally have. It may also be possible to get leverage in systems like this by building in general characterizations of the kinds of users they may have.

Pessimism is moreover not called for as far as decision properties are concerned. If the system has a human patient, this user has to be modeled. The correct, but of course not novel, conclusion to be drawn from this chapter is just that useful modeling essentially depends on having an adequate set of decision properties, and also that conservative modeling strategies are in order.

It might further be maintained that, even allowing for other application areas more challenging than instruction (as ordinarily treated), there is really no need to take the kinds of modeling aim tacitly presupposed in this chapter seriously. There is therefore no great cause for pessimism. Thus it may be possible to do very useful modeling, in a modest way, allowing for some non-decision properties, basic role distinctions and different functions, but not going beyond the more obvious sorts of information and relatively straightforward uses of them. This is in fact all that we are able to attempt now. But our performance in practice is not very encouraging here, and the examples of the chapter equally suggest that it may not be so easy to identify and deploy even quite obvious information about users.

Adopting a modest approach to modeling does not, moreover, imply that there is no need for the kinds of distinctions drawn in this chapter. Wahlster ([WAHL86a]) has argued, for example, that the distinctions are over-complex, imagining what contortions they would imply for the designer of a marriage counseling system which might have be faced with a concerned mother, son and son's wife. But as Figure 7 shows, simply considering different goals for the system implies a need to recognize the kind of distinctions I have drawn, as without them the system is unlikely to be able to satisfy its goals. Thus focusing on different parties carries different role distinctions with it. (A situation with multiple agents would require equally careful distinctions.)

As a basis for discussing and implementing user models, the general framework presented in this chapter has obvious points of contact with those offered by [BELK84, DANI86a, KOBS*, MORI*, RICH83, WAHL*], for example. But the perspectives adopted in all these analyses vary, and there are interesting differences in granularity. Thus Morik's treatment [MORI*] of interactive settings for user

1. GOAL: help son
   agent : mother
   patient : son
   wife : son's view = P3
   real nature = P1

2. GOAL: help son and wife
   agent : mother
   patient1 : son
   patient2 : wife
   wife : son's view = P1.3
   real nature = P1.1
   son : wife's view = P2.3
   real nature = P2.1

3. GOAL: help mother
   agent : mother
   patient : mother
   son : mother's view = P3
   real nature = P1
   wife : mother's view = P3
   real nature = P1

Figure 7. Goal perspectives and modelling implications. A marriage guidance system. P1 = objective decision properties, P3 = subjective decision properties

and system actions in terms of the system elements open to change, the means of change, and the agencies of change, provides a rather different and in some ways more, though in other ways less, detailed analysis of modeling contexts than that provided here, suggesting that it may be useful to try to put the two together.

The general framework offered here is also neutral with respect to what may be called the orientation of a system to a particular generic task, for example explanation [VANB86b, VANB86a, CLAN86b, SLEE85], especially explanation seen as an instructional activity. The implications of these generic tasks obviously have to be taken into account in user modeling: my point here has been rather to emphasize the need to look at user modeling outside the popular generic tasks. Thus, explanation may in some cases be at most a minor system need.

Again the analysis offered here makes only weak assumptions about cooperation: cooperation with the patient may be minimal or non-existent, and sufficiently attained with the agent with very little effort. However, it is clear than in individual applications it may be necessary to take into account the complexities of cooperative behaviour illustrated by [AIRE86, JAME*], as well as the need to apply specific response strategies in dialog including, for example, explicit clarificatory and negotiatory dialog, and to tailor responses to the user as illustrated by [CARB*,

COHE*, GOOM86, MCCO88], for instance. The chapter's examples assumed such requirements but did not analyze them, since the chapter was more concerned with emphasizing the great variation of user modeling contexts and hence allowing for the fact that questions of cooperative response may not arise though user modeling itself is appropriate.

Beyond making a broad distinction between static and dynamic subjective properties, the chapter does not attempt to give the kind of detailed analysis of the properties of beliefs, goals, plans, etc. carried out by [KOBS*], or to consider the detailed relations between the user's intensional states and the reference world investigated by [VANB86b, VANB86a, CHIN*, MCCO88]. The broad view adopted has to be filled out in this way, as appropriate to individual system contexts. However the framework provided does draw attention to the fact that in many contexts it may be proper to talk about user modeling which is not exclusively devoted to, or dominated by, the user's intensional states, and also that users may, for example, have goals without also having plans.

The chapter is also deliberately focused on the user model as opposed to any discourse or dialog model, say of the kind illustrated by [LITM85], taking it for granted that these two are distinct but not analyzing or defining the precise relationship as this is done, for example, by [SCHU88, CHIN88a, COHE88, KOBS88, MORI88, SPAR88, WAHL88]. The analysis of user models clearly has to be supplemented for implementation purposes by a view of their relation to discourse models (and any other models like a world model). The object of the chapter is to emphasize the fact that in complex systems the user model cannot be equated with the discourse or dialog model, i.e. it is necessary to distinguish communicative properties of the user from real properties of the user, but then of course also to allow for very subtle relationships between these. It is clearly possible, in particular, to envisage further complications of the various kinds illustrated by [JAME*], [WILK83] and the HAM-ANS hotel application [HOEP83b], where dialog participants may be deliberately deceitful, or may mix individual and institutional attitudes.

The examples in the chapter illustrate the variety of possible sources of user modeling information also considered by [WAHL*], for example; and it is clear that though, as indicated, it may be very difficult to rely on structural relations between different properties or property types, especially for a wide scatter of non-decision properties, modeling has to look for relations between objective and subjective properties of the kind considered by [BROO86, COHE*, MORI*, RICH79b]. How these are related is just one element of the specific modeling process; and the framework for modeling given in the chapter has of course to be supplemented by an account of actual modeling procedures, i.e. of the inference mechanisms required to construct and deploy models, like those described in [WILE86] and for goals and plans in [VANB86b, CARB*, GOOM86, POLL86a]. The aim of the chapter has been to indicate the need to consider the various aspects of modeling, especially in relation to system functions, before detailed procedures are designed.

The analyses of the information-seeking interactions between the users and staff of specialised library services carried out by Belkin and his colleagues [BELK84, BROO85, BROO86], where user modeling is of the essence, serve to bring out just how hard modeling is in such complex situations. Establishing the nature of the

library user's information need is user modeling in the broad sense of the present chapter, though Belkin and his colleagues themselves give "user model" a much narrower meaning. Characterizing someone's scientific literature needs requires a large apparatus of 'functional experts' to deal with the various facets of the modeling, and the application area is one which clearly illustrates the intractability of modeling in those cases where the user's subjective, i.e. mental properties are in question, and even more, in those cases where the user's goals, plans, beliefs, etc. are dynamically changing through the interaction. But the particular problem in this case is that the modeling information is necessarily inadequate, because the user is seeking information which, as he has not yet got it, he cannot properly specify. Brooks' and Daniels' detailed analyses [BROO86, DANI86b] of human information interviews show just how difficult and how complicated capturing information needs and other relevant properties of users is: we are nowhere near being able to extract information from dialogs like this automatically.

However, their interview transcripts also show quite clearly that even human beings may not find it useful or possible to seek to carry modeling very far. The overall message of my chapter is that as the situations in which user modeling may be envisaged vary enormously, because they depend on combinations of user populations, system tasks, and application domains, the strategy to be adopted for modeling has to be evaluated for each individual system. But it may well then turn out that, quite apart from the fact that we cannot yet, for example, provide sufficiently powerful natural language processors, the conjunction of necessary ignorance about users and system resource limitations e.g. in relation to real time operation, may mean that if modeling is done at all, it can only be done in a very limited way. It is nevertheless still user modeling, and may be as adequate as is in fact required. This view is, on the one hand, a justification for the use of stereotypes, as in [CHIN*], for example, and on the other for such modest applications of modeling as the individualised parsing proposed by [LEHM*]. It also has the important advantage of being unlikely to mislead the user into thinking the system is smarter than it is.

Acknowledgements. This work was carried out during the tenure of a GEC Research Fellowship. I am grateful to Richard Young and to the participants in the Workshop on User Modeling held at Maria Laach in 1986 for discussion and comment.

7. References

[AIRE86], [BELK84], [BROO85], [BROO86], [BUCH84], [CARB*], [CHIN88a], [CHIN*], [CLAN86b], [COHE88], [COHE*], [DANI86a], [DANI86b], [GOOM86], [HOEP83b], [JAME*], [KOBS88], [KOBS*], [LEHM*], [LITM85], [MCCO88], [MORI88], [MORI*], [POLL86a], [RICH79b], [RICH83], [SCHU88], [SLEE82c], [SLEE85], [SPAR84], [SPAR86], [SPAR88], [VANB86a], [VANB86b], [WAHL86a], [WAHL88], [WAHL*], [WILE86], [WILK83]

13 User Models and Conversational Settings: Modeling the User's Wants

Katharina Morik

1. Introduction

In general, the cycle of system development can be viewed as consisting of four phases:

- theoretical considerations, design
- practical implementation and experimentation
- evaluation of the system, documentation
- reorganizing the field of research, thus using the experience which has been gained by workers in the field.

The fourth phase is indicated by activities such as defining the technical terms of the field systematically so that deficiencies in coverage of the accomplished work can be detected. With a new or modified view of the field, the cycle then starts again. In user modeling, the fourth phase currently seems to have begun.

This chapter follows the trend by first clarifying terminological issues. Second, a general view of user modeling is presented. Conversational settings of interactive systems and types of model changes are categorized and illustrated in the context of the system HAM-ANS[1]. Already completed user modeling is classified and the state of the art in user modeling determined with respect to this framework. Areas where further research is needed are highlighted. The following main problems are pointed out:

- the integration of non-mental, non-communicative actions,
- the integration of the user model acquisition and the interpretation phase, and
- the change of user models.

Third, the framework is in turn applied to the HAM-ANS approach to user modeling, particularly focusing on modeling the user's wants and evaluation standards. There, the fact that a non-communicative action of the system is possible forces the system to produce a wider range of speech acts than mere information. Finally, we propose a way to integrate the acquisition and the interpretation phase

[1] HAMburg Application-oriented Natural-language System [HOEP83b]

and to use this procedure for the update of the user model in the HAM-ANS approach, thus overcoming the limitations of accomplished work. The central point is that on the one hand, representing the evaluation standards of the user allows the system to recognize the user's wants from her/his utterances. On the other hand, the evaluation standards may be inferred from the utterances by using the textlinguistic notion of "isotopy" in a new way.

2. Terminology

In order to clarify the term "user model", the closely related terms "dialog memory" and "agent model" must be considered as well. A dialog memory can be viewed as part of a user model, namely the part which represents the dialog-dependent knowledge of the user [MORI84]. Entries out of the dialog memory may cause entries in the user model, and entries of the user model may support the interpretation of an utterance, the interpretation then being stored in the dialog memory. However, in order to keep technical terms precise, user modeling on the one hand and building and exploiting a dialog memory on the other hand should not be identified. This would lead to a reduction of what user modeling is about by disregarding all aspects other than dialog-dependent knowledge of the user as known to the system, while in fact there is some information which is to be covered by a user model and which may not be covered by a dialog memory.

Let us think, for example, of a visit to the dentist's. The dentist will have some expectations concerning the client even before the client says a word. This is due to the conversational setting, the roles of dentist and client. The same two persons meeting in another environment (e.g. at a horse race or the opera) would not rely on the dentist-client expectations but on the information which then belongs to their roles. Of course, by analogous reasoning, the dentist may transfer the way the particular person plays his or her role as a client to the way this person will behave at the races. This transfer, however, is also based on role-specific information.

A *dialog memory* contains all beliefs that can be inferred with certainty from utterances, so that they belong to the mutual belief space. These are beliefs a speaker cannot deny when uttering a sentence. For example, a speaker cannot talk of "today's king of France" without the obligation to claim that s/he believes there is a king of France today. Thus, presuppositions which can be inferred from articles or question particles belong in the dialog memory. The objects and their properties introduced in a dialog are typical entries in a dialog memory as well. The dialog memory is structured so that it can be used for such things as determining the appropriate description (definite or indefinite determiner), anaphoric expression or characterization. The linguistic rules which determine the inferences are valid and binding for all conversational settings and are not parametrized according to assumptions about the user. If, however, linguistic rules are specialized for a particular user in a particular conversational setting [LEHM*], this is clearly not a matter of the dialog memory but of the user model, because then these rules are not valid and binding for all users.

A *user model* is an agent model where

- the situation in which the agent plays a role is the conversational setting of a system-user interaction concerning a particular task, and
- the agent is the person who solves his or her problem with the help of the system, either directly or through an intermediate person.

The user model allows the system to adapt to different users, tasks, and roles.

This definition links "user model" to the more general "agent model". An *agent model* is a knowledge source which contains explicit assumptions about a person acting in a certain setting. As psychological studies have shown there is a kind of consistency in the image of other persons: personal traits are not all viewed independently of each other. There are stereotypes which group several personal traits together and stereotypes themselves are combined following a naive psychology. In this sense an agent model forms a consistent image of the agent. If the person modeled by the system is the protagonist of a story and the system understands the story, we speak of a *protagonist model.* In principle, modeling a person in the actual[2] conversational setting and modeling a person in another setting (which may also be a conversational setting), is not very different. However, for a system which plays no role in the story it understands, but does take part in the interaction with the user, user modeling is a more complex task.

The definition excludes from the user model the information about the user which is not relevant to the user's role in the system-user interaction or to the task under concern. Thus, in the example of [WAHL86a], where the age and the bonus of the user is stored in a database and the system can answer questions about the user's bonus and status, the crucial point is whether this information at least partially determines the user's role in the interaction and is related to the task of system and user. If, for instance, the system helps employees to plan their business trips, and if the decision whether the traveler is supposed to take a train or an airplane depends on his status, then this information is related to the task. As [SPAR84] named it, the information is a decision property. Now, another requirement from the above definition is that the system uses the information to adapt to different users, tasks, roles. Thus, it is not sufficient that the system uses the information in some way, but it must use it systematically for individualization. In the example, the user's status constrains planning, ruling out several arrangements. However, even with these modifications, the example need not fit into our intuitive understanding of user modeling. This is due to the unrelatedness of the entries in the database – they do not necessarily form a consistent image of the user. But this is also a definitional property for a user model as inherited from the definition of an agent model. Therefore, in order to transform Wahlster's database example into an example of a user model, we must perform an additional modification: the system must have a process which forms a consistent image out of the database entries, this image then being the user model.

Another problem with defining a user model is whether the person directly interacting with the system or the person whose goals are involved in the interaction

---

[2] A discussion of the relation between the notions "dialog memory", "discourse model" and "user model" can be found in [MORI88].

should be modeled. In the above definition the person to be modeled is the one whose task corresponds to the task of the system. The definition closely relates the user model to the task which the user wants to perform with the help of the system, and thus to the system's task. Task and role specificity separates a user model from a general agent model. The problem with determining one person to be modeled is that several persons can be involved. Let us think, for example, of a group which asks for a price reduction in a train station environment. At least in Germany, there are several special offers for groups but also for elderly people or children. In order to find the best arrangement, it is not sufficient to model the group as one entity; instead, the members of the group have to be modeled along with the possible ways of grouping them together.

Another illustration of the problem of restricting user models to modeling one person is the embedding of situations (see [SPAR84] for a discussion of agent/patient modeling). The intermediate person directly interacting with the system is the user to be modeled concerning individualized linguistic rules. The person on whose behalf the interaction takes place is the user to be modeled concerning the task.

In most cases, however, there is not more than one person with respect to the task. This person may be introduced indirectly ("the brother of the wife of my son ..."), but this does not change the situation of an intermediate person as the direct user, where the person to be modeled is the one whose problem has to be solved by the system. Moreover, the task-related situation may be an embedded situation as well: the problem of a mother with the problems of her son. If she asks, for instance, whether she should wait and see or do something about the son's problems, the answer also depends on the kind of a problem the son has. In this case, the mother is the person to be modeled according to the definition above. The son is the protagonist of the mother's story.

3. Conversational Settings

We have closely related user modeling to the tasks and roles of system and user. Let us now develop a systematic view of the situation. We start with a descriptive example of a conversational setting, introduce three factors from which we can generate several conversational settings (3.1), and discuss three typical settings of interactive systems using HAM-ANS examples (3.2 – 3.4). The general view of user modeling is accomplished by collecting change types (3.5) and introducing types of user model acquisition (3.6). This will allow us to determine the state of the art and point out underdeveloped areas in current research (3.7).

3.1. Factors of conversational settings

Figure 1 shows Mrs. User thinking of a hotel room she wants to book. She communicates with a computer system which also has some ideas about a hotel room, so both have a domain model. In most man-machine interactions, the dialog is about enhancing the user's domain model. These are *information-seeking dialogs.*

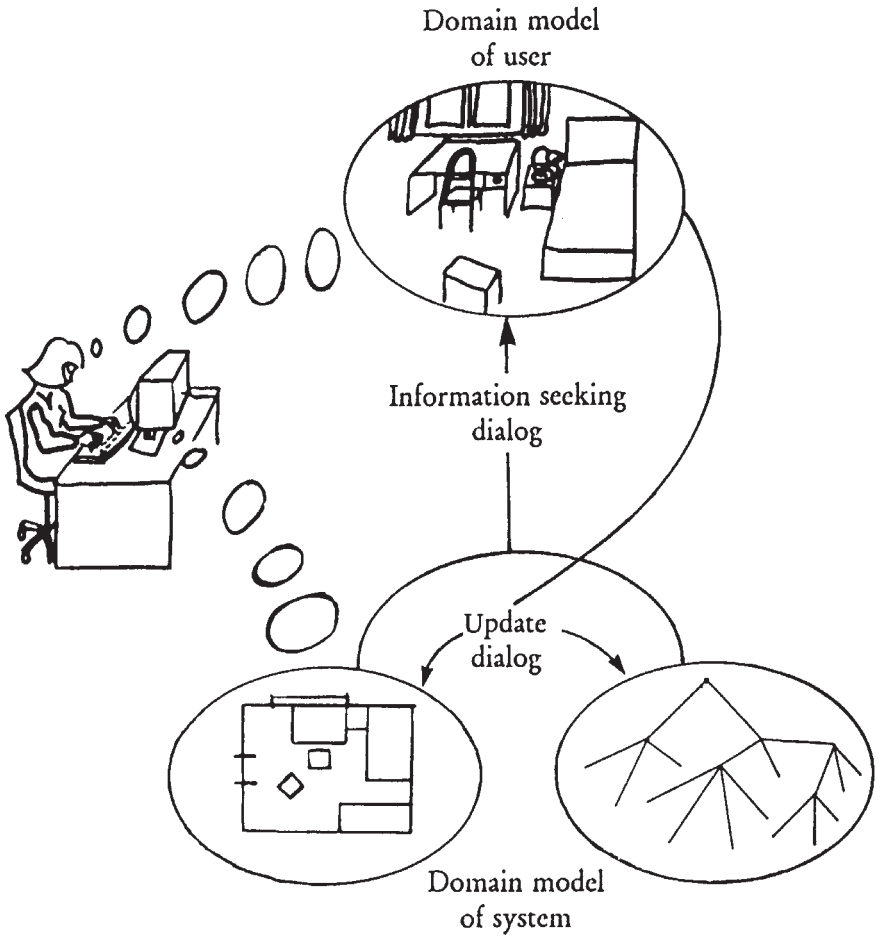


Figure 1. Mrs. User and the Computer

Of course, the dialog could also be about filling the system's domain model. Then it is an *update dialog*. For an update dialog, the system must be able to change its domain model. Although for system developers these two kinds of dialogs are difficult enough, they are not sufficient, because reducing differences between the domain models of the communicative partners is not what dialogs are all about. Mrs. User's wants are not directed towards a model – even if it can be manipulated in the process. She wants to book a real room, she really wants to use it. Dialogs where the system is able to perform a non-communicative action because of the dialog are called *action-achieving dialogs*. Human communication most often deals with the real world, although models are used for communication. The obligations resulting from dialogs refer to the real world and not only to models. In the above example, the hotel room should be really booked if the user wishes. Taking this into consideration gives a more general view of dialogs and their conversational settings.

Viewing actions as the most important topic with respect to the goals of a dialog and the dialog itself, we characterize the conversational setting according to the actions of system and user as they are modeled by the system, i.e. as far as they are perceivable and tractable for the system. Actions may occur in the

- domain of discourse (i.e. part of the real world),
- domain model (i.e. knowledge about the domain)
- knowledge about the partner (i.e. knowledge about the partner's beliefs and wants).

If Mrs. User in our example wants to have a TV set in her hotel room,

- the system could have a TV set brought into the room, thus changing the domain of discourse,
- update the knowledge of the user about the room by informing her that there is a TV set in the room,
- update the domain model after being informed by the hotel manager that there already is a TV set in the particular room,
- update the user model by recording the want for a TV.

These changes may be performed by several different means:

- non-mental, non-communicative actions (e.g. putting a TV set into a room)
- communicative actions (e.g. informing or being informed about the presence of a TV set)
- observation (e.g. looking into the room to check whether there is a TV)
- inference (perhaps the user's want was inferred from an utterance meaning).

A change by inference is made if a proposition is deduced. If we adopt the view that all implications of available knowledge are known as well, the derived proposition is already known. Therefore, in a narrower sense, no change occurred[3]. Of course, inferences and observations are a means for changing models and not the real world. So all of what is to be changed cannot be combined with all means of change, but some means are specific to certain kinds of change.

So far, the actions were on the side of the system. But it is also of importance whether the user plays a role in the domain of discourse, or not. Mrs. User in our example plays a role in the domain: she'll become a hotel guest using the room she booked. In the standard computer applications, however, the user is not involved in the domain (or only by chance, not systematically): think of all the supplier-parts database applications, for instance.

Using the three factors

- what can be changed by the system,
- by what means, and
- whether the user plays a role in the domain

we may now generate several conversational settings. We show three typical settings, each first without the user playing a role in the domain and then with the user playing a role in the domain. For illustration purposes, we discuss HAM-ANS

[3] cf. the discussion of the wise men puzzle in [KONO84] for the (im-)plausibility of the assumption that we know all implications of our knowledge.

applications and determine how they fit into the conversational settings. HAM-ANS is a natural-language access system to diverse application classes [HOEP83b]. The application classes are exemplified by three applications. The first application provides natural-language access to a relational database system. There, HAM-ANS is connected to a shipping and fishery database. In the second application, which exemplifies natural-language access to a scene analysis system, the system answers questions about a street intersection. A film consisting of about 70 snapshots was analyzed, the movements of objects (pedestrians and cars) represented as trajectories[4]. Questions are then answered on the basis of the internal semantic representation. In the third application, HAM-ANS plays the role of a hotel clerk, informing the user about the hotel and its rooms, recommending a room, and booking it if the user wishes.

3.2. The First Conversational Setting — Information-Seeking Dialogs

As the *first setting* let us take the most common situation for AI interactive systems. The system brings about changes only on the user's mind by communicative actions. It changes its own dialog memory accordingly.

If the user plays no role in the domain of discourse and the user's task is not modeled, overanswering or overunderstanding (i.e. understanding ill-formed input) must resort to linguistic clues and general rules of cooperativeness [WAHL83]. The first application class of HAM-ANS, (front-end to a database system) fits into this setting. The domain model (i.e. the database schema as conceptual knowledge and the database content as referential knowledge) is static. Neither user nor system play any role in the domain of discourse. The system informs the user about ships and their cruises. Different users can have very different tasks for which they use the information from the database. The meteorological data gathered during a scientific cruise might interest a meteorologist in quite another way than it would interest a shipowner. Because tasks of users are not modeled in that particular HAM-ANS application, providing the user with the information s/he wants is guided by principles of cooperativeness, which are not specific to a particular user or task[5]. The system doesn't need to decide which action is appropriate under certain circumstances, but has a built-in, implicit strategy: answer questions about the domain, being as informative as possible. However, some situations occur during the dialog in which the system may choose to deviate from 'terminal communication': for instance, to output the result of a database query on the printer. This is, of course, a communicative action, but it differs in kind from the default interaction in the dialog. In these cases, finding out what the user wants is realized by just asking her/him.

If, on the other hand, the user plays a role in the domain of discourse, her/his actions are to be modeled. An example of this conversational setting is the Montreal train station scenario [ALLE79, LITM85]. The actions of the user are represented

---

[4] The same conversational setting underlies NAOS [NEUM83].

[5] As Marburger has shown [MARB87], dealing with linguistic rules of cooperativeness enhances dialog performance considerably.

together with their mental preconditions. This information is used to clarify the user's information requests. Determining exactly which information the user needs leads to a more finely grained model of information request speech acts, as has been shown by the above mentioned work. Here, the system finds out which information the user needs. The overall goal and the possible actions of the user are known to the system before the dialog.

3.3. The Second Conversational Setting — Reporting

The *second setting* is that the system changes its domain model because of observed new facts. We might think of a vision back-end system which observes state changes in the real world and a natural-language front-end system which communicates the events to the user. The second application of HAM-ANS would fall into the class of this setting, if the domain model were not fully constructed before the start of the dialog. Observation and communication do not take place at the same time. The domain model is treated as static, although the domain consists of events in a time sequence[6]. User and system do not play any role in the domain or domain model. The only state changing action is the dialog itself. As in the case of the database application, a user model concerning the user's wants is not constructed nor exploited; instead, only general linguistic rules are used in order to find out what the user wants to know about the street crossing and the events there. Of course, the system's knowledge about the user's dialog-dependent knowledge is used for generating descriptions [JAME82].

3.4. The Third Conversational Setting — Action-Achieving Dialogs

Let us take as the *third setting* a situation that, because of its complexity, has to my knowledge not yet been fully realized in interactive AI systems. The system performs changes in the domain by non-communicative actions, changing its domain model accordingly. At the same time it is communicating with the user. An example of this would be an intelligently interfaced robot. In this case, the user does not perform any action in the domain but controls the robot's actions.

If the user plays a role in the domain as well, s/he might want the system to change the state of affairs, thus enabling her or him to perform a certain action. In the example above, Mrs. User wants to watch TV, and the robot's bringing in the TV enables her to do so. In a restricted manner, from the user's point of view, HAM-ANS playing the role of a hotel clerk is also an example of the third setting. The system simulates the booking of a hotel room for the user, fulfilling a prerequisite for the user's intended action (to stay in the hotel for a while)[7]. Booking is a non-communicative, although not non-mental action which changes the state of affairs. Although other changes are not modeled, the interior of the rooms cannot be changed, for instance, because of the booking the dialog is a (restricted) action-achieving dialog.

---

[6] Thus, time adverbs and verbs of locomotion could be modeled [MARB83].

[7] HAM-ANS was never used in a real hotel.

As empirical work has shown (e.g. [NOTH84, SCHG79, POLL83]), the most salient difference between information-seeking and consultation dialogs is that the wants of the client are fully specified before the dialog or not treated at all in the former, whereas the specification is found interactively in the latter. A *consultation dialog* is a specification phase for the intended action. The unspecified goal of the user is known to the system before the dialog: the user wants to book a room. The specification is to be found out from the dialog. The system recognizes the user's wants and selects the particular room which suits the user best. Regarding the specification which is determined interactively during the dialog, it is a step in the direction of consultation dialogs.

Whatever the role of the user, as soon as the system is able to perform changes not only on the model, but on the domain itself, the dialog may include speech acts other than information requests and information giving. In other words, action-achieving dialogs show a variety of speech acts. On the side of the user, this is due to the justified belief of the user that the system is able to fulfill the request. A wider range of speech acts stresses the problem of speech act recognition. Ambiguities at the level of speech acts must be resolved. For example, the user input

Could you reserve a single room ...?

can be interpreted as the question of whether there is still a single room available, a question about the capabilities of the system, or as the request to book the room[8]. The system must decide which knowledge sources it needs to change. The first two cases require merely a communicative action which changes the mutual belief space (the dialog memory). In the third case, the system must perform a change in the domain and hence in the domain model. Then it must decide whether an additional communicative action is necessary. The number of choices which the system must make is considerably greater than if the system is restricted to communicative actions only.

Also on the side of the system, this conversational setting requires a variety of speech acts to be handled. As opposed to nearly all other natural language dialog systems, HAM-ANS in its hotel application is able to *generate* speech acts other than the inform act as an answer to a user's information request. HAM-ANS can ask the user, recommend a room, and – of course – answer questions[9].

3.5. Change Types

We have characterized conversational settings by the changes that system and user may perform. From the point of view of a system developer who builds a system for a particular setting, the complexity of her/his task depends on the number of different actions which system and user are allowed to perform. But there is one further point: the complexity of how changes are modeled. We want to distinguish five *change types* ordered here by increasing complexity:

---

[8] For more precise rules using assessments of speaker/hearer for recognizing the speech act cf. [MORI82:239ff].

[9] A more detailed description of generating recommendations is given below (4.1).

- addition
- replacement
- modification
- revision
- development

Although these terms apply to any model of changes, changing the domain, the domain model, or the dialog memory, let us illustrate their meaning by discussing user models. By *addition* we mean adding new information without any consistency or integrity checks[10]. New information is entered into the user model. The entries are assumed to be independent from each other, so that adding a new fact does not change the already stored ones. Contradictions are not even noticed in this change type. For user models which form a coherent image of the user, this change type is not applicable.

By *replacement*, we mean that a new piece of information overwrites an already stored one. This is the simplest way of handling contradictions: if a new piece of information is in conflict with existing entries, just throw them out! Only simple contradictions are recognized – that is, those concerning the same property. For example, a contradiction between two assumptions concerning the solvency of a user can be detected: "The solvency is high" is in conflict with "The solvency is medium". The new assumption here overwrites the older one. However, in this change type all properties of a user are assumed to be independent. Thus any relation between them are not represented. For instance the relation between the profession and the solvency is not represented.

Just as replacement, *modification* refers only to information about a single property of the user. However, here the properties are not assumed to be independent. So, it allows for reenforcement or weakening of entries. New information may modify existing entries, or existing entries influence the way new information is entered, respectively. Modification concerns the level of belief or importance of a certain property, not the value of a property.

In the stereotype approach (in the sense of [RICH79a]) the user facets are independent from each other, thus a new facet may simply be added. From the facets properties of the user are inferred. These are not necessarily independent from each other. In HAM-ANS, the properties which are derived from the user facets are user requirements of a hotel room. If two facets create the same requirement, the importance of this user want is increased. If two facets create different instances of a requirement, modification is needed to create the appropriate 'compromise' requirement [MORI85a]). Entering a new requirement directly, i.e. without inferring it from a facet, employs the same modification process. If a facet has to be changed, the overall process of inferring properties from facets with the modification included starts again. This leads to the same result and is much easier than revising the properties according to the changed facet.

For *revision*, the representation must permit the tracing of all entries which are based on the same (false) assumption. The conflict resolution concerns more as-

[10] If there is no closed world assumption, it is a monotonic update.

sumptions than the conflicting two. Dependencies among entries in the user model are thus taken into account[11].

By *development* we mean a model of changes that include the history of the changes. As [BARB82] points out, using "no-good" sets for a false assumption and its dependents – as is most often done in revision – is not a history keeping facility. The development type of change is of particular importance to user models of tutoring systems. There, not only the level of expertise at a given point is important, but also its progress. If two assumptions about the user's performance level are in conflict, the user's performance level can have been raised in the meantime due to her/his learning.

In the terminology of [SPAR84, SPAR*] the first four change types refer to static modeling, i.e. the user remains the same but the system changes its knowledge about the user. The development type refers to dynamic modeling in that properties of the user are not necessarily viewed as permanent. We do not, however, identify the development change type with the system's task of recognizing plans and goals of the user, as does Sparck-Jones. Systems like those of [ALLE79, LITM85], which recognize the plan of the user to a certain extent do not model the user dynamically, because the user's overall task is not permitted to change. Representing actions is independent from using the development change type.

3.6. User-Driven and System-Driven Acquisition

An important factor for the complexity of the system-developer's task is whether the construction of a model for a particular user is system-driven or user-driven. *System-driven* means that the acquisition of the user model is separate and prior to the interaction which attempts to achieve a user's goal. The dialog step need not be exclusively restricted to the acquisition of a model for the particular user. Prompting the user to characterize her-/himself (as was done by GRUNDY [RICH79a]) is just the special case of restricting a dialog phase or dialog step for building up the user model. In any case, separating the acquisition phase from the phase where the user model is exploited reduces the choices which the system has to make in order to behave cooperatively.

User-driven means that the user model is constructed during the entire dialog, e.g. simultaneously with achieving the user's goals. No particular steps are reserved for this part. Thus, the system does not know before understanding the utterance whether the utterance has to be analyzed by the component which constructs the user model. The need to recognize which input enriches the user model increases the complexity. Thus, a user-driven acquisition of the model of a particular user is more difficult than the system-driven one, because acquisition is integrated in the other activities of the system.

[11] "Revision" is used in broad sense here, too. It covers sophisticated truth maintenance systems as well as if-added procedures in frame systems.

3.7. Lessons from the General View

We have described conversational settings by the roles system and user can play. We introduced five change types and we distinguished two ways of acquiring particular user models. Now we want to use this framework in order to determine deficiencies in the field. Table 1 summarizes the application of the framework to related work.

Table 1

system playing a role in the domain	user playing a role in the domain	modeled aspects of the user	use of user model	change of user model	system-/ user-driven	author
no	no	mutual beliefs	over-answering	addition	user-driven	[WAHL83]
simulated	yes	plan, knowledge	over-answering	no	system-driven	[ALLE79] [LITM85]
no	yes	plan,goals mutual beliefs	speech act generation	no	system-driven	[APPE82b]
no	yes	plan	reference resolution	no	system-driven	[GROS81]
no	no	wants	recommenda-tion	revision	system-driven	[RICH79a]
simulated	yes	wants	recommenda-tion	no	system-driven	[MORI85a]
no	yes	plan, knowledge	over-understan-ding	revision outlined	user-driven	[CARB*]
no	no	impression of rooms	commenting on rooms	no	system-driven	[JAME83]
no	no	(mutual) beliefs	description generation	addition	user-driven	[KOBS85a]
no	yes	level of expertise	advice	yes-	user-driven	[CHIN*]
no	yes	plan, knowledge	advice	no	user-driven	[FINI83]
no	no	view on objects	correcting misconcep-tions	no	user-driven	[MCCO*]

Although it doesn't cover all the work in the field, it clearly shows some points where further research is urgently needed. Integrating *non-communicative actions* into the dialog pushes the need for speech act handling and a more flexible system design. If it is not a justified default assumption that the user's input is always information-seeking, a larger variety of speech acts are to be recognized. In an information-seeking dialog, informing and requesting acts are to be differentiated

by the propositional content (e.g. request for justification, meta-communication, request for an "inform-if", etc.). In an action-achieving dialog, speech acts other than informing and requesting come into play (e.g. command, recommendation, warning, etc.). Thus, the system must not only determine the propositional content but also the speech act. Moreover, the system has to decide which action it has to perform in order to act cooperatively. This requires a system architecture including a decision process (some sort of a dialog management component) which selects among communicative and non-communicative reactions. The embedding of language use into non-linguistic behavior thus comes to the foreground.

Integrating the acquisition of the user image into the normal processing of the dialog is lacking for almost all systems, which systematically model and use more aspects of the user than his dialog-dependent knowledge. Of course, the mutual belief space is built up during normal dialog processing. Modeling the user's beliefs and directly uttered goals is always integrated into dialog processing – if it is done at all –, because it is part of the dialog model [GROS86]. On the other hand, integrating "naive psychology" into the processing of dialogs widens the scope of the system. An additional aspect of what is going on in dialogs is then to be incorporated into procedures. Carberry's work [CARB*] tackles this problem with the user-driven acquisition of underlying plans and goals of the user.

The *revision* of user models is another weak point of current systems. Most of the systems which update the user model during the dialog do so merely by adding entries.[MORI85a] describes a modification procedure (cf. 4.1 below). Only [RICH79a, RICH*, CARB*] outline a revision procedure.

In summary, we see as shortcomings of accomplished systems:

- the integration of non-mental, non-communicative actions,
- the integration of the user model acquisition and the interpretation phase, and
- the revision of user models.

4. User Modeling in HAM-ANS

HAM-ANS is a dialog system. A dialog system can be contrasted with a question-answering system which is only able to handle questions posed to it in isolation. It requires constructing dialog memories during the dialog and using them for a variety of tasks, e.g. understanding and generating ellipses, determining reference and coreference. New objects are introduced into the dialog with a characterization and indefinite description. In order to keep track of what the user knows as a result of the dialog, the system builds a referential net for the partner (the user's domain model as known to the system) and establishes connections between the objects there and in the referential knowledge of the system (the referential part of the system's domain model). This knowledge about the user's beliefs is used for determining and generating the appropriate reference.

Whereas the dialog-dependent knowledge of the user can be handled in all applications, the user's wants are only modeled for the hotel scenario, being an instance

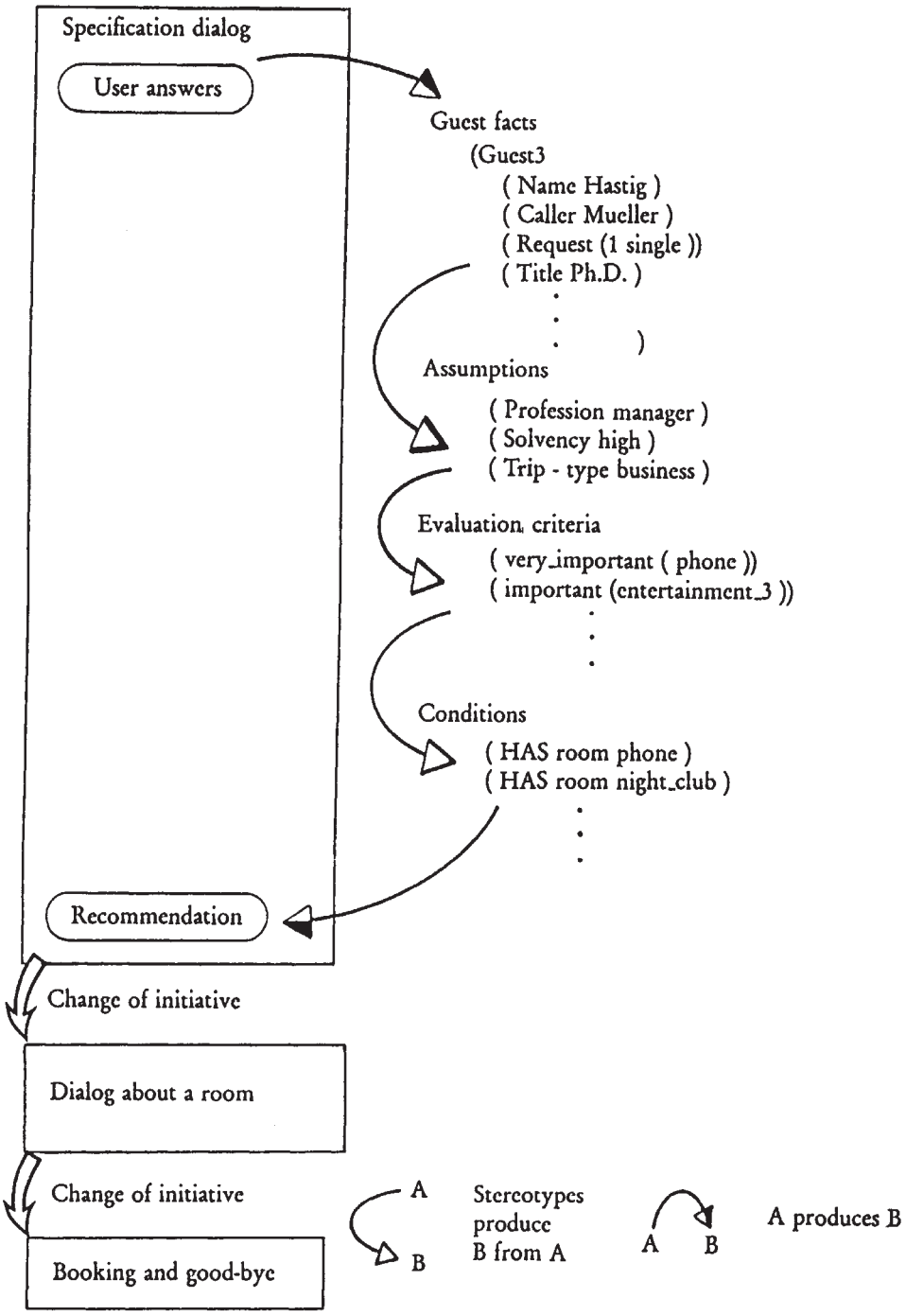


Figure 2. Modeling user's wants in HAM-ANS

by inferences. For instance, the entertainment opportunities are grouped: cultural entertainment, night life, family entertainment etc. Through inferences, they are specialized into, for example, theater, museum, opera. Thus user modeling in HAM-ANS is a multi-level approach: the given user facts, the supposed assumptions about the user, her/his evaluation criteria and finally the conditions which match the properties of the hotel or its rooms (see Figure 2).

of the third setting, because only there the user is really concerned with the domain of discourse.

4.1. Modeling the User's Wants

In this section, modeling the user's wants as implemented in the hotel application of HAM-ANS is described.

It is well known from linguistics and philosophy that value judgements and several speech acts refer to some sort of evaluation standard [HARE52, GREW78, ZILL82]. In particular, recommendations rely on the presumed evaluation standards of the hearer. If a recommendation is to be actually implemented by the system and not just read into the system responses by the user, the system must be able to recognize the user's evaluation standards, derive the demands placed by the user on objects or situations.

In the hotel setting the user may well be asked whether a room is to be booked or not. Asking her or him for all her/his evaluation criteria, however, is a clumsy behavior. Rather, the criteria should be inferred from the user's utterances and should then be used for selecting "interesting" properties of rooms or of the hotel, generating the appropriate speech act, and deciding what speech act the user has performed.

A system which makes recommendations of novels is GRUNDY [RICH79a]. The basic assumption underlying the consultative function is that people are interested in books in which characters have the same type of personality as they themselves have. Therefore, recognizing the personality type of a user can be used directly for the evaluation of books. For HAM-ANS the utilization of knowledge about the user is not so straightforward.

As in GRUNDY, the specification phase in HAM-ANS is system-driven. The system asks the user about all the information which is normally required for a hotel reservation. No psychological characteristics are requested. The explicit specifications given by the user, i.e. from when to when, what kind of a room should be booked for whom, are stored as certain *facts* about the wants of the prospective hotel guest (who need not be identical with the user, of course). From these *facts* different independent *stereotypes* are inferred. These stereotypes are stored as assumptions about the hotel guest. They are not certain. The stereotypes differ from user classes [CHIN*] in that a single user belongs to several different *stereotypes*. They are mixed together in order to form the profile of the particular user. Thus, a user is not merely recognized as belonging to a certain class, but the user profile is constructed out of several stereotypes. This allows for representing a wider variety of user models in a flexible way.

In the next step, *evaluation criteria* are created from the stereotypes. Each stereotype creates a bundle of criteria. All the criteria are put together and are sorted into those concerning the hotel and those concerning the room. If two stereotypes produce the same criterion, its weight is strengthened. This is the basic *modification* process of the HAM-ANS user-modeling component that has been mentioned above. Some criteria are directly related to properties of the hotel or its rooms as they are stored in the referential knowledge. Others are expanded

The requirements of the prospective hotel guest as assumed by the system can then be used for different purposes. The relation to the dialog structure has been described elsewhere [MORI85a]. The most important purpose of evaluation criteria is the handling of speech acts. HAM-ANS, in its hotel application, is able to *generate* speech acts other than the inform act as an answer to a user's information request. According to the result of the match between the user's requirements and the properties of the hotel and its rooms, the appropriate recommendation is chosen. In order not to leave important information buried in the propositional content, the speech act *recommendation* is split up into

- strong recommendation
- weak recommendation
- restricted recommendation
- negative recommendation.

The result of the match between evaluation conditions and actual properties is a list which consists of two parts, namely the fulfilled and the unfulfilled criteria. This list serves as a justification for the selection of the appropriate speech act and determines the propositional content of the restricted recommendation. Only those features of the hotel or room which seem to be of interest for the user are verbalized. For example, in the dialog with a wealthy manager, the system cannot warmly recommend any of the room categories. It therefore comes up with a restricted recommendation as follows:

SYS: The room does not have a TV but a bar, a desk, a telephone and a seating arrangement.

The system doesn't mention the view, the radio, or the feather bed[12].

The representation of the assumed criteria and the derived requirements is a *qualitative* one. The information is not encoded in numerical weights but stored as contexts of semantic networks in the same way as the domain model, thus allowing for explanations.

4.2. User-Specific Overanswering

We have seen that the acquisition phase is system-driven. The system asks the user for the facts about her/his planned stay at the hotel. The system infers stereotypes and evaluation standards of the user and recommends the best suited room category. The system then passes the initiative to the user who may put questions about the particular room category. In this user-driven dialog phase, the inferred user requirements are of good use as well. They allow for determining the underlying intent of the question and thus for user-specific overanswering. See the following examples for illustration:

1. USER: Is there a TV in the room?
   SYS: No, but a radio.

[12] See [HOEP84a] for a detailed description of this and other HAM-ANS dialogs.

2. USER: Is there a desk in the room?
   SYS: No, but a table of a good height.

This behavior is accomplished as follows:

If the evaluation of a user question returns a negative result, the content of the question is matched against the user requirements. If this succeeds, an inference procedure is triggered which finds advantages of the desired facility. For instance, the height of a desk is marked as an advantage since it allows writing. With the new criterion "height=high", a transformed user question is produced and evaluated on the basis of the referential knowledge. Here, a high table is found as the next best facility and presented to the user. Note, however, that this kind of overanswering can be inadequate if the queried feature is not recorded in the user requirements. Look at the following example:

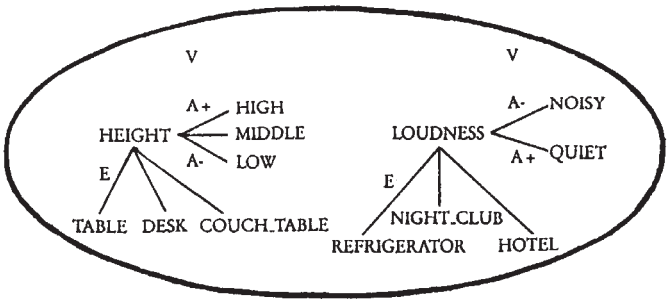
3. USER: Is there a night-club in the hotel?
   a) SYS: No, the hotel is quiet.
   b) SYS: No, but there is a night-club near the hotel.
   c) SYS: No.

Both answers, a) and b) sound reasonable. Which one is appropriate depends on the wants of the particular user. If a user asks whether p, s/he may want that p or that not p. It is not sufficient to take every question as a request for the feature in question! The *evaluation criteria* of the users – viewed as *frozen plans* – form a sound basis to decide between the two possibilities. If the user requires the hotel to have a night-club, the same procedure as for example 2) yields answer b). An advantage of a night-club is found (entertainment) and the referential knowledge is searched for next best entertainment facilities. The search succeeds because the environment of the hotel is represented as well in the referential knowledge about the hotel. If the queried feature does not match the user's evaluation standards, the system looks for a disadvantage of the feature in question. Given that night-clubs are loud, another value of the same dimension (loudness) can be found in the conceptual network. The inference procedure builds an expression with the new value of the same dimension, e.g. the hotel is quiet. This is matched against the requirements. If this succeeds, the system concludes that not the queried facility, but its opposite is desired instead. The system tries to confirm the new (opposite) property in the domain model. If this succeeds, answer a) can be generated.

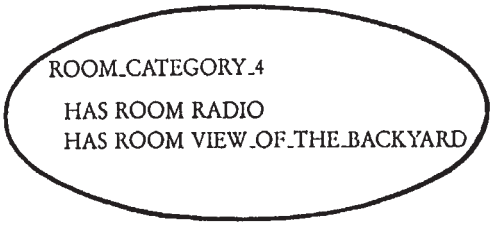
A prerequisite of this procedure is the conceptual knowledge that defines the dimensions with their possible values for concepts (see Figure 3). V-links represent the possible scalar attributes of a dimension. An example of a dimension is "color", the attributes being "red", "blue", "yellow" etc. E-links indicate which objects can have attributes of a particular dimension[13]. In the referential network, the link between a particular object and a particular value is given by the REF-link. Annotating scalar attributes with +/– allows for representing the extremes of a dimension (e.g. high(+)/low(–), quiet(+)/noisy(–)). Excerpts of knowledge sources are shown in

[13] Relativa are treated as scalar attributes, too, although linguistically this is, of course, not adequate.

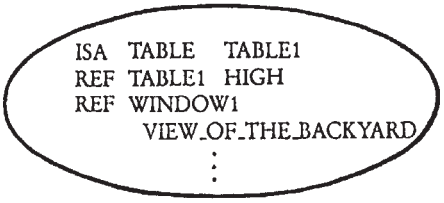
Conceptual knowledge



Referential knowledge about a room category



Referential knowledge about a room



Referential knowledge about the hotel



Requirements



Isotopy - list



Figure 3. Excerpts from some knowledge sources

Figure 3. The extremes allow for finding the opposite value instead of the negated value (which could cover more than one value). Note that +/− does not prejudice the user's assessment of the corresponding attributes. Rather, the possible advantages and disadvantages of a feature are to be represented in other knowledge sources (PLUS, MINUS) which are also used by the inferences.

4.3. Integrating the Acquisition Phase into the Dialog Using Isotopy

So far, evaluation standards have been used to relate the user questions to his/her wants, thus disambiguating the intention underlying the question. But the questions may also be used for adding new evaluation criteria. Most often particles in user utterances have been proposed for deriving user requirements [JAME80, KOBS85a]. But what if the user does not type in the particles? In typed man-machine interaction, particles, such as "hopefully", are rarely found. Another well-known way is the use of precondition-action-effect sequences. In a static domain, where the only modeled action is to book a room, however, this does not apply.

Another approach, which I want to propose here, is to use the textlinguistic notion of "isotopy" [GREI66]. *Isotopy* is the co-occurrence of semantic features. Isotopy is a way of achieving coherence, as Greimas states. That is, a user does not put questions one after the other without any semantic relation between them. The coherence is established by properties which are common to the facilities mentioned in a sequence of questions. If neither a task structure [GROS81] nor an intended action [ALLE79], [LITM85] are known to the system these common features still form a basis for determining the interests of the user. They can be inferred from the domain model and the content of the questions in the following way. The dimensions referred to in a question (the dimension is found in the conceptual knowledge) are pushed into the *isotopy list*. The intersection of the dimensions involved in the subsequent question and the already stored dimensions is then viewed as a good candidate for the coherence establishing feature. The intersection becomes the first element of the isotopy list. The next question again adds a dimension which is intersected with the preceding ones, and so forth. In coherent dialogs, several questions refer to the same dimension.

This type of coherence in user utterances can be used for determining user wants, i.e. updating the evaluation standards of the user as a result of the dialog. Let us look at an example of a user-driven acquisition.

4) USER: Has the room view of the street?
   SYS: No, view of the backyard.
5) USER: Fine.
   SYS: May I then book a room for you?
6) USER: One moment, I have another question.
   Is there a night-club in the hotel?
   SYS: No, the hotel is quiet.

There is a common feature of the view of the street and the night-club inside the hotel, namely that they are loud. Therefore, the corresponding dimension,

loudness, comes to the foreground. The foreground reflects the point of interest of the user. The advantageous value of the dimension "loudness" is "quiet". The value is determined by a consulting the PLUS knowledge source. Therefore, in our example, we can add to the requirements as a new criterion that everything be quiet. This excludes noisy air conditioning as well as noisy refrigerators or night clubs. Of course, this new requirement is also processed by the modification routine. Recall that requirements are used for selecting the appropriate answer out of the set of true, non-misleading alternatives (4.2). In our example here, the answer concerning this feature is a). Even if no preferences as to loudness were known before, this time for question 6) which is the same as question 3) above, answer a) can be chosen instead of 3b) or 3c) (see above)[14].

If, on the other hand, the user had not responded favorably to a suggested view of the backyard, the answer would then have been c). The MINUS of a view of the backyard is that it is ugly. Nothing is known about the dimension "beauty" of a nightclub. So there is no feature common to the view and the existence of a night-club available. In this case, the user might have switched from "beauty" or "brightness" to another underlying concern ("amusement" or "loudness"), which might become clear in the next question. In this situation the evaluation standards of the user cannot be enhanced by a new requirement, but this does not disturb the answering process. Rather, it enables overanswering the next time by augmenting the isotopy list. A diagram of the overall process is shown in Figure 4.

The concept of isotopy, which is operationalized here as a coherence establishing dimension to which several questions refer, is a very general one. Greimas [GREI66] introduced it in the course of analyzing jokes. There, although actually two dimensions (to speak in our terms) are in the intersection of several propositions, only one of them is recognized by the hearer – until the last sentence, which intersects with *the other* one only. It is yet an open question how to handle cases like this, where two dimensions are in the intersection and, moreover, the first choice has to be revised.

We have shown how the general concept of isotopy can be exploited for the user-driven acquisition of the user model and for the choice of the appropriate answer. This method is applicable even in domains which do not allow for precondition-action-effect sequences.

5. Conclusion

Conversational settings in which both system and user play a role in the domain of discourse and where more than the beliefs of the user are changed require that speech acts other than the information request/informing act are handled. For a variety of these speech acts, evaluation standards are presupposed in speech act descriptions of linguistic theory [MORI82]. Therefore, operationalizing evaluation standards forms a basis for modeling a wide range of speech acts. Evaluation standards may be used

[14] The answer to question 4) as the one to question 1) above here counts as a direct system answer; this is due to the general overanswering capability of HAM-ANS ([WAHL83]).

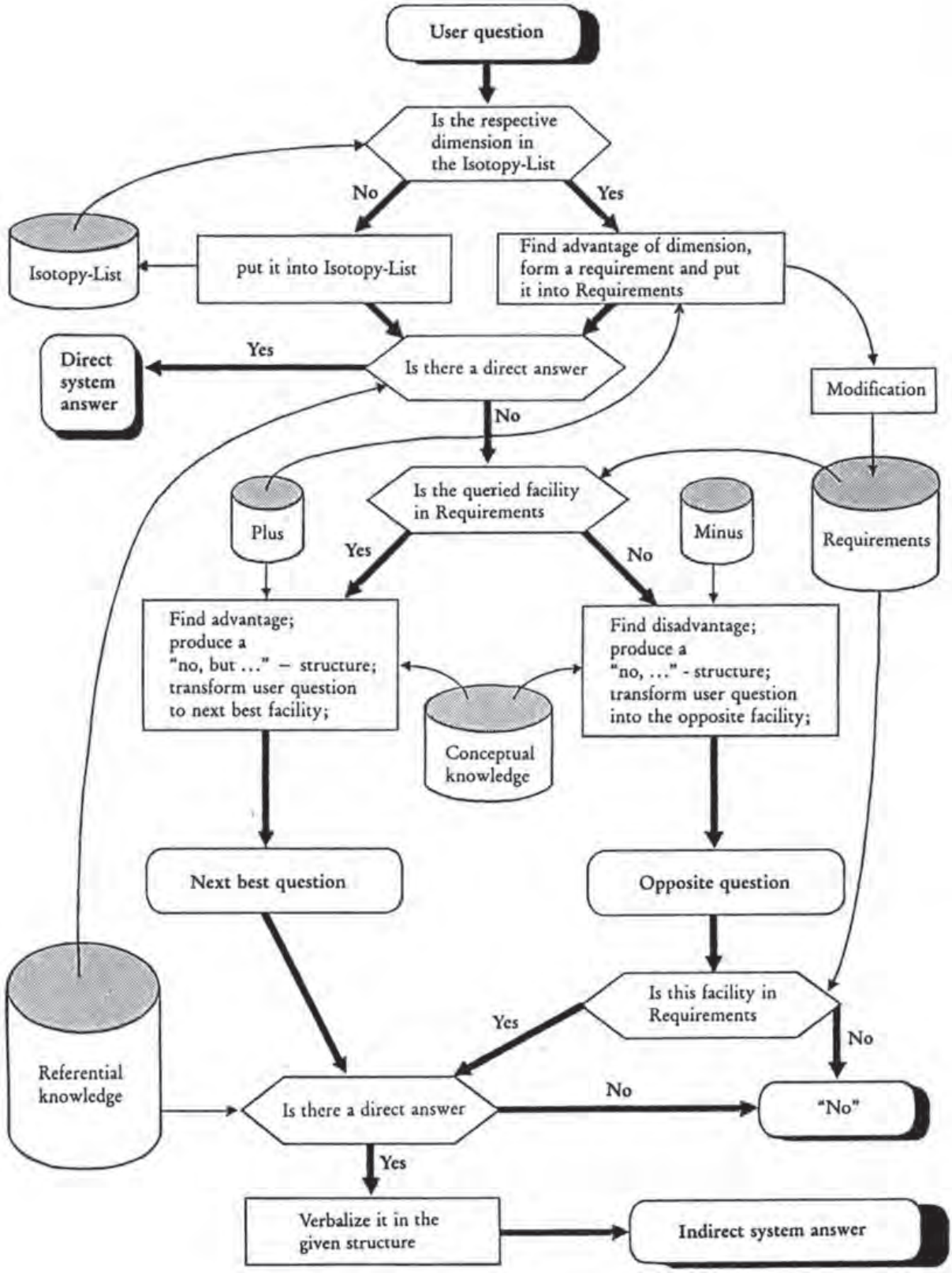


Figure 4. Overanswering using evaluation standards and isotopy

for speech act recognition *and* speech act generation which has been shown in this chapter for the example of recommendation speech acts.

Moreover, evaluation standards enable the system to decide whether a queried property or its opposite is required. This, in turn makes it possible to overanswer questions which otherwise would have to be just negated.

Evaluation standards vary from person to person. Therefore they are modeled as part of the user model: modeling the user's wants. A new way of user-driven acquisition of user-models has been proposed. Common properties of objects under discussion form a type of coherence, which has been called "isotopy" in textlinguistics. Isotopy can be used for determining the user's interests and choosing between alternative cooperative answers.

6. References

[ALLE79], [APPE82b], [BARB82], [CARB*], [CHIN*], [FINI83], [GREI66], [GREW78], [GROS81], [GROS86], [HARE52], [HOEP83b], [HOEP84a], [JAME80], [JAME82], [JAME83], [KOBS85a], [KONO84], [LEHM*], [LITM85], [MARB83], [MCCO*], [MORI82], [MORI84], [MORI85a], [MORI88], [NEUM83], [NOTH84], [POLL83], [RICH79a], [RICH*], [SCHG79], [SPAR84], [SPAR*], [WAHL83], [WAHL86a], [ZILL82]

14 Student Modeling in Intelligent Tutoring Systems — Implications for User Modeling[1]

Robert Kass

Abstract

User modeling and student modeling, though similar in many ways, have generally been pursued in parallel for the past decade. This chapter explores the role of student modeling in intelligent tutoring systems from the perspective of user modeling, with the goal of determining what researchers in user modeling can learn from student modeling. The chapter focuses on three issues: the information modeled about students, how that information is represented, and how the student model is built. Several intelligent tutoring systems that do student modeling are critically examined (in particular BUGGY, LMS, GUIDON, and WUSOR), and recommendations for user modeling based on this examination are made.

1. Introduction

For the past decade, user modeling in interactive systems and student modeling in intelligent tutoring systems have developed along similar, but seemingly separate, paths. Research in both user modeling and student modeling has been driven primarily by the needs of the respective systems that use the models. As a consequence, interaction between researchers in these two fields has been limited, although they seem to have many features in common.

This chapter is an attempt to bridge the gap between user modeling and student modeling. Here, student modeling is viewed from a user modeler's point of view, asking "What can user modeling research gain from student modeling?" In asking this question, three issues will be important:

1. What information about a student must be modeled?
2. How is the information represented?
3. How is the student model built?

These issues will be examined by studying several intelligent tutoring systems that do student modeling. The first four systems (BUGGY, LMS, GUIDON, and WUSOR) not only demonstrate the range of modeling approaches used in student

[1] This work was supported by a grant from the Digital Equipment Corporation.

modeling, but are considered to be "classics" in the field. Some later approaches are also examined to see the progress after, and the effects of, these classic systems.

This chapter is organized as follows. Section 2 gives a brief history of computer aided instruction and describes intelligent tutoring systems in general. Section 3 presents the general techniques used in student modeling for representing and building the student models. Sections 4 and 5 present several intelligent tutoring systems that do student modeling, discussing some of their contributions and limitations. Section 6 summarizes the features of student modeling with respect to the three issues mentioned above, and draws several conclusions about how user modeling can benefit from the work in student modeling.

2. Intelligent Tutoring Systems

Intelligent tutoring systems (ITS's) can best be characterized by differentiating them from other types of computer-aided instruction (CAI) systems.[2] The simplest (and oldest) type of CAI system is *generative.* Generative CAI systems were designed for situations requiring repetitive practice of a skill, such as vocabulary or arithmetic. From these systems evolved *adaptive* CAI, which makes use of information obtained from the student to guide the selection of problems, or the course of the tutoring sessions. Adaptive CAI systems might use simple techniques to estimate a student's skill, and select problems based on this estimate. More complex systems use a branching structure to direct the flow of the tutorial. The student's answer to a question will determine which branch to follow in selecting the next question. Branches are predetermined by the tutorial author, who must anticipate all possible mistakes and combinations of mistakes a student might make. Adaptive systems are better than simple generative systems because they can tailor problems and information to the specific needs of each student – provided the author of the tutoring material correctly anticipates student misconceptions. Neither generative nor adaptive CAI systems have knowledge of the domain content: the information is supplied from an outside source in a fixed format.

The goal in building intelligent tutoring systems is for the system itself to be aware of the knowledge it contains, as well as what the student knows. Such systems should be "... capable of automatically *inducing* and *using* a structural model of the student's reasoning strategies" [BROW75, p. 2]. An intelligent tutoring system is capable of being "self-adaptive," in that it can adapt the tutoring session on the basis of self-contained knowledge. Tutoring involves three forms of knowledge [SELF74]:

1. Knowledge of how to teach (including knowledge about students in general),
2. Knowledge of what is being taught (domain knowledge),
3. Knowledge of who is being taught (student model).

This knowledge must be accessible to a tutoring system for it to make decisions about what problems or information to give, and how best to introduce material to the student.

---

[2] This classification of CAI systems is taken from [SLEE82c].

An ITS consists of four components: an expert module, a student model, a tutoring component, and a communication module [WOOL84b].

Expert Module. The expert module contains the knowledge the system has about the domain of instruction. This is usually the knowledge that a human expert in this domain would have, or at least the knowledge of a very good student. The expert module can be either a "black box" or a "glass box." A black box expert provides correct answers to questions, but the inner workings of the expert are not accessible. A classic example of a black box expert is the expert in SOPHIE-I [BROW76, BROW82], which is capable of simulating an electronic power supply. Situations are posed to the expert, which can determine how the power supply would behave, but not provide justifications for its answers. In a glass box module the reasoning of the expert can be observed and explored. For example, the GUIDON system [CLAN82] uses the expert system MYCIN (later NEOMYCIN) as its expert module, and is capable of following the expert reasoning by keeping track of the rules used to make a diagnosis.

Student Model. The student model contains information about individual students using the program. The student model must be able to indicate the abilities of a student with respect to the domain being taught, as well as student preferences for particular methods of tutoring.

Tutoring Component. The tutoring component is responsible for determining *what* information to give to the student, *how* that information should be conveyed, and *when* it is appropriate to provide such information. A good tutoring system will have a variety of strategies it may employ to help a student. The tutoring component not only contains knowledge of these strategies, but must be able to determine the strategy most appropriate for a given situation. This is especially true when the student is encouraged to discover new information. In these situations a student will frequently make mistakes, but the mistakes may be useful in furthering the student's understanding of the subject. A tutor must be able to decide when to let the student flounder a little, and when to interrupt and guide the student.

Communication Module. The communication module handles the actual communication between student and tutor. On output, it must ensure that information provided by the tutoring component is effectively presented to the student. On input, it must interpret the student's responses in a manner intelligible to the system. Since the student's view of the domain may be significantly different from that of the expert module, this task of interpretation may be quite hard.

Intelligent tutoring systems may interact with the student in a variety of ways. For example, the system may drive the interaction by leading a student through a series of exercises (such as SPADE-0 [MILL82], which guides a student through writing a program in the LOGO language), or it may interact with the student only when he requests help. ACE [SLEE77, SLEE82d] observes students analyzing Nuclear Magnetic Resonance (NMR) spectra, but only gives advice when the student asks for it. Often the tutoring system is viewed as a coach, observing students working on a problem, and interrupting to give advice when it believes the interruption will benefit the student. A large number of systems use this approach, including the WUMPUS Advisor (WUSOR) [CARR77, GOLD82], GREATERP (a Lisp tutor) [FARR84, REIS85], an advisor for the VAX/VMS operating system [SHRA82], and

a tutoring system for users of the ReGIS graphics display [HEIN85]. An ITS may support a mixture of interaction types, such as GUIDON [CLAN82], which can act as a coach, an advisor, or take direct control of the interaction to question the student, depending on the situation and the form of tutoring it determines to be the most effective.

3. Student Modeling

Most intelligent tutoring systems model the knowledge of the student in relation to their own domain or "expert" knowledge. This expert model is illustrated first, then various methods for relating student models to expert models, and approaches to building student models, are analyzed.

3.1. The Expert Model

Any tutoring system must have correct domain knowledge in order to know when the student is correct. In intelligent tutoring systems this correct knowledge is often treated as a built-in "expert." The expert knowledge represents a paradigm for what a student should know. For example, in the electronic board game WEST [BROW75], a student must combine (using arithmetic operations) three digits obtained from spinners to determine how many spaces to move on the board. The board includes special squares (cities and shortcuts) which, if landed on, advance the player even further. On the other hand, the player may bump an opponent backwards by landing on a square occupied by the opponent. The WEST expert can tell, for any given spin, how to combine the numbers from the spinners in order to achieve the best possible move. Using this knowledge, the tutoring component of WEST can tell whenever a player makes a suboptimal move. Thus the WEST expert plays the best possible game that could be played.

Expert knowledge may be divided into independent units. SOPHIE-III [BROW82] contains both an electronics expert and a troubleshooting expert. The electronics expert knows about general laws of electronics (such as Ohm's or Kirchhoff's laws), about how particular electronic devices behave (such as resistors, transistors and diodes), and about particular electronic circuits, while the troubleshooting expert has knowledge about diagnosing circuit faults in general. In the NEOMYCIN knowledge base used by the GUIDON2 tutor, knowledge of the specific domain is separated from strategic knowledge of how to diagnose a disease.

Sometimes the expert model should not truly be an expert in the domain. Anderson [ANDJ85a] points out that in tutoring programming to novices, the problem solution of a programming expert is often too sophisticated for a novice to understand. Thus the basis of knowledge for teaching a novice should be knowledge that can be grasped by the student. In GREATERP [REIS85] the "expert" is actually an *ideal student*. The knowledge of the ideal student is what would be expected of a very good student of programming. This knowledge can be understood by the novice programmer, yet still stand as a paradigm of how to program.

3.2. Relation of the Student Model to the Expert Model

A tutor's goal is for the student to acquire the skills and knowledge necessary to successfully perform tasks in the domain of tutoring. The method adopted by intelligent tutoring systems for achieving this goal is to induce in the student a model of the domain matching the system's expert model. Since this expert knowledge must be available to the tutor anyway, the student model in an ITS is usually built in relation to the expert model held by the system. Three techniques are used to relate the student model to the expert model in a tutoring system: overlay modeling, differential modeling, and perturbation modeling.

3.2.1. Overlay Modeling

The overlay model, proposed by Carr and Goldstein [CARR77],[3] is the simplest of the modeling techniques. Overlay modeling assumes that all differences between the student's behavior and the behavior of the expert model can be explained by the lack of a skill or skills on the part of the student. Thus the knowledge of the student is viewed as a *subset* of the expert's knowledge. Since overlay models do not require information beyond the contents of the expert model, they are relatively easy to implement. An overlay model is illustrated in Figure 1.

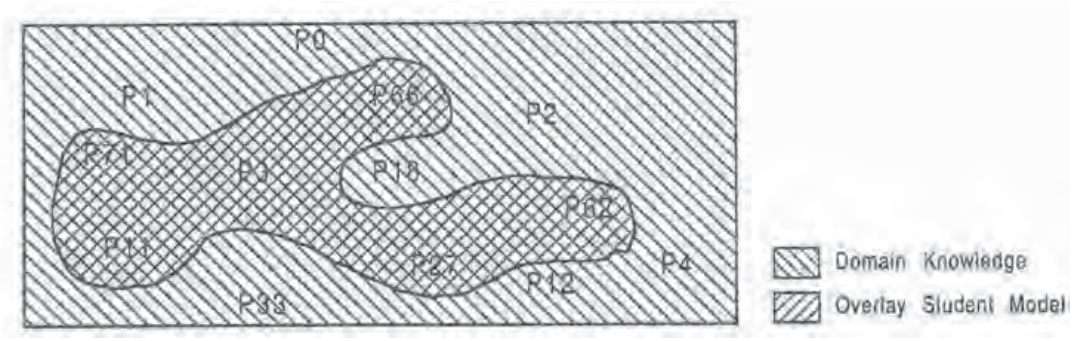


Figure 1. An overlay student model

The overlay technique was used by Carr and Goldstein to implement WUSOR-II, the advisor for the game "Hunt the Wumpus." This game is played in a warren of interconnected caves. A player must search through the warren to find and kill the deadly Wumpus. The Wumpus is killed by shooting an arrow into the cave occupied by the Wumpus. If a player walks into a cave containing a pit or the Wumpus, he loses. If he enters a cave containing bats, he will be carried off by the bats and dropped at random into another cave. To help the player, clues are given when he is in a cave adjoining a cave containing bats or a pit, or when he is within

[3] Although Carr and Goldstein coined the term "overlay model," the technique was used earlier by Carbonell in the SCHOLAR program [CARN70a].

two caves of the Wumpus. New players need to acquire a variety of skills to play the game well, such as: how to search the warren, how to deal with single and multiple evidence of danger, and how to apply probabilistic reasoning in making choices. The expert possesses all of these skills.

Overlay modeling works well in WUSOR-II, because (1) the goal of the tutoring system is to impart the knowledge of the expert, (2) the expert truly is an expert in this situation, and (3) the expert model is simple enough that, with time, even a novice can grasp it. The reasoning done by the expert will lead to playing the best game possible, hence the goal of WUSOR-II is to impart this knowledge to the student. Thus, any failure on the part of the student is attributed to the student's missing a skill which the expert has.

The difficulty in building an overlay model for a particular student depends on the type of information in the model. If the information is at the surface level, such as modeling the student's knowledge of program commands (as ANCHISES does for the EMACS text editor [ZISS85]), the student modeler can simply mark what the student knows as he uses the terms. On the other hand, if the expert model contain deep knowledge, such as strategies for diagnosis (as the NEOMYCIN expert model for the GUIDON2 tutor has [LOND82]), recognizing what knowledge the expert and student share can be much more difficult.

Overlay modeling has serious drawbacks. An overlay model has no provision for dealing with knowledge or beliefs of the student that differ from the expert's. If the student has an alternate strategy for Hunt the Wumpus that plays well, an overlay model will not recognize it, perhaps causing the tutor to try "correcting" the student. If the player simply has a wrong strategy the overlay modeler can detect it, but the tutor will not be able to point out why the strategy used by the student is wrong, because it has no knowledge other than that of the expert. Thus overlay models are ultimately incomplete.

3.2.2. Differential Modeling

The differential model is a modification to the overlay model. Instead of comparing student knowledge with expert knowledge, the differential model is produced by comparing the performance of the student with the expert's performance in the same situation. Differential modeling divides system knowledge into two classes: knowledge the student *should* know (if the student is to behave like the expert), and knowledge the student could not be expected to know. The knowledge a student is expected to know is just the knowledge the expert model has used to produce its behavior. Overlay modeling assumes that any knowledge in the expert model but not in the student model is not known by the student. Differential modeling subdivides this class into knowledge the student could not be expected to have, and knowledge the system believes the student does not have. A differential model is illustrated in Figure 2.

A differential model is used in the WEST game described earlier. As the student plays the game, the expert model also simulates playing the game from the viewpoint of the student. If the student, for example, consistently fails to make use of shortcuts the expert uses, the tutor can conclude that the student lacks knowledge about how to use shortcuts. On the other hand, if the student consistently fails to bump the

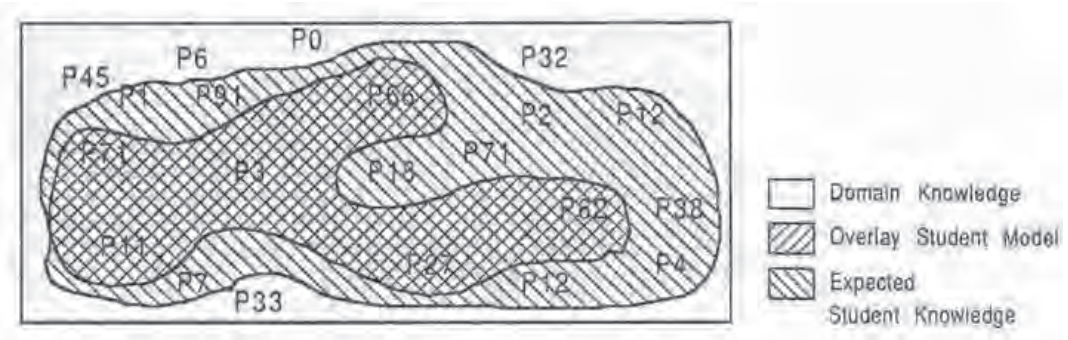


Figure 2. A different student model

opponent's piece, while the expert, given the same situation, also avoids bumping, then no conclusion can be drawn about whether the student has knowledge of how to bump opponents or not.

The differential model avoids the problem of being overly presumptuous about the knowledge of the student, but still suffers from most of the difficulties of the overlay model. Differential modeling still assumes that the knowledge of a student is only a subset of the knowledge of the expert. Such an assumption leads to a student model that is incomplete, since it cannot represent any belief of the student outside the domain knowledge of the expert.

3.2.3. Perturbation Modeling

Perturbation models can be used to represent student beliefs beyond the range of the expert model, while maintaining a close link between the student and expert models. Perturbation modeling builds a student model similar to the expert model, differing only on the basis of small perturbations to some of the knowledge in the expert model. Thus the student and expert models are basically the same, with small differences in some areas of the knowledge base. The perturbation model subsumes the overlay model in that a student's lack of knowledge is viewed as another form of perturbation. A perturbation model is illustrated in Figure 3.

Many existing intelligent tutoring systems use perturbation models. The common technique is to encode the expert knowledge, then augment that knowledge with likely misconceptions or "bugs" the student might have. For example, the DEBUGGY system [BROW78] uses a procedural network (which will be described shortly) to represent the correct methods for performing subtraction. Attached to this network is a set of procedures representing common mistakes made by students performing subtraction. These bugs are the perturbations to the expert model.

The Leeds Modeling System (LMS) [SLEE81] uses production rules to represent the correct methods of algebra, and *mal-rules* for the common misconceptions students have about how to solve algebraic equations. Mal-rules play the same role as the buggy procedures in DEBUGGY. Other systems make use of a *bug library* to represent the common misconceptions of students in a domain. Exam-

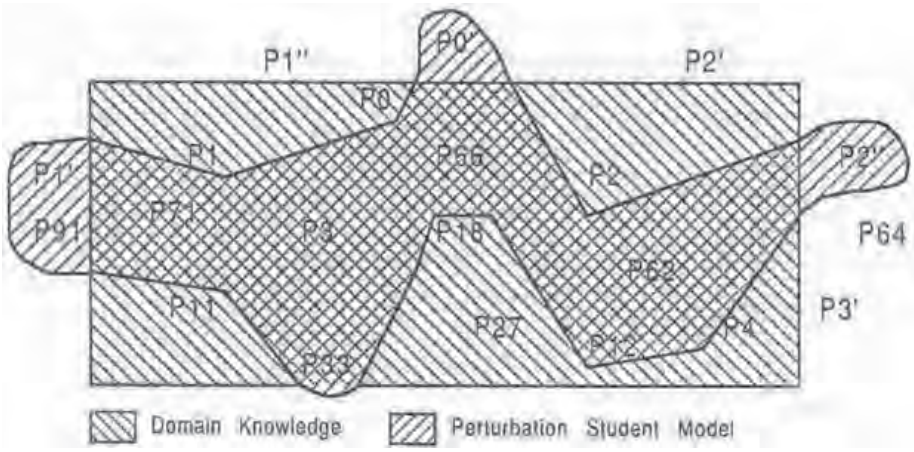


Figure 3. A perturbation student model

ples of such systems are PROUST [JOHN84], the Geometry Tutor [ANDJ85b], and GREATERP [REIS85].

Perturbation modeling complicates the process of building an individual student model, because it introduces more ways to explain the student's behavior. An error in the student's performance may be due to a missing skill, as with the overlay and perturbation models, or it may be due to the student's use of an incorrect skill. Perturbation modeling expands the space of possible student models greatly, requiring special techniques to reduce the number of models that must be searched. In fact, one system (LMS [SLEE83]) approaches the problem of modeling a student as one of controlling the size of this search space.

The perturbation model still assumes that a student and the expert are similar enough that a comparison can be made. This assumption has intuitive appeal. In human interaction, people expect other individuals to be basically like themselves. If no common ground can be found, communication usually falters. In a teaching environment, when the teacher recognizes that a student has no conception of what he is talking about, he will try to drop back and start over at a lower level until enough common understanding exists to work back to the original subject. Thus perturbation modeling seems to be a reasonable and intuitively attractive method for modeling students.

4. Four Intelligent Tutoring Systems That Model Students

In this section four intelligent tutoring systems that use student modeling are analyzed in some detail. These systems were chosen because together they represent the range of approaches to student modeling used in intelligent tutoring systems. Each of the systems is also a "classic" in the field, although the oldest was built less than ten years ago. The four systems are:

1. the BUGGY programs: BUGGY, DEBUGGY and IDEBUGGY
2. the Leeds Modeling System (LMS)
3. GUIDON, and GUIDON2
4. WUSOR (version III)

4.1. BUGGY

The BUGGY programs were designed to model the incorrect ("buggy") subtraction skills observed in the performance of children on tests.[4] Three programs were built during the course of the project. BUGGY, the most commonly mentioned program is actually a program to help arithmetic teachers recognize student bugs. BUGGY selects a bug, then asks the teacher to give it sample problems to solve using this incorrect procedure. A teacher must select problems that isolate the error, tell the program when they think they have found the bug, then prove that they have found it by simulating the bug on several problems generated by the system. The DEBUGGY program reads a student's test answers and proposes the bug or bugs that most likely caused the student's errors. IDEBUGGY is an interactive version of DEBUGGY, with the added capability of dynamically generating problems to help identify student bugs.

4.1.1. Description of BUGGY

The goal of the BUGGY programs is to build a *diagnostic model* of the student: a model of the internalized set of incorrect instructions or rules capable of duplicating a student's behavior. A successful diagnostic model can predict not only *which* problems a student should get wrong, but exactly *what* the wrong answer will be.

The diagnostic model is represented using a procedural network [SACE77] to simulate the behavior of a student. A procedural network consists of a *conceptual* part, used by BUGGY to represent the subskills used in subtraction, and an *operational* part, consisting of methods (both correct and buggy) for performing the action associated with the conceptual part. In the BUGGY programs, this network formed a lattice called the *skill lattice*, representing all possible ways a student could attempt to solve subtraction problems. When a student's problems are presented to DEBUGGY or IDEBUGGY, various combinations of good and buggy methods are tried to find a complete procedure that most closely matches the performance of the student. A set of heuristics reduces the number of models to be tried by using features of the domain, such as dependencies or incompatibilities between the subskills. Burton [BURT82] reports that eventually the DEBUGGY system recognized 110 primitive bugs, and was able to directly recognize 20 common multiple bugs, plus any combination of up to four primitive bugs.

The student modeling requirements for the BUGGY systems are not great. Typically only a small set of problems is available for use in determining the buggy procedure used by a student. Acquiring a specific model for the student consists of

[4] The work discussed here is reported in [BROW77, BROW78, BURT82]. Of these, [BROW78] is the most commonly referenced work and contains the majority of the results of the work. However, [BURT82] is a good summary of the entire project.

searching for a path through the skill lattice that best matches the performance of the student. Because the procedural network includes buggy methods in addition to the correct methods, the BUGGY systems use a perturbation model for modeling the student.

4.1.2. Contributions of BUGGY

Several features of the BUGGY systems are important and interesting contributions to the use of student models in an ITS, and for intelligent tutoring systems in general. Perhaps the greatest contribution is the overwhelming empirical evidence that students do make systematic errors in doing multi-column subtraction. Before this work, many teachers believed that students who performed poorly made random errors due to lack of intelligence or lack of interest. Brown and Burton proved that even many of the worst students were very consistent in applying a procedure to solve problems. Their poor performance was due to using an incorrect procedure. This observation is important for the future of student modeling, since it implies that student errors are due to identifiable misconceptions. Furthermore, student behavior is consistent, hence those misconceptions can be recognized and modeled.

The DEBUGGY system also introduced a method for dealing with noise in student test data. Even though students were consistent in their buggy procedures, they still made occasional mistakes due to copying or number fact errors. DEBUGGY has a set of *coercions*, reflecting typical mistakes of this sort, which it can apply to a problem. These coercions help to reduce the problem that random noise can bring to the modeling process.

The BUGGY systems also provide a detailed model of the subtraction procedure. One result is the realization that even a simple procedure such as subtraction can be quite complex, and there are many ways that a student can get it wrong. The skill lattice for representing the possible correct and buggy procedures of students had the side-effect of being useful in determining how effectively a test measures each of the component subtraction skills.

4.1.3. Criticisms of BUGGY

Several criticisms can be leveled at the BUGGY systems as well. The bugs listed by Brown and Burton tend to be superficial. Many of the bugs seem to be different surface manifestations of a few core bugs. The model can be used to reproduce student mistakes, but not to explain *why* the student has made those mistakes. Thus BUGGY does not provide a deep model of student subtraction procedures.

The expert knowledge of the system, although encoding the procedural knowledge necessary to do subtraction, is a black box – it cannot be used to justify or explain the reasoning of the expert or the student.

All knowledge for this system had to be hand-coded. People had to pore over student tests and try to determine what bug or combination of bugs could account for the answers given. When bugs were discovered they were added to the system. There is no ability to augment the knowledge of the system directly from the interaction.

4.2. LMS

The Leeds Modeling System (LMS) [SLEE81, SLEE82a, SLEE83, SLEE84] was designed to be a vehicle for exploring student modeling capabilities. LMS approaches student modeling as a search problem, with two goals oriented towards making the search computationally feasible: (1) contain the search required in creating a student model, and (2) have the modeler behave in a data directed manner.

4.2.1. Description of LMS

LMS uses production rules to represent the correct skills necessary for solving simple algebraic equations. Augmenting these correct rules are additional *mal-rules* representing possible misconceptions the student might have. Thus LMS uses a perturbation model for representing the knowledge of students. Building a student model consists of selecting from the set of rules and mal-rules an ordered list of production rules capable of simulating the performance of the student on algebra problems.

Modeling in LMS occurs in two phases. The first phase occurs offline, when LMS generates a complete set of possible student models from the correct rules and mal-rules, retaining those that give unique results to predefined problem sets. The second phase occurs online, using the models generated by the offline phase to determine whether the student's behavior can be explained by any of the models generated. The online phase proceeds in stages, with each stage designed to augment a growing model of the student. At each stage the student must solve a set of problems designed to test a particular algebraic skill. For example, at one stage the student may be given a set of problems that require him to move a number from the left hand side of the equals sign to the right by adding the additive inverse of the number to both sides of the equation. A mal-rule for this operation might move the number from left to right, without changing the sign of the number.

The power of LMS is that it uses a *systematic* search to build the model of the student. Unlike DEBUGGY, which used heuristics to limit the number of models that had to be searched, LMS systematically searches the space of possible models, using an incremental method to build the student model that effectively limits the size of the search space. At each stage in the online phase of modeling, the potential models are limited to models generated from the existing model by adding either a correct or buggy rule for the skill being tested, or perhaps a mal-rule corresponding to an earlier skill.[5]

LMS was used by a small number of students who were later interviewed by educational psychologists who had not seen their performance with LMS [SLEE82a]. The object of this experiment was to compare the student models generated by LMS with those inferred by the interviewers. This experiment demonstrated that LMS still lacked several mal-rules. When those rules were added the models developed were similar.

[5] This last capability was added when it was discovered that even though a student may perform a skill correctly once, later they may use an incorrect method to perform that skill.

Further studies by Sleeman reported in [SLEE84] indicate that the simple production rule model is insufficient. Analysis of further student's algebraic mistakes, and more interviews, suggests that there are four classes of student errors in algebra.

1. *Manipulative errors*: where a variant of the correct rule is used, with one stage omitted or replaced by another.
2. *Parsing errors*: students frequently misunderstood the notation of the problem.
3. *Clerical errors*: errors due to miscopying, etc.
4. *Random errors*: errors for which no explanation can be given.

Only the manipulative class of errors is explained by the original LMS model. Subsequently, additional mal-rules were added that account for most manipulative and parsing errors.

Matz [MATZ82] has suggested that many parsing errors are a result of students making incorrect extrapolations from the domain of arithmetic to that of algebra. Sleeman calls this process *misgeneralization* and suggests it is another source of common student errors. Misgeneralization is a mechanism that can explain not only the behavior of a student, but also the underlying reasons for the behavior. This psychological knowledge is much more useful to a tutoring system, for the system is now capable of directly addressing the root of the student's problem.

4.2.2. Contributions of LMS

LMS makes three major contributions to the study of student modeling. First, it is possible to constrain the search space in constructing a model such that the search technique is computationally feasible.

Second, a data-driven modeler is possible. In particular, the model building machinery of LMS is relatively domain independent. A production system could be written for other situations in which a procedural skill is taught and LMS would deal with them equally well. On the other hand, this technique might not work well with domains containing a lot of factual knowledge, or in domains that are not well defined.

Finally, student errors may be of different classes, representing fundamental differences in the underlying cause for the error. Developing a psychological theory for student errors will be an important step to truly effective modeling.

4.2.3. Criticisms of LMS

Partly because LMS focuses on a particular aspect of the modeling problem, there are many criticisms which can be raised about the system.

The models produced are not psychological models of student learning. Hence, even though a suitable model can be built for simple algebra, there is no theoretical basis for claiming that LMS should work in other domains. Furthermore, although the model for algebra can explain the behavior of a student, it gives no indications as to why students have this particular set of mal-rules, or why students in general make some types of mistakes but not others. Sleeman recognizes this problem in [SLEE84], but provides no solution in LMS.

LMS suffers from the possibility of non-monotonicity in the model. As the model is built incrementally, it is possible that a later rule may subsume or supercede a rule already in the model. Such an interaction might cause the model to no longer explain student performance previously observed. A solution to this seems to be more extensive testing of the model when a new rule is added, to ensure that previous explanatory ability has not been lost.

A very practical problem with LMS is the amount of hand-coded information which must be explicitly provided by the system builder. Not only must the designer encode the basic rules and associated mal-rules of the domain, but also the many problems used to distinguish between various rules during the stages of online model building. This requires the builder to be able to anticipate a good deal of the student's behavior in a manner similar to designers of branching adaptive CAI programs.

4.3. GUIDON

The GUIDON [CLAN81, CLAN82], GUIDON2 [LOND82], and later systems [CLAN86a] developed as an attempt to use MYCIN-like knowledge bases as a basis for tutoring medical diagnosis. The overall system has been considered fairly successful, but only after some significant changes to the representation of the MYCIN knowledge itself.

4.3.1. Description of GUIDON

GUIDON plays the role of a *case method* tutor. Students who use GUIDON are given an initial description of the symptoms of a patient suffering from some strain of meningitis (the domain of tutoring). This is the *case*. The student then proceeds to ask questions in an attempt to develop a hypothesis about the particular form of disease the patient has. GUIDON observes this dialog and may interrupt to ask questions in order to get the student to clarify his/her strategy, to give advice, or even redirect the dialog. Thus GUIDON is basically a coaching type of tutor.

GUIDON2, the successor to GUIDON, uses a differential approach to modeling the student. The expert model is used to generate a reasonable *range of predictions*, the set of actions and hypotheses that GUIDON2 expects the student *should* have because they are what the expert *would* have [LOND82]. Usually this range of predictions will be sufficient to model the student's behavior. If not, GUIDON2 starts a data-driven search to find a new range which will.

An important discovery of the GUIDON project is that a MYCIN-like knowledge base is not sufficient to provide a good basis for tutoring. Because of the explanation facilities available in MYCIN, it seems reasonable to expect that this rule based knowledge would be a good foundation for intelligent tutoring systems as well. The work with GUIDON proved these expectations to be incorrect, for two reasons. First, MYCIN rules are very shallow, skipping many of the intermediate reasoning steps that experts make. Secondly, MYCIN rules are not homogeneous. The rules in MYCIN mix several types of disparate knowledge, often in the conditions of the same rule. These problems with MYCIN led to the development of a new knowledge base representation, called NEOMYCIN [CLAN83].

NEOMYCIN is designed to be a *psychological model* of how a medical expert diagnoses diseases. In NEOMYCIN, domain knowledge is classified into several categories, and is separate from the knowledge pertaining to diagnosis in general. Protocols of experts diagnosing patient diseases were used to create a set of *meta-rules* that control the overall behavior of the NEOMYCIN expert. Analysis of these protocols indicate the expert has an overall goal of determining what information to seek next. The expert thinks in terms of a hierarchy of causes and process characteristics of a disease, and uses this information to order the data and search process in developing a diagnosis. This knowledge is independent of the domain and should be applicable to other situations where GUIDON is used as a tutor.

The domain knowledge of NEOMYCIN is grouped into several sections. An etiological taxonomy is present reflecting the causal relationships between various concepts relevant to the domain. This taxonomy contains many of the intermediate concepts an expert uses when reasoning from symptoms to an actual disease. The domain knowledge also includes world facts important for the diagnosis. The domain rules of the system are classified into one of four categories:

1. *Causal* rules linking observed symptoms with concepts in the etiological taxonomy.
2. *Trigger rules* that associate data with the etiologies. These data correspond to working hypotheses the expert has.
3. *Data/hypothesis* rules that associate circumstantial data and laboratory data with diseases in general.
4. *Screening* rules used by the expert to control the search.

GUIDON also includes knowledge specific to managing the student model. Since a differential model is used, the knowledge of the student is considered to be a subset of the knowledge of the domain expert. The modeler (called IMAGE in GUIDON2) contains a set of rules to update the model after each step in the dialog. These rules use the same form of evidential reasoning as MYCIN, so the modeler can include degrees of belief for elements in the student model.

4.3.2. Contributions of GUIDON

GUIDON has made many contributions in the area of student modeling. A primary contribution has already been mentioned: the discovery that a MYCIN-like knowledge base was insufficient to support intelligent tutoring. In particular this means that a rule-based knowledge representation by itself is not sufficient to ensure a good expert model for tutoring. Typical production rule systems tend to be very shallow in their knowledge. Such knowledge may be effective for an expert system, but does not form a psychological model of how people structure knowledge about the domain.

A second contribution of GUIDON is the presentation of a psychological model of how medical experts perform diagnosis. This model, NEOMYCIN, demonstrates the need to separate several disparate types of knowledge, not only between the domain and strategy components, but also in the domain itself. It also demonstrates the need to structure factual knowledge to complement the rule-based knowledge.

Thus student modeling will require a very sophisticated and diverse knowledge representation – no single, simple representation will do.

Finally, the use of evidential reasoning to build the student model gives GUIDON a tolerance for noisy data. Since noise is an important problem in modeling a student, the explicit evidential reasoning used by GUIDON seems appropriate.

4.3.3. Criticisms of GUIDON

Although GUIDON satisfies many of the criticisms of the previous systems, there are still some inadequacies. First, since GUIDON uses a differential model with no bug library, the student model is limited to a subset of the expert model. This assumption is too strong, even for medical school students with a good background in the domain.

GUIDON assumes its students are completely familiar with the concepts, factual data and laboratory tests referenced during the tutorial. Thus the tutoring is limited to the diagnostic process itself, and not factual concepts in the domain of diagnosis.

The system cannot evolve new notions of possible student models. Thus GUIDON could not "discover" that a student was using a rule which was unknown by GUIDON and remember this rule for later modeling. A truly good system with this capacity could recognize when a student used a new, valid rule the system was not familiar with. Such a rule could then be added to the expert's knowledge base, allowing the expert to learn as well [KIMB82, OSHE82].

Finally, GUIDON has no sense of the learning limitations of the student. After going through a long session a human tutor would expect the student to be tired and less capable of assimilating new knowledge. GUIDON does not have a model for student learning and forgetting, or for the performance of a student under different cognitive loads.

4.4. WUSOR

WUSOR is an advisor for the logical game "Hunt the Wumpus." A short outline of this game was given earlier. Three versions of WUSOR have been built. The first used no modeling techniques and the second [CARR77] used a simple overlay model. The third version, WUSOR-III is more sophisticated than the previous versions, is the most interesting with respect to student modeling, and hence will be the focus of this section.

4.4.1. Description of WUSOR-III

The modeling in WUSOR-III [GOLD82] focuses on how knowledge is acquired by a student. Knowledge of how a student learns information in a domain enables accurate prediction of what items an individual will learn next. This greatly aids model building, since it serves as a strong constraint on the search space of possible models. It also aids the intelligent tutor, by giving a clear indication of topics it would be appropriate to provide help on, and topics the student is likely to misunderstand at a given point in the session.

The WUSOR-III coach consists of five distinct modules. These include the expert and student models and the tutoring component itself. In addition, a module called the *psychologist* is responsible for updating the student model. The fifth module, the *genetic graph*, is unique to WUSOR. The genetic graph is a data structure containing the domain knowledge used by the expert. On top of this domain knowledge (represented as rules), links that relate the rules in various ways are added. The graph is "genetic" in that it represents the relationships between concepts from an evolutionary viewpoint: concepts are linked via the methods by which they may be learned.

The genetic graph contains four types of links:

1. *Generalization*: R' is a generalization of R if R' is obtained from R by quantifying over some constant. *Specialization* is the inverse of generalization.
2. *Analogy*: R' is analogous to R if there exists a mapping from the constants of R' to the constants of R.
3. *Refinement*: R' is a refinement of R if R' manipulates a subset of the data manipulated by R. *Simplification* is the inverse of refinement.
4. *Deviation*: R' is a deviation of R if R' has the same purpose as R, but fails to fulfill that purpose in some circumstances. *Correction* is the inverse of deviation.

These links represent the way individuals learn new concepts from concepts they already know. Thus the genetic graph is proposed as a psychological model of how students acquire knowledge of the domain.

The structure of the genetic graph also provides information about student learning. The connecting links between domain rules naturally represent the interrelationships between concepts in the domain. Usually there will be areas of the graph with a large number of interconnections. These groups are called *islands*. Islands represent small bodies of closely related material that tend to be learned in units. Islands thus provide a hierarchical structure to the knowledge of the domain, indicating areas of knowledge which a student will learn, and the underlying finer concepts.

WUSOR-III models the student as a pair of specialists, one for problem solving and the other for learning. The student model is constructed as an overlay of the genetic graph. The knowledge of the problem solving specialist is simply an overlay of the domain knowledge, which constitutes the nodes in the graph. The learning specialist consists of a set of strategies for learning new material. The types of links between nodes in the genetic graph reflect the particular learning strategies that can be used to learn domain information in the graph. The student model also overlays these links in the graph, indicating the learning strategies used by the student in acquiring new information about the domain.

The genetic graph represents the evolution of a student's knowledge. Initially, the student model will be an overlay of only a few nodes in the genetic graph. As the student acquires new knowledge, the student model will grow along the paths linking previously acquired knowledge with new knowledge. Thus at any time in the learning process a "frontier" exists in the genetic graph, reflecting the current working knowledge of the student. Nodes far to the interior of the graph will tend to be superceded by those on the frontier. The active model of the student at any given time is thus the frontier of the student model in the genetic graph.

4.4.2. Contributions of WUSOR

WUSOR's main contribution to student modeling is the notion of the genetic graph. As with GUIDON, the genetic graph represents a psychological model of the student, but the genetic graph mechanism focuses not only on student knowledge, but student *learning* as well. Thus the genetic graph provides a framework for representing how a student acquires knowledge and skills in a domain.

A further advantage of the genetic graph is its ability to predict what knowledge the student is likely to acquire next. The frontier of the graph serves as a means for limiting the search necessary when attempting to recognize the knowledge of a student. By the same technique, the frontier aids in selecting tutoring material, since it provides a set of items most likely to be relevant and understandable to the student at a given point in the session.

The structure of the genetic graph can be used to better understand the complexity of the domain. Domains with strongly connected genetic graphs should be easier to learn than those having relatively few connections. Thus islands in the graph will tend to be learned as units, while shifting from one island to another may be very difficult. This shift in complexity may explain why students tend to have difficulties of particular sorts, or with particular material.

A final, potential advantage of the genetic graph structure is that it may be used upon itself. The domain of the genetic graph could be the learning strategies themselves. Thus a system could tutor not only an underlying domain, but in addition the learning skills used to learn about the domain.

4.4.3. Criticisms of WUSOR

Unfortunately, WUSOR stills suffers from many problems. First, the genetic graph is based on an overlay model. The incompleteness of such an approach seriously limits the modeling of abilities of WUSOR.

The knowledge representation of the underlying domain is unstructured. WUSOR uses simple production rules for representing its knowledge. As Clancey has pointed out, deep structure in the domain knowledge is needed, as well as multiple knowledge representation techniques. Extending the genetic graph for such an underlying domain may not be possible.

WUSOR suffers from the noise problem. Although the frontier of the student model in the genetic graph should represent the current knowledge of the student, this knowledge is often uncertain. Students acquiring new concepts may alternately use them correctly, then incorrectly. Furthermore, students frequently do not apply what they learn immediately. Thus knowledge on the frontier of the student model may very well be known by the student, but the lack of use will lead the psychologist module to doubt that knowledge.

As with the other systems discussed in this section, all knowledge must be predefined by the system authors. The WUSOR system is not capable of recognizing any student behavior beyond the knowledge it has stored in the genetic graph. For WUSOR, this means the author must not only encode the domain knowledge, but also all possible links that connect this knowledge. Such a task would prove very daunting in practice.

The genetic graph is rather ad hoc. Different types of links can be created to meet various needs. There is no theoretical basis for the types of links in the graph, the justification of their presence is their intuitive appeal.

5. Further Work in Student Modeling

One point made very clear from the work reviewed in the last section is that deep psychological models of students are needed to provide effective tutoring. This means that the model must go beyond simply simulating student behavior: it must be able to explain the underlying causes for that behavior. Since the work in the four "classic" systems described in the previous section, several other researchers have begun to address the problem of building deep student models. In this section three recent approaches are presented: the work of Johnson and Soloway on PROUST, the work of Anderson *et. al.* on GREATERP and the Geometry Tutor, and Brown and VanLehn's Repair Theory.

5.1. PROUST

PROUST [JOHN84, JOHN85] was developed to identify non-syntactic bugs in novice student's programs. To do this, it must cope with the high degree of variability that occurs in programming, yet correctly identify where a student goes wrong in writing a program. Accurate tutoring in this situation requires that the tutor not only identify what has gone wrong, but also determine what the *intentions* of the student were in writing the program in this way. Plan recognition by itself is not sufficient, because the plans may be incorrect implementations of correct plans, or correct implementations of plans to achieve an incorrect goal. Thus PROUST not only models the plans students use in writing programs, but also their intentions in using those plans.

PROUST uses a *goal structure* and a knowledge base of relations between goals and plans to identify program bugs. Problem descriptions are given to PROUST, including a specification of the problem in a special goal-oriented language. This problem specification is then combined with the knowledge base of information about goal and plan interactions to determine the possible methods for solving the problem in terms of subgoals and subplans. When a student program is analyzed, PROUST selects a subgoal from the problem description and attempts to find program statements in the student program that satisfy it. When a match between the subgoal and the student's code cannot be found, PROUST attempts to find a match using a *bug library* of typical errors. It will also attempt to restructure the subgoals to find a better match. PROUST continues to select subgoals and tries to match them with the code until all subgoals have been tried. All possible interpretations are generated. At the end, PROUST will select the interpretation that most completely explains the student's program.

The bug library of PROUST has been classified using a device called a Goal and Plan (GAP) tree [SPOH85]. A GAP tree is a means of categorizing potential

bugs in a classification that reflects the way they tend to appear. Buggy plans are represented as partial GAP trees, providing substantial information about the bug in a problem independent manner.

Although Johnson and Soloway do not claim PROUST uses a psychological model of how students program, it does go beyond simply modeling surface behavior of the student. PROUST is capable of identifying not only the plan used by a student in writing a program, but also the student's intention in using that plan. Although PROUST could not be used to explain *why* the student used an erroneous plan, it does identify the student's misconception that a particular plan can be used to accomplish a particular goal. Thus PROUST provides useful information for remediation.

5.2. GREATERP and the Geometry Tutor

Anderson *et. al.* have implemented intelligent tutoring systems in two domains, using the same basic underlying system. GREATERP is a tutor for novice programmers in Lisp, while the Geometry Tutor is designed to help students learning to write high school geometry proofs. Both tutors are based on the ACT* theory of cognition.

The ACT* theory has four main features:

1. A production rule system, where each rule represents a unit of skill.
2. The production rules are goal directed, a rule only fires if a goal matches.
3. A working memory stores all the current knowledge, but this working memory is limited in size.
4. A learner becomes more skilled by acquiring new productions. This process is called *knowledge compilation.*

The student model is implemented in a GRAPES, a Goal-Restricted Production System that implements the features of ACT*. Both tutoring systems have an expert called the *ideal student* whose domain knowledge is encoded in GRAPES, as well as a bug catalog of common mistakes or poor strategies used by the student. Together, the ideal student and bug catalog form the *ideal and buggy model* (IBR).

The model of the student is built by a technique called *model tracing.* Essentially, the tutor traces the student's behavior through its ideal and buggy model. At any point many possible production rules may be candidates for application by the tutor. The tutor infers which rule is appropriate by comparing the student's actions with the results of applying each of the rules. If the rule selected is a correct rule (a rule from the ideal student model), the session continues. If the rule is a buggy rule, the tutor will interrupt to correct the situation. This tutoring method is based on empirical evidence that the best time to correct a student misconception is when it occurs. As a consequence of this philosophy, the student model is never more than one step away from the correct model. This small divergence enables not only accurate modeling, but also more effective tutoring.

GREATERP and the Geometry Tutor both require a great deal of predefined knowledge for the ideal student model and bug catalog. Given this, the systems

can perform very well, since they restrict the range of possible student models at any point. Thus GREATERP and the Geometry Tutor can be viewed as extending Sleeman's work with LMS, by including deep knowledge at the level of units of human cognition.

5.3. Repair Theory

Repair theory was proposed by Brown and VanLehn [BROW80] to provide a deeper model for the arithmetic bugs demonstrated by students. In particular, repair theory is able to generate bugs that students demonstrate, while avoiding bugs that have not been observed, or that are non-intuitive.

The fundamental basis for repair theory is the proposition that when a student reaches an *impasse* in performing a procedure, s/he will attempt to *repair* the procedure and continue as normal. An impasse occurs when the problem the student is working calls for a subprocedure the student does not know, or when the student has been performing an incorrect procedure that leads to a "dead end." At this point the student attempts to "repair" the procedure by performing one or more reasonable operations in an attempt to return the form of the problem to one the student knows how to solve.

The impasse-repair sequence is modeled in four parts:

1. A restricted production system for representing the components of a skill and their buggy alternatives. Rules in the system are limited to a single action and goal clause. The rules are interpreted on a stack. Each rule is only tried once and specialized rules are tried before more general rules. Otherwise rules are executed in order. The rules of the system are compiled into a Generalized AND/OR (GAO) graph.
2. A set of principles are used for deleting fragments of the correct procedure. This is used to simulate the impasse reached by the student when solving a problem requiring a skill s/he is missing. Since unconstrained deletion is too general, some restrictions on deleting are necessary. For example, if two rules have the same goal and one is a special case of the other, the more general rule cannot be deleted without first deleting the more specific rule.
3. A set of repair heuristics are used to propose repairs when an impasse is reached. These heuristics range from simply quitting, or skipping the problem area, to shifting the focus, or trying analogous rules to overcome the impasse.
4. A set of critics to filter out some repairs. The set of repair heuristics will tend to generate repairs that are either non-intuitive or are not observed in actual students. Such bugs are called *star bugs*. The purpose of the critic is to filter out all repairs that result in star bugs. Some critic heuristics include rules such as "don't change a number once it has been written" and "don't leave a blank in the middle of the answer."

The repair heuristics and the critics mentioned above are specific to the arithmetic domain. Thus the theory as stated is not a general theory. Brown and VanLehn feel it is possible to produce a body of weak heuristics that can be tailored to the particular domain of a system.

The proposals of repair theory result in several intuitively appealing features:

- Repairs are independent of impasses. This distinction separates the cause of a problem from the method used for its resolution.
- The problem solver is local. When an impasse arises, repair theory attempts to fix the problem on the basis of the current state of affairs.
- Due to the use of the stack, the problem solver can only call existing subgoals or primitive actions. This limits the actions which can occur during a repair, providing a tighter theory.

6. Summary: Comparing User Modeling and Student Modeling

User models and student models are both instances of the more general class of *agent* models [KOBS*]. A user model is simply the model a system keeps of the individual currently using it, while a student model is the model of an individual an ITS views as a student. When a student directly interacts with an ITS, the model of the individual is both a student model and a user model, implying that user models and student models are in general quite similar.

User models and student models do have distinct characteristics, however, at least in the way they are generally viewed. This distinction is rooted in how the models are used, resulting in different modeling approaches. This section will explore the different purposes of modeling students and users, and examine how these purposes affect modeling techniques. A major conclusion of this study is that user models will eventually need to adopt many of the capabilities sought in student modeling, thus student modeling research is important to user modelers.

6.1. Approaches to Modeling

Work in student modeling and user modeling has been driven by differing system needs. An ITS presumes that the student either lacks knowledge of the material being taught, or has misconceptions about that material. Furthermore, the student's knowledge is expected to *change* as the tutoring session proceeds. Consequently student modeling has had to consider how the student represents and reasons about the knowledge he has. Originally, student modeling sought models capable of duplicating student behavior – diagnostic models. A diagnostic model can roughly be thought of as answering the question "How did the student perform this task?" Diagnostic modeling was soon found inadequate, because the models could not guarantee that the student actually used the skills the model attributed to him, only that the model's performance was similar to the student's performance. Thus diagnostic modeling cannot adequately explain how a student performed a task, and cannot even begin to explain *why* the student performed the task in a particular manner.

The need to explain why students behave as they do requires psychologically-based student models. A principle underlying tutoring systems is that the tutor can

be most effective if it knows the fundamental reasons for a student's behavior, so it can tailor the tutoring to correct misconceptions, and instill correct knowledge and skills. Effective modeling of student behavior at this level requires an understanding of how the students represent and acquire information.

On the other hand, the demands on user models have not been so great. User models are generally needed to aid in the interaction between the system and a user: to help in understanding what the user wants the system to do, or to enable the system to tailor its responses to the particular user and his needs. Thus user modeling has concentrated on modeling the user's goals, plans, and beliefs. In such interactions, the user's knowledge usually is assumed to remain static. The emphasis in user modeling is primarily on *what* the user believes or wants, and sometimes on *how* the user plans to accomplish his goal; rather than on trying to model why a user holds certain beliefs, or how the user's beliefs might change.

The different expectations of student and user models is the main reason for the differences in their modeling approaches. The need to explain why students behave as they do has forced student modelers to consider how students organize and reason with their knowledge, resulting in deep, psychological models of students. Such models have not been necessary for typical user modeling applications. For example, a cooperative data base system needs to know what a user is looking for, and what he knows already, in order to respond with the appropriate information in the best form; but it generally is not expected to know why the user wants the information.

This gap between expectations of student and user models is not permanent, however. As the types of systems that employ user models develop, the expectations of the user model will change. For example, an advisory system must reason about user beliefs, and why the user holds those beliefs. To recommend a course of action for the user, the system must convince the user of its recommendation. If the user holds beliefs that seem to contradict the system's recommendation, the system may have to reason about why the user holds a belief, in order to demonstrate that the belief is incorrect, or perhaps to demonstrate that the recommendation really does not contradict the user's belief. Thus an advisory system needs deep knowledge of the relationships between the goals, plans and beliefs held by the user.

Deeper models of the user will also be required to support more robust interactions with the user, such as natural language dialog. In reasoning about what to say, a conversational participant must think about the hearer's knowledge, and about the conclusions a hearer might draw from a statement. For a system to participate in robust interactions, it will need the ability to simulate the user's reasoning. One example of this is the anticipatory feedback process described by Wahlster and Kobsa [WAHL*]. To support such reasoning the user model must have a deep knowledge of the the reasoning processes of humans. Ultimately, this knowledge could form a psychological model for human communication and reasoning.

6.2. Contributions of Student Modeling to User Modeling

Both the similarities between user and student models, and the anticipation that user models will develop in ways similar to student models suggest that user mod-

eling can profit from the progress made in student modeling. This section discusses five areas where student modeling work can contribute to user modeling: the relationship between student models and expert models, the advantage of controlled interaction, the need for deep modeling, the need for generative models, and the noise problem.

6.2.1. Relationship Between Student and Expert Models

Student models are always built with respect to a domain, or expert, model. This is a powerful aid both for representing the potential assumptions the system makes about students, and in building an individual student model.

In an ITS, building the student model with respect to the expert model is eminently practical, since an expert model must be present to represent the "ideal" student. When overlay or differential modeling is used, this expert knowledge is a sufficient basis for student modeling, while a perturbation model only requires that the expert model be augmented with the expected differences that a student might exhibit.

Likewise, user models can benefit from the presence of a domain model in the system. Such a domain model is likely to exist in the circumstances where a user model would be used. User models are used to facilitate communication with the user, both to understand the user and to produce responses. The communication component of such a system must have domain knowledge, as well as knowledge of the user to accomplish these tasks. Thus, when a user model is used, a domain model should be present as well, so the user model can be built in relation to a domain model.

The expert model also aids student modeling when building a model of an individual student. Since the expert model stands as a paradigm for the model of a student, the search for an appropriate student model can focus on models similar to the expert model. Thus the expert model creates expectations of what the student model should be. This technique is useful in user modeling as well. If a system has the correct representation of the knowledge in a domain, it can use this knowledge as a default expectation for the user's knowledge.

Techniques such as overlay, differential, or perturbation modeling are not limited to student modeling domains. These approaches are appropriate for user modeling when a body of domain knowledge is the main focus of interaction. Student modeling has also developed ways for inferring individual models based on the student's behavior and the domain model. Such techniques should be applicable to user modeling as well. In fact, subsequent to his work on LMS, Sleeman used a similar approach to build a model of the student in UMFE, a user modeling front end system used in conjunction with an explanation facility for an expert system [SLEE85].

6.2.2. Advantages of Controlled Interaction

Student modeling in intelligent tutoring systems benefits from the controlled interaction with the student. Typically, an ITS takes responsibility for the interaction with the student, by driving the interaction with questions or problems that the student must respond to. This controlled interaction aids student modeling by limiting

the range of possible student models that must be considered. The best examples of how a controlled interaction enhances student modeling are the incremental technique for building a student model in LMS, and the model tracing technique used in GREATERP and the Geometry Tutor. The controlled interaction limits the possible responses a student can make, thus limiting the number of possibilities that must be considered at any stage in building a student model.

User modeling is not so lucky. Systems that use a user model cover a range of interaction types, but usually the user controls the interaction, or the user and system share control. This means that a user may violate the expectations of the system by changing the subject, volunteering additional information, or perhaps addressing a related subject. Despite these problems, approaches such as model tracing are powerful, and may be useful for some user modeling situations. In practice, most interactive systems have a limited range of capabilities. If the system can identify what the user wants from the system, this will often constrain the interaction to particular topics and forms. At this point, a model tracing technique could be used to effectively build the user model.

6.2.3. Need for Deep Modeling

The need for deep user models has already been argued in Section 6.1. Building deep user models will not be an isolated activity, however. Systems that require a deep user model require a domain model that is equally deep. Thus highly structured domain models will be available to build user models in relation to, continuing the close relation noted above.

6.2.4. Need for Generative Models

As deeper models of users or students are built, the potential for differences between the user or student's representation of knowledge and the domain model's representation become greater. Thus, more complex perturbation models will be required. On the other hand, these differences may be explained by a small number of fundamental differences between user and system. One contribution of the work on repair theory was the notion that a small number of principles accounted for the variety of bugs in student arithmetic. Repair theory formed a *generative* theory of how such bugs were produced. With repair theory, instead of searching a large collection of potential models for one that most accurately matches a student's behavior, a principled theory of the process can predict when errors are likely, and what mechanism caused the error.

For user modeling, a generative theory would focus on characteristics of human reasoning. For example, such a theory might include rules for how people make generalizations or analogies, or how people think about actions. User beliefs, goals, and plans could be inferred by applying these rules to information already believed about the user, expanding the user model greatly. Such rules could also be used to explain the likely ways in which a user's beliefs, goals, or plans differ from those in the domain model. A first step in this direction is the work on inferring user beliefs based on the interaction, the domain model, and the existing user model using a set of implicit acquisition rules [KASS87a, KASS87b].

A generative theory is not easy to develop, however. Even in a simple domain like arithmetic the results were not completely satisfactory. Nevertheless, the underlying motivation is valid. Ultimately, a generative theory for user modeling must be founded on psychological principles of human behavior.

6.2.5. The Noise Problem

Another contribution of student modeling to user modeling is the recognition of the noise problem. For the most part, the problem of inconsistency on the part of the user has been ignored in user modeling. Although a good method for handling noise has not been achieved in student modeling either, it seems to have received more attention there. Once again, a proper treatment of user inconsistencies must wait for deeper psychological explanations of human memory, communication, and so on. Student modeling seems to recognize the problem, and that a principled approach to handling it is possible. User modeling can certainly benefit from such an approach as well.

7. Conclusion

This chapter has explored the role of student modeling in intelligent tutoring systems, with the objective of discovering what user modeling researchers might learn from the related domain of student modeling. After analyzing several intelligent tutoring systems that do student modeling, it appears that a fundamental difference between user and student modeling approaches has been due to the depth of information modeled. Student modeling has been forced to build deeper, psychological models of students to support the needs of a tutoring system. The progress in student modeling is important to user modeling because the demands on user models can be expected to follow a similar path. As interactive systems attempt to support more robust communication, reasoning about the user will require deeper knowledge of the user's beliefs and reasoning abilities. Thus the lessons learned in student modeling are appropriate lessons for user modeling as well.

8. References

[ANDJ85a], [ANDJ85b], [BROW75], [BROW76b], [BROW77], [BROW78], [BROW80], [BROW82], [BURT82], [CARN70a], [CARR77], [CLAN81], [CLAN82], [CLAN83], [CLAN86a], [FARR84], [GOLD82], [HEIN85], [JOḢN84], [JOHN85], [KASS87a], [KASS87b], [KIMB82], [KOBS*], [LOND82], [MATZ82], [MILL82], [OSHE82], [REIS85], [SACE77], [SELF74], [SHRA82], [SLEE77], [SLEE81], [SLEE82a], [SLEE82c], [SLEE82d], [SLEE83], [SLEE84], [SLEE85], [SPOH85], [WAHL*], [WOOL84b], [ZISS85]

15 GUMS — A General User Modeling Shell

Timothy W. Finin

Abstract

This chapter discusses the application of various kinds of default reasoning in system maintained models of users. In particular, we describe the general architecture of a domain independent system for building and maintaining *long term models* of *individual users*. The user modeling system is intended to provide a well defined set of services for an *application system* interacting with various users, and must build and maintain models of them. As the application system interacts with a user, it can acquire knowledge of him, and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling shell* (hereafter called *GUMS*) that we have implemented in Prolog. This system possesses some of the desirable characteristics we discuss.

1. Introduction — The Need for User Modeling

Systems that attempt to interact with people in an intelligent and cooperative manner need to know many things about the individuals with whom they are interacting. Such knowledge can be of several different varieties and can be represented and used in a number of different ways. Taken collectively, the information that a system has about its users is typically referred to as its *user model.* This is so even when it is distributed throughout many components of the system.

Examples that we have been involved with include systems that attempt to provide help and advice [FINI83, HOWE84, SHRA82], tutorial systems [SCHU86b], and natural language interfaces [WEBB84]. Each of these systems had to represent information about individual users. Most of the information was acquired incrementally through direct observation and/or interaction. These systems also had to infer additional facts about their users based on the directly acquired information. For example, the WIZARD help system [FINI83, SHRA82] had to represent which VMS operating system objects (e.g. commands, command qualifiers, concepts, etc.) a user was familiar with, and had to infer which other objects he was likely to be familiar with.

This chapter describes the general architecture for a domain independent system for building and maintaining *long term models* of *individual users.* The user modeling system is intended to provide a well defined set of services for an *application system* that interacts with various users, and needs to build and maintain models

of them. As the application system interacts with a user, it can acquire knowledge about him and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling shell* (hereafter called *GUMS*) implemented in Prolog. This system possesses some of the desirable characteristics we discuss.

The next section will discuss some of the general issues that arise in building user models. In particular, we will describe the range of possible issues that one might address in terms of four aspects: who is being modeled; what information is represented in the model; how the model is acquired and maintained, and how the model might be used. The third section provides an overview of the *GUMS* system and the kind of user modeling it supports. The fourth section describes the three kinds of default reasoning that *GUMS* employs: stereotypical reasoning, explicit default rules and failure as negation. Section five describes some of the details of the *GUMS* system, while the concluding section discusses some of the limitations of *GUMS* and avenues for future research.

2. What Kind of User Model?

The concept of incorporating user models into interactive systems has become common, but what has been meant by a user model has varied and is not always clear. In this section we will discuss, in general terms, what might be meant by the term *user model.* We will then characterize the approach to user modeling described in this work by answering the following questions: who is being modeled; what aspects of the user are being modeled; how is the model to be initially acquired and maintained; and why the model is needed.

The term "user model" has been used in many different contexts to describe knowledge that is used to support an interactive system. A survey of user modeling in support of natural language interfaces can be found in [KASS88b] and a discussion of user modeling for intelligent tutoring systems is provided by [KASS*]. In this section, we will look at a number of distinctions that will allow us to focus on an interesting and important class of "user models".

An initial definition for "user model" might be the following:

A user model is that knowledge about the user, either explicitly or implicitly encoded, which is used by the system to improve the interaction.

The definition is too weak, since it endows every interactive system with some kind of user model, usually of the implicit variety. In this chapter, we will focus our attention on those models where the information about the user is *explicitly* encoded. In particular, we are interested in user models that are designed along the lines of "knowledge bases". By this we have in mind the kinds of distinctions that are usually drawn to distinguish a *knowledge base* from a *data base.* In the context of user models, five features are important:

1. *Separate Knowledge Base* – Information about a user is collected in a separate module rather then distributed throughout the system.

2. *Explicit Representation* – The knowledge in the user model is encoded in a sufficiently expressive representation language. Such a representation language will typically provide a set of inferential services. That allows the knowledge of a user to be implicit, but automatically inferred when needed.
3. *Declarative rather than Procedural* – The knowledge in the user model is, as much as possible, encoded in a declarative rather than procedural manner.
4. *Support for Abstraction* – The modeling system provides ways to describe abstract as well as concrete entities. For example, the system might be able to discuss classes of users and their general properties as well as individuals.
5. *Multiple Use* – Since the user model is explicitly represented as a separate module, it can be used in several different ways (e.g. to support a dialog or to classify a new user). This requires that the knowledge be represented in a more general way that does not favor one use at the expense of another. It is highly desirable to express the knowledge in a way that allows it to be reasoned about as well as reasoned with. Similarly, the model ought to provide no barriers to its use by several different applications.

User models having these features fit nicely into current work in the broader field of *knowledge representation.* In fact, we could paraphrase Brian Smith's *knowledge representation hypothesis* [SMIB82] to get something like:

Any user model will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge the system has of the user and b) independent of such external semantical attribution, play a formal but causal and essential role in the behavior that manifests that knowledge.

In the remainder of this section we will explore the large space of user modeling possibilities by addressing the *who, what, how* and *why* of user modeling. Our own system will then be placed within this space.

2.1. Who Is Being Modeled?

Two aspects of who is being modeled are the degree of specialization and temporal extent. The degree of specialization involves whether we are modeling classes of users or individuals. Temporal extent refers to the persistence of the knowledge encoded in the user model – does it expire at the end of the current session or does it form a long-term knowledge base about the user.

A user model can lie somewhere on a specialization scale that ranges from *generic* to *individual.* A generic user model assumes a homogeneous set of users – all individuals using the system are similar enough with respect to the application that they can be treated as the same type of user. Many natural language systems, for example, maintain a single generic user model in order to infer the user's goals, or to recognize a user's misconceptions [ALLE82b, CARB83, CARN83c, GERS81]. A somewhat more individuating point on this scale is to employ a set of fixed, generic models to represent different subclasses of users. For example, the UC

system [WILE86, CHIN86, CHIN*] classifies a user as belonging to one of four possible generic classes: *novice, beginner, intermediate, or expert.*

Modeling the very general beliefs held by large classes of users is extremely important when building intelligent systems. For example, in modeling a person's beliefs about a timeshared computer system we may want to include a rule like:

If a user U believes that machine M is running, then U will believe that it is possible for him to log onto M.

This sort of rule is required to support the kinds of cooperative interactions studied in [JOSH84a, JOSH84b], such as the following:

User: Is linc.cis.upenn.edu up?
System: Yes, but you can't log on now.
Preventative maintenance is being done until 11:00am.

Individual user models, on the other hand, contain information specific to a single user. A user modeling system that keeps individual models thus will have a separate model for each user of the system. A simple example of this kind of model is the commonly used convention of customizing the behavior of a utility such as an editor or mail system by allowing each user to have a *profile* file that states his preferences for various system parameters [RICH83].

A natural way to combine the system's knowledge about classes of users with its knowledge of individuals is through the use of *stereotypes* – generic models with specializations for individuals. A stereotype is a cluster of default facts about the user that tend to be related to each other. One can model an individual as belonging to a particular stereotype (e.g. *novice Unix User*) with a set of overriding facts (e.g. *knows how to use pipes*). Examples of systems that have used stereotypes include GRUNDY [RICH79b], and the Real-Estate Advisor [MORI85b].

2.2. What Is to Be Modeled?

The *contents* of a user model naturally vary from application to application. Following Kass and Finin [KASS*], we will classify this knowledge into four categories: goals and plans, capabilities, attitudes, and knowledge or belief.

A user's goals and plans are intimately related to one another. The user's *goal* is a state of affairs he wishes to achieve, while a *plan* is a sequence of actions or events that he expects to result in the realization of a goal. Furthermore, each step in a plan has its own subgoal to achieve, which may be realized by yet another subplan. One of the hallmarks of an intelligent, cooperative system is that it attempts to help the user achieve his underlying goals, even when not explicitly stated. The principle way for one agent to know another's underlying goal(s) is by recognizing his observable actions as constituting a (possibly partial) plan for achieving a domain-relevant goal. A great deal of work has been done on the problem of *plan recognition* in support of cooperative interactions. Examples in the natural language domain include Allen and Perrault [ALLE80] and Carberry [CARB83, CARB88, CARB*]. Intelligent Tutoring Systems have introduced the idea of keeping a catalog of "buggy" plans [BROW78, SLEE82a, JOHN84] as a means of recognizing mis-

2.3. How Is the Model to Be Acquired and Maintained?

The acquisition and maintenance of the user model are closely related topics. By acquisition, we mean the techniques for learning new facts about the user. Maintenance involves incorporating this new information into the existing model and squaring away any discrepancies or contradictions. We will briefly review some of the possibilities for these two related problems.

The knowledge in a user model can be acquired either *explicitly* or *implicitly*. Explicitly acquired knowledge might come directly from the system designer in the form of built in knowledge of stereotypes, from the application system the user model is serving, or from the user himself. GRUNDY, for example, begins with a new user by interviewing him for certain information. From this initial data, GRUNDY makes judgments about which stereotypes most accurately fit the user, thus forming an opinion about the preferences of the user based on this initial list of attributes. As another example, each time the UMFE system [SLEE85] needs to know a new fact about its user, it simply asks.

Acquiring knowledge about the user implicitly is usually more difficult. Implicit acquisition involves "eavesdropping" on the user-system interaction in order to observe the user's behavior and from it to infer facts that go into the model. Various aspects of implicit acquisition have been explored, including the use of presuppositions [KAPS82], misconception recognition [MCCO85a, MCCO*], stereotype selection [RICH79b, RICH*, MORI85b], and the use of default rules [PERR88, KASS87a, KASS87b].

Maintenance involves incorporating new knowledge about an individual user into an existing model. The new, incoming information might be consistent or inconsistent with the current model. If it is consistent, maintenance involves triggering inferences to add additional facts to the model. If the new knowledge is inconsistent with the current model, then the inconsistency must be resolved in some manner. The possibilities depend on the underlying representation and reasoning systems on which the modeling system is built. Two major approaches are *evidential reasoning*, which allows the model to hold that a certain fact is true with a certain degree of belief, and *default reasoning* [REIT80], in which certain facts can be held in the absence of evidence to the contrary.

2.4. Why Is the Model There?

The knowledge about a user that a model provides can be used in a number of ways in an intelligent, interactive system. The particular opportunities depend, of course, on the application. Figure 1 presents a general taxonomy of possible uses. At the top level, user models can be used to support (1) the task of recognizing and interpreting the information seeking behavior of a user, (2) providing the user with help and advice, (3) eliciting information from the user and (4) providing information to him. Situations where user models are used for many of these purposes can be seen in the examples presented throughout this book.

The model can be accessed in two primary ways: facts can be added, deleted or updated from the model, and facts can be looked up or inferred. A forward chaining

conceptions or missing knowledge. The use of plan recognition is also relevant to the problem of providing help and advice, as in [FINI83, WILE86].

A second category of knowledge to be modeled has to do with the user's capabilities. Although some systems might have to reason about a user's physical capabilities, such as the ability to perform some action that the system may recommend, it is more typical for a system to model and reason about mental capabilities, such as the ability to understand a recommendation or explanation provided by the system. Examples of the latter often arise when a system is generating an *explanation*, as in Wallis and Shortliffe [WALL82] and Paris TAILOR system [PARI88]. Webber and Finin [WEBB84] have surveyed ways that an interactive system might reason about its user's capabilities to improve the interaction.

A third category of knowledge represents a user's bias, preferences and attitudes. GRUNDY [RICH79b] was an early user modeling system that used a model of a user's preferences to recommended books to read. The Real-Estate Advisor [MORI85b] and HAM-ANS [HOEP83b, MORI*] are other systems which have tried to model this. Swartout [SWAR83] and McKeown [MCKE85a] address the effects of the user's *perspective* or *point of view* on the explanations generated by a system.

The final category of information we will consider is *knowledge* and *belief*.[1] Modeling the user's knowledge involves a variety of things: domain knowledge, general world knowledge, and knowledge about other agents. Knowing what the user believes to be true about the application domain is especially useful for many types of systems. In generating responses, knowledge of the concepts and terms the user understands or is familiar with allows the system to produce responses incorporating those concepts and terms, while avoiding concepts the user might not understand. This is particularly true for intelligent help systems [FINI83, WILE86], which must provide clear, understandable explanations to be truly helpful. Providing definitions of data base items (such as the TEXT system does [MCKE85c]) has a similar requirement to express the definition at a level of detail and in terms the user understands.

A final form of user knowledge that is important for interactive systems is knowledge about other agents. As an interaction with a user progresses, not only will the system be building a model of the beliefs, goals, capabilities and attitudes of the user, the user will also be building a model of the system. Sidner and Israel [SIDN81b] make the point that when individuals communicate, the speaker will have an *intended meaning*, consisting of both a *propositional attitude* and the *propositional content* of utterance. The speaker expects the hearer to recognize the intended meaning, even though it is not explicitly stated. Thus a system must reason about what model the user has of the system when making an utterance, because this will affect what the system can conclude about what the user intends the system to understand by the user's statement. Kobsa [KOBS84] has studied some of the difficult representational problems involved with building recursive models of the beliefs of other agents.

---

[1] In the context of modeling other individuals, an agent does not have access to objective truth and hence cannot really distinguish whether a proposition is *known* or simply *believed to be true*. Thus the terms *knowledge* and *belief* will be used interchangeably.

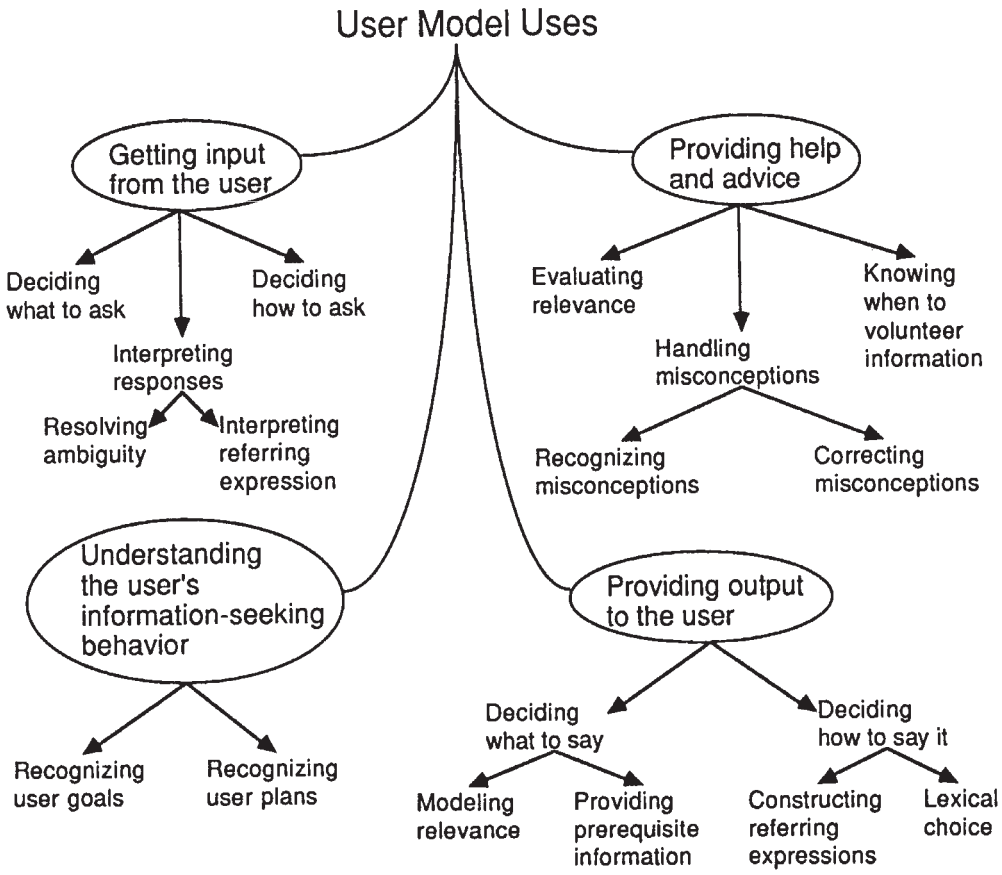


Figure 1. Uses for knowledge of the user

component, together with a truth maintenance system, can be used to update the default assumptions and keep the model consistent.

3. A General User Modeling System

The previous section described a large conceptual space of user modeling possibilities in terms of four dimensions. This section will place our own work in this space and give an overview of its major characteristics. Our paradigm for user modeling might be described as *User Model as Deductive Database*. The *GUMS* system allows one to construct models of individuals that appear to be a collection of facts that can be asserted, retracted or queried. Some of these facts are deduced through the use of inference rules associated with stereotypes to which the user belongs.

Who is being modeled? The *GUMS* system is designed for building **long term** models of **individual users**. We want to represent the knowledge and beliefs of individuals, and do so in a way that results in a persistent record which can grow and change as necessary. These users are associated with one or more *stereotypes* from which additional facts derive.

What is being modeled? Our interest is in providing a domain-independent user modeling tool. The contents of the user model will vary greatly from appli-

cation to application and is left for the model builder to decide. For example, a natural language system may need to know what language terms a user is likely to be familiar with [WEBB84], a CAI system for second language learning may need to model a user's knowledge of grammatical rules [SCHU86b], an intelligent database query system may want to model which fields of a data base relation a user is interested in [MOTR86], and an expert system may need to model a user's domain goals [POLL85]. However, our paradigm of user model as deductive database makes modeling certain things much easier than others. It is easy to construct models that are expressed as a set of relations. In particular, we do not provide any general facilities for building deep models of an agent's reasoning strategies or procedural knowledge.

How is the model acquired and maintained? Our approach in *GUMS* is to let the application be responsible for acquisition and to provide a set of services for maintenance. The application must select the initial stereotypes for the user and add new facts about the user as it learns them. How the application discovers these facts (i.e. explicitly or implicitly) is up to the application. Maintaining the model involves incorporating these new facts, checking that they are consistent with previously learned or inferred facts, and resolving any discrepancies and contradictions. This can be accomplished by undoing contradictory default assumptions and/or shifting the user to a new stereotype. The details of how this is done will be described later.

Why is the model there? Given our goal of producing a general purpose, domain independent use modeling facility, the uses to which the knowledge in the model is put are outside of our design. We will, however, give a more syntactic account of how the model is to be used in terms of how the application interacts with *GUMS*.

3.1. The System Organization

Our goal is to provide a general user modeling utility organized along the lines shown in Figure 2. The user modeling system provides a service to an application program that interacts directly with a user. Note that the user modeling system does not have access to the interaction between the application and the user. Everything it learns about the user must come directly from the application. The *GUMS* system has a separate knowledge base for each application it serves. Each knowledge base consists of two parts: (1) a collection of *stereotypes* organized into a taxonomy, and (2) a collection of models for the individuals. The individual models are installed in the stereotype hierarchy as leaves. Figure 3 shows such a hierarchy. As will be seen later, a stereotype is a collection of facts and rules that are applicable for any person who is seen as belonging to that stereotype. These facts and rules can be either *definite* or *default*, allowing for an individual to vary from the stereotypic norm.

The application program gathers information about the user through its interaction, choosing to store some of this information in the user model. Thus, one service the user modeling facility provides is accepting (and storing) new informa-

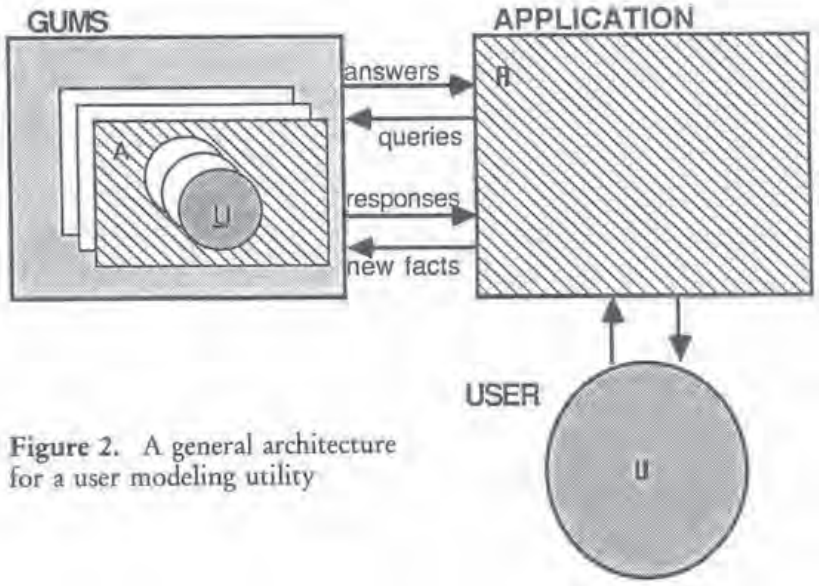


Figure 2. A general architecture for a user modeling utility

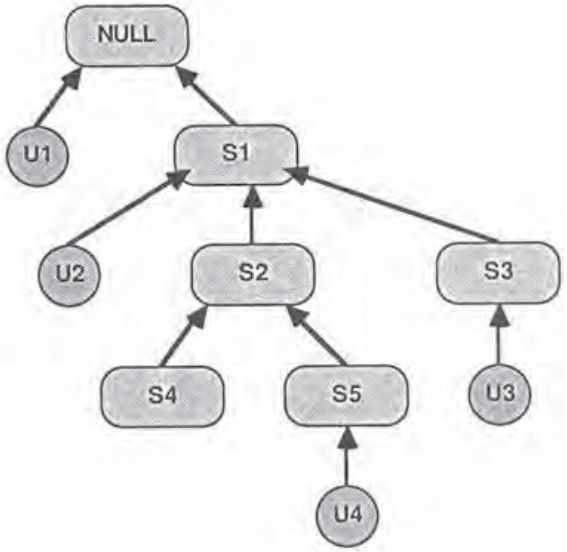


Figure 3. A hierarchy of stereotypes and individuals

tion about the user. This information may trigger an inferential process which could have a number of outcomes:

- The user modeling system may detect an inconsistency and so inform the application
- The user model may infer a new fact about the user which triggers a demon causing some action (e.g. informing the application).
- The user model may need to update some previously inferred default information about the user.

Another kind of service the user model must provide is answering queries posed by the application. The application may need to look up or deduce certain information about its current user.

3.2. Default Reasoning in GUMS

User modeling is most useful in situations in which a system must draw a number of plausible conclusions about a user from a small base of definite knowledge about him. Thus, default reasoning of some kind or another is at the heart of most user modeling systems. Our approach in the *GUMS* system is to use several forms of default reasoning techniques: stereotypes, explicit default rules, and failure as negation. These three types of default reasoning are closely related and, in general, formalizable in terms of a single type of default reasoning. We have found it convenient, however, to separate the default reasoning in *GUMS* for several reasons.

In the *GUMS* system the different kinds of default reasoning are used to capture generalizations of different *grain size* and form a *hierarchy* with respect to the strength of their conclusions. Stereotypes are used to capture generalizations of large classes of users. Within a stereotype, explicit default rules are used to express the stereotypic norms which are allowed to vary across individuals. Failure as negation is used as a general technique for gathering weak evidence for beliefs about a user when no stronger evidence exists.

3.2.1. Stereotypes and Individuals

In *GUMS*, a stereotype consists of a set of facts and rules that are believed to apply to a class of users. Thus a stereotype gives us a form of default reasoning in which each rule and fact in the stereotype is taken to hold in the absence of evidence that the user does not belong in the stereotype.

Stereotypes can be organized in hierarchies in which one stereotype subsumes another if it can be thought to be more general. A stereotype S_1 is said to be more general than a stereotype S_2 if everything which is true about S_1 is necessarily true about S_2. Looking at this from another vantage point, a stereotype inherits all the facts and rules from every stereotype that it is subsumed by. For example, in the context of a *programmer's apprentice* application, we might have stereotypes corresponding to different classes of programmer, as is suggested by the hierarchy in Figure 4.

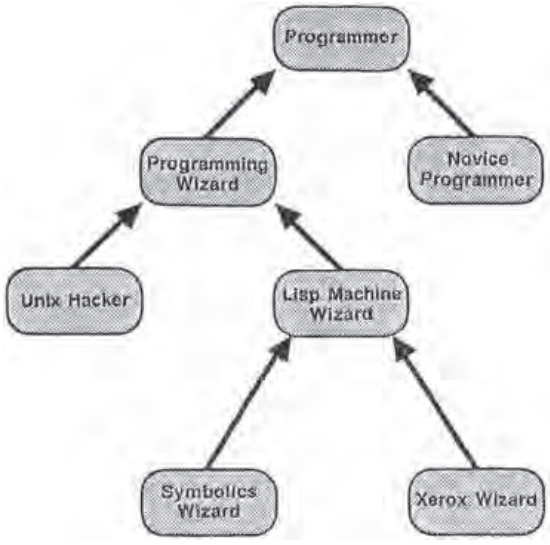


Figure 4. A hierarchy of stereotypes

In general, we will want a stereotype to have any number of immediate ancestors, allowing us to compose a new stereotype out of several existing ones. In the context of a *programmers apprentice*, for example, we may wish to describe a particular user as a *SymbolicsWizard* and a *UnixNovice* and a *ScribeUser*. Thus, the stereotype system should form a general lattice. Our current system, however, constrains the system to a tree.

A stereotype is comprised of several kinds of knowledge, as depicted in Figure 5. Each stereotype can have a single subsuming stereotype, and a set (possibly empty) of subsumed stereotypes and a set (again, possibly empty) of individuals currently believed to be modeled by the stereotype. The real contents of the stereotype, however, consists of two databases of facts and rules, one definite and the other default. The definite facts and rules constitute a *definition* of sorts for the stereotype, in that they determine what is necessarily true of any individual classified under this stereotype. If our knowledge of an individual contradicts this information, then he cannot be a member of this stereotype or any of its descendent stereotypes. The default facts and rules, on the other hand, define our initial beliefs about any individual who is classified in this stereotype. They can be overridden by specific information that we later learn about the individual. The final component of a stereotype is a collection of meta-knowledge about the predicates used in the definite and default knowledge bases. This information is used in the processing of negative queries and is described later.

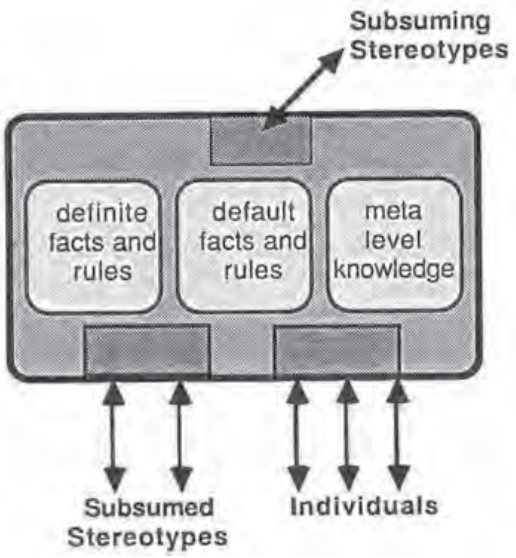


Figure 5. A stereotype

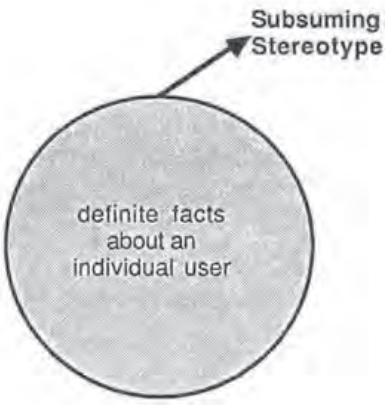


Figure 6. An individual

As an example, consider modeling the knowledge a programmer might have. We can be sure that a programmer will know what a *file* is, but we can only guess that a programmer will know what a *file directory* is. If we have categorized a given user under the *programmer stereotype* and discover (perhaps through direct

interaction with him) that he is not familiar with the concept of a *file* then we can conclude that we had improperly chosen a stereotype and must choose a new one. But if we got the information that he did not know what a *file directory* was, this would not rule out the possibility of his being a programmer.

The knowledge structure for an individual is simpler than that for a stereotype. An individual has exactly one stereotype (in the current version) with which he is currently associated, and a collection of definite, ground facts which are true, as is shown in Figure 6. The facts known about an individual override any default facts known or deducible from the stereotype associated with the individual. They must, however, be consistent with the stereotype's definite knowledge base. If a contradiction arises, then the individual must be reclassified as belonging to another stereotype.

There are a number of ways we might go about trying to reclassify an individual when it is discovered that the current stereotype is no longer valid. In general, we can distinguish *domain independent* and *domain dependent* techniques. Domain dependent techniques are particularly useful when we have some understanding of the reasons why a user might be mis-classified. For example, the user may be learning new domain knowledge while he is using the system and this new knowledge may cause him to "grow out of" the current stereotype. In such a situation, we may wish to directly encode a "growth path" of stereotypes. Another domain dependent technique is to treat certain user attributes as triggers which, when true, evoke particular stereotypes associated with them [RICH*].

Domain independent techniques for selecting a new stereotype in the face of conflicting information involve (conceptually) two stages: generating candidates for the new stereotype and selecting from among them. A very general and powerful technique would be to use a classification strategy similar to that used in the KL-ONE family of representation languages [BRAC85]. Such a strategy would consider all of the possible stereotypes and find the set of *most specific* subsuming stereotypes.

In the *GUMS* system, we have used a very simple domain independent strategy for finding a new stereotype when the known facts about an individual contradict some facts associated with the user's current stereotype. The ancestors of the current stereotype are searched in order of specificity until one is found in which there is no contradiction. Thus, we are guaranteed to find a stereotype which subsumes the user, although it will in general not be the most specific one possible.

Although we have adopted this strategy because of its relative simplicity, both in terms of complexity and inherent computational costs, it does correspond to some of our intuitions on how people adopt new stereotypes. The intuition is that the more we learn about an individual, the less we depend on stereotypic reasoning to fill in our beliefs about him. This is mirrored in the *GUMS* system in its stereotype shifting strategy. As the system learns more about a user, it will discover contradictions with its current stereotype model. When faced with a contradiction, it moves up the hierarchy until it finds a stereotype in which the contradiction is resolved. Moving up the hierarchy corresponds to a lessening of stereotypic reasoning, since the higher models are necessarily less complete.

3.2.2. Default Reasoning with Rules

Within a stereotype, we need to describe some knowledge as being necessarily true with respect to (properly classified) members of the stereotype and other knowledge as *default* information. This default information is to be taken as true in the absence of contradictory information. In the *GUMS* system, each rule or fact in the stereotype is either *definite* (i.e. necessarily true) or *default*. The **certain/1** predicate is used to introduce a definite fact or rule and the **default/1** predicate to indicate a default fact or rule, as in:

certain (P). a definite fact: P is true.

certain (P if Q). a definite rule: P is true if Q is definitely true, and P is assumed to be true if Q is only assumed to be true.

default (P). a default fact: P is assumed to be true unless it is known to be false.

default (P if Q). a default rule: P is assumed to be true if Q is true or assumed to be true and there is no definite evidence to the contrary.

Figure 7 depicts graphically the relationships between clauses with *certain* and with *default* information.

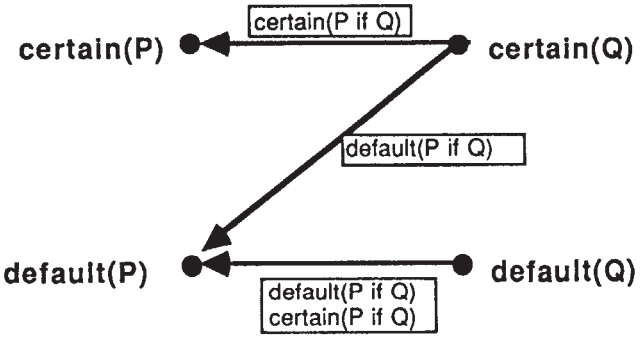


Figure 7. The relationship between certain and default clauses

As an example, consider a situation in which we need to model a person's familiarity with certain terms. This is a common situation in systems that produce text as explanations or in response to queries where there is a wide variation in the users' familiarity with the domain. For example, we might use the rules and facts:

(a) default(understandsTerm(ram)).
(b) default(understandsTerm(rom) if understandsTerm(ram)).
(c) certain(understandsTerm(pc) if understandsTerm(ibmpc)).
(d) certain(~understandsTerm (cpu)).

to represent the following assertions, all of which are considered as pertaining to a particular user with respect to the stereotype containing the rules:

(a) Assume the user understands the term *ram* unless we know otherwise.

(b) Assume the user understands the term *rom* if we know or believe he understands the term *ram*, unless we know otherwise.
(c) This user understands the term *pc* if he understands the term *ibmpc*.
(d) This user does not understand the term *cpu*.

3.2.3. Negation As Failure

GUMS also treats negation as failure in some cases as a default rule. In general, logical deduction is performed using an open world assumption. That is, the failure to prove a proposition is not taken as evidence that it is not true. Many logic programming languages, such a Prolog, encourage the interpretation of unprovability as logical negation. Two approaches have been put forward to justify the negation as failure rule – making the *closed world assumption* and *completing the database*. In the closed world approach [REIT78], we assume that anything not inferable from the database is by necessity false. Making this meta-level assumption introduces a potential problem in that we do not know what the corresponding object-level assumptions are. The second approach is based upon the concept of a completed database [CLKL78]. A completed database is the database constructed by rewriting the set of clauses defining each predicate to an **if and only if** definition that is called the completion of the predicate. The purpose of the completed definition is to indicate that the clauses that define a predicate define **every** possible instance of that predicate.

Any approach to negation as failure requires that a negated goal be *ground* (i.e. contain no uninstantiated variables) before execution.[2] Thus we must have some way of insuring that every negated literal will be bound. In *GUMS* we have used a simple variable typing scheme to achieve this, as will be discussed later.

We have used a variant of the completed database approach to show that a predicate within the scope of a negation is closed. Making the meta-level assertion *declare (closed(P))* signifies that the definition of the predicate *P* is actually an *if and only if* definition. This same technique was used by Kowalski [KOWA79] to indicate completion. We further assume that a predicate is not completed unless it is declared closed. For example, consider the following clauses:

```
declare(closed(p)).
p if q
r if q
```

Since the predicate *p* is complete, proving that *q* is false allows us to deduce that *p* is false. Since the predicate *r* is not complete, proving that *q* is false does not allow us to conclude that *r* is false, although it does provide weak evidence that *r* may be false.

Thus in *GUMS* we have the ability to express that a default should be taken from the lack of certain information (i.e. negation as failure), as well as from the presence of certain information (i.e. default rules). For example, we can have a

[2] A slightly less restrictive rule could allow a partially instantiated negated goal to run but would produce the wrong answer if any variable was bound.

default rule for the programmer stereotype that can conclude knowledge about linkers from knowledge about compilers, as in:

```
default(knows(linkers) if knows(compilers))
```

We can also have a rule that will take the lack of knowledge about compilers as an indication that the user probably knows about interpreters, as in:

```
certain(knows(interpreters) if ~ knows(compilers))
```

This system also allows explicit negative facts, both definite and default. When negation is proved in reference to a negative fact then negation is not considered a default case. Similarly, negation as failure is not considered a default when the predicate being negated is closed. Such distinctions are possible because the *GUMS* interpreter is based on a four value logic.

The distinction between truth or falsity by default (i.e. assumption) and truth or falsity by logical implication is an important one to this system. The central predicate of the system is the two argument predicate *show*, which relates a goal expressed as a literal to a truth value. Thus *show (Goal, Val)* returns in the variable *Val* the "strongest" belief in the literal *Goal*. The variable *Val* can be instantiated to *true, false, assume (true)*, or *assume (false)*. The meanings of these values are as follows:

true	definitely true according to the current database.
assume (true)	true by assumption (i.e. true by default)
assume (false)	false by assumption
false	definitely not true according to the current database.

These values represent truth values for a given user with respect to a given stereotype. If the stereotype is not appropriate, then even definite values may have to change.

The relative strength of the four truth values is shown by the partial ordering in Figure 8. Evidence that a fact is either *true* or *false* is the strongest and outweighs any evidence that the fact is assumed to be true or false. The *GUMS* system is "optimistic" in that it takes direct evidence that a fact is assumed to be true to be stronger than evidence that a fact is assumed to be false. This choice has been made to reflect the fact that, in our current applications, the unprovability of a proposition is weak evidence that it is false. Note that this multi-valued logic is simpler than those based on the notion of a *bilattice* discussed by Ginsberg [GINS86].

Having a four value logic allows us to distinguish conclusions made from purely logical information from those dependent on default information. The four value logic also allows a simple type of introspective reasoning that may be useful for modeling the beliefs of the user. We currently use a default rule to represent an uncertain belief about what the user knows or believes, but we could imagine a situation where we would like to model uncertainties that the user has in his beliefs or knowledge. One such predicate is an embedded *show* predicate. The embedded *show* relation holds between a goal G and a truth value X if the strongest truth value that can be derived for G is X.

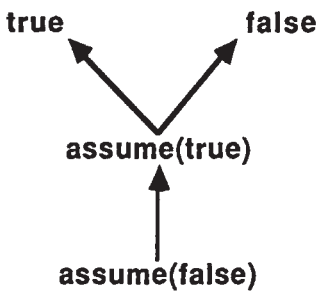


Figure 8. The relative strength of the four truth values

As an example, consider a situation in which we need to express the fact that a user will use a operating system command that he believes might erase a file only if he is certain that he knows how to use that command. We might encode this using the *show* predicate as follows:

```
certain(okay_to_use(Command) if
        can_erase_files(Command),
        show(know(Command),true)).
```

The embedded show predicate forces the *know (Command)* goal to be a definite fact in order for the conclusion to hold.

Another predicate, *assumed (Pred)*, will evaluate the truth of *Pred* and "*strengthen*" the result. The *strengthen* operation maps assumed values into definite values (e.g. **assume(true)** becomes **true, assume(false)** becomes **false** and **true** and **false** remain unchanged). The *assumed* predicate is used to express a certain belief from an uncertain knowledge or belief. For example we might want to express a rule that a user will always want to use a screen editor if he believes one may be available.

```
certain (willUse(screenEditor) if
        assumed(available(screenEditor))).
```

The interpreter that *GUMS* is based on is a metalevel interpreter written in Prolog. The interpreter must generate and compare many possible answers to each subquery, because of the multiple value logic and the presence of explicit negative information. Strong answers to a query (i.e. *true* and *false*) are sought first, followed by weak answers (i.e. *assume(true)* and *assume(false)*). Because strong answers have precedence over weak ones, it is not necessary to remove weak information that contradicts strong information.

Another feature of this system is that we can specify the **types** of arguments to predicates. This type information is used to allow the system to handle non-ground goals. In our system, a **type** provides a way to enumerate a complete set of possible values subsumed by that type. When the top-level *show* predicate is given a partially instantiated goal to solve, it uses the type information to generate a stream of consistent, fully instantiated goals. These ground goals are then tried sequentially.

That goals must be fully instantiated follows from the fact that negation as failure is built into the evaluation algorithm. Complex terms are instantiated to every pattern allowed by the datatype given the full power of unification. To specify the

type information, one must declare the data types, the subtype relations between the types, the instances of these types and, for each predicate, the required type of each argument. For example, the following assertions declare that the **canProgram** predicate ranges over instances of *person* and *programmingLanguage*, that the type *functionalLanguage* is a sub-type of *programmingLanguage* and that the value *scheme* is an instance of the type *functionalLanguage*:

```
declare(schema(canProgramm(person,programmingLanguage))).
declare(subtype(programmingLanguage,functionalLanguage)).
declare(instance(functionalLanguage,scheme)).
```

To see why it is necessary to fully instantiate goals, consider the following problem. We would like to model a programmer's knowledge of tools available on a system. This might be done in order to support an intelligent help system, for example. We might want to encode the following general rules:

C Programmers tend to use all of the relevant tools that are available to them.
If a C programmer does not use some relevant tool, then this is probably because he is not aware of it.
A person is probably a C programmer if they have ever used the C compiler.
Every C tool is relevant to a C programmer.
We have complete knowledge of every tool a user has used.
Lint is a C tool.

which could be expressed in *GUMS* as follows

```
default(knows(Tool) if relevant(Tool), cprogrammer).
default(~knows(Tool) if ~used(Tool), relevant(Tool)).
default(cprogrammer if used(cc)).
certain(relevant(Tool) if cTool(Tool), cprogrammer).
declare(closed(used)).
ctool(lint).
```

Suppose we want to discover a tool of which an individual user is not aware. To answer this, we would like to pose the query *~knows(X)* where X is an uninstantiated variable. Let's assume that this is to be done with respect to an individual knowledge base that includes the following definite facts:

```
used(cc).
used(emacs).
used(ls).
```

The normal interpretation of variables in Prolog goals is that they are *existentially* quantified. The normal interpretation of negated goals is failure as negation, i.e. unprovability. When these two are combined, as in the goal **~used(X)**, the result is that the negation "flips" the quantifier, resulting in a goal which is the equivalent of $\forall$**x not(used(x)).** Moreover, satisfying a negated goal will never result in any variables in the goal being instantiated. Thus, in the example above, we would be unable to discover any tools which this user did not know about since the negated goal **~used(X)** must fail because it is possible to prove **used(emacs)**.

It is possible to "fix" this particular example by reordering the subgoals in the second rule, to get:

```
default(~knows(Tool) if relevant(Tool), ~used(Tool)).
```

This will ensure that the variable *Tool* is instantiated before the negated subgoal is reached. This solution, and the more general one of delaying the evaluation of a negated goal until it is ground, will not always work. Our strategy is to force all goals to be completely ground and thereby avoid this problem.[3] Posing the goal **~knows(X)** to *GUMS* results in the attempt to satisfy a stream of goals which includes the ground goal **~knows(lint)**.

4. The Current GUMS System

The *GUMS* system is currently implemented in CProlog. At the heart of the system is a backward chaining meta-interpreter that implements the default reasoning engine.

Our current system has several limitations. One problem is that it does not extract all of the available information from a new fact learned about the user. If we assert that a predicate is *closed*, we are saying that the set of (certain) rules for the predicate forms a **definition**, i.e. a necessary and sufficient description. In our current system, however, the information still only flows in one direction. For example, suppose that we would like to encode the rule that a user knows about *I/O redirection* if and only if they know about *files* and about *pipes*. Further, let's suppose that the default is that a person in this stereotype does not know about *files* or *pipes*. This can be expressed as:

```
certain(knows(io_redirection) if
        knows(pipes),
        knows(files)).
default(~knows(pipes)).
default(~knows(files))
declare(closed(knows(io_redirection))).
```

If we learn that a particular user **does** know about *I/O redirection* then it should follow that he necessarily knows about both *files* and *pipes*. Adding the assertion

```
certain(knows(io_redirection))
```

however, will make no additional changes in the data base. The values of **knows(pipes)** and **knows(files)** will not change. A sample run after this change might be:

```
?- show(knows(io_redirection),Val).
        Val = true
?- show(knows(pipes),Val).
        Val = assume(false)
```

[3] This is similar to the treatment of negation used by Walker [WALK87].

```
?- show(knows(files),Val).
        Val = assume(false).
```

The reason for this problem is that the current interpreter was designed to be able to incorporate new information without actually using a full truth maintenance system. Before a fact *F* with truth value *V* is to be added to the data base, *GUMS* checks to see if an inconsistent truth value *V'* can be derived for *F*. If one can be, then a new stereotype is sought in which the contradiction goes away. New knowledge that does not force an obvious inconsistency within the database is added as is. Neither redundant information nor existing default information affect the correctness of the interpreter. Subtler inconsistencies are possible, of course.

Another limitation of the current system its inefficiency. The use of default rules requires us to continue to search for solutions for a goal until a strong one is found or all solutions have been checked. These two limitations may be addressable by redesigning the system to be based on a forward chaining truth maintenance system. The question is whether the relative efficiency of forward chaining will offset the relative inefficiency of truth maintenance. The use of an assumption based truth maintenance system [DEKL86] is another alternative that we will investigate.

5. The GUMS Command Language

Our current implementation provides the following commands to the application:

show(Query,Val) succeeds with *Val* as the strongest truth value for the goal *Query*. A Query is a partially or fully instantiated positive or negative literal. *Val* is returned and is the value of the current belief state. If *Query* is partially instantiated then it will return more answers upon backtracking if possible. In general, one answer will be provided for every legal ground substitution that agrees with current type declarations.

add(Fact,Status) sets belief in *Fact* to true if it is consistent with the current set of beliefs about the user. As described earlier, consistency is checked by trying to deduce the negation of the Fact. If *Fact*, or any legal instance of it, is found to be contradictory, *GUMS* adopts successively higher stereotypes until one is found in which all of the added facts are consistent. If no stereotype is successful then no stereotype is used and all answers will be based entirely on added facts. *Fact* must be partially or fully instantiated and can be either a positive or negative literal. *Status* must be uninstantiated and will be bound to a message describing the result of the addition (e.g. one of several error messages, *ok*, the name of a new stereotype, etc.).

create_user(UserName,Stereotype,File,Status) stores the current user model if necessary and creates a new model representing the user who then becomes the current user. *UserName* is instantiated to the desired name. *Stereotype* is the logical name of the stereotype that the system should assume to hold. *File* is the name of the file in which information pertaining to the user will be stored. *Status* is instantiated by the system and returns error messages. A user must be created in order for the system to be able to answer queries.

store_current(Status) stores the current user's information and clears the workspace for a new user. *Status* is instantiated by the system when an error occurs.

restore_user(User,Status) restores a previous user after saving the current user if necessary. *User* is the name of the user. *Status* is instantiated by the system to pass error messages.

done stores the system state of the user modeling system, saving the current user if necessary. This command should be the last command issued and needs to be issued at the end of every session.

6. Conclusions

Many interactive systems have a strong necessity to maintain models of individual users. We have presented a simple architecture for a general user modeling utility based on the ideas of default logic. This approach provides a simple system that can maintain a database of known information about users as well as use rules and facts associated with a stereotype which is believed to be appropriate for this user. The stereotype can contain definite facts and rules of inference as well as default information and rules. The rules can be used to derive new information, both definite and assumed, from the currently believed information about the user.

We believe that this kind of system will prove useful to a wide range of applications. We have implemented an initial version in Prolog and are beginning to use it to support the modeling needs of several projects. There are a number of interesting issues that need to be explored in extending the *GUMS* system, including the use of different reasoning systems such as assumption based TMS's and the development of better strategies for assigning or re-assigning an individual to a stereotype.

7. References

[ALLE80], [ALLE82b], [BRAC85], [BROW78], [CARB83], [CARB88], [CARB*], [CARN83c], [CHIN86], [CHIN*], [CLKL78], [DEKL86], [FINI83], [GERS81], [GINS86], [HOEP83b], [HOWE84], [JOHN84], [JOSH84a], [JOSH84b], [KAPS82], [KASS87a], [KASS87b], [KASS88b], [KASS*], [KOBS84], [KOWA79], [MCCO85a], [MCCO*], [MCKE85a], [MCKE85c], [MORI85b], [MORI*], [MOTR86], [PARI88], [PERR88], [POLL85], [REIT78], [REIT80], [RICH79b], [RICH83], [RICH*], [SCHU86b], [SHRA82], [SIDN81b], [SLEE82a], [SLEE85], [SMIB82], [SWAR83], [WALK87], [WALL82], [WEBB84], [WILE86]

Appendices

Bibliography

This bibliography contains all references that have been cited in the book. The codes at the end of the references indicate the citing authors and the page numbers where cited (codes with asterisks always refer to chapters in this book). Moreover, a number of papers have been added which were not cited, but certainly should form part of a representative bibliography of the field of user modeling. These references contain no author codes and page numbers.

Abbreviations:

IJCAI-77 5th International Joint Conference on Artificial Intelligence, Cambridge, MA, 1977.
IJCAI-81 7th International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981.
IJCAI-83 8th International Joint Conference on Artificial Intelligence, Karlsruhe, Fed. Rep. of Germany, 1983.
IJCAI-85 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, 1985.
IJCAI-87 10th International Joint Conference on Artificial Intelligence, Milan, Italy, 1987.
IJCAI-89 11th International Joint Conference on Artificial Intelligence, Detroit, MI, 1989.
AAAI-80 1st National Conference on Artificial Intelligence, Stanford, CA, 1980.
AAAI-82 2nd National Conference on Artificial Intelligence, Pittsburgh, PA, 1982.
AAAI-83 3rd National Conference on Artificial Intelligence, Washington, DC 1983.
AAAI-84 4th National Conference on Artificial Intelligence, Austin, TX, 1984.
AAAI-86 5th National Conference on Artificial Intelligence, Philadelphia, PA, 1986.
AAAI-87 6th National Conference on Artificial Intelligence, Seattle, WA, 1987.
AAAI-88 7th National Conference on Artificial Intelligence, St. Paul, MN, 1988

[ABEL73] Abelson, R. (1973): The Structure of Belief Systems. In: R. C. Schank and K. M. Colby, eds.: Computer Models of Thought and Language. San Francisco, CA: Freeman. [QUIL*: 111]

[ABEL79] Abelson, R. (1979): Differences between Belief and Knowledge Systems. Cognitive Science 3, 355–366. [QUIL*: 111]

[ADEL82] Adelman, H. S. (1982): Identifying Learning Problems at an Early Age: A Critical Appraisal. Journal of Clinical Child Psychology 11, 255–261. [COHE*: 314]

[AIRE83] Airenti, G., B. G. Bara and M. Colombetti (1983): Planning Perlocutionary Acts. In: Proc. IJCAI-83, 78–80. [KOBS*: 61]

[AIRE84] Airenti, G., B. G. Bara and M. Colombetti (1984): Plan Formation and Failure Recovery in Communicative Acts. In: Proc. of the 1984 European Conference on Artificial Intelligence, Pisa, Italy, 259–268. [KOBS*: 61]

[AIRE86] Airenti, G., B. G. Bara and M. Colombetti (1986): Modeling the Relationship between User and System in Man-Machine Dialogue. Paper presented at the International Workshop on User Modeling, Maria Laach, Fed. Rep. of Germany. [SPAR*: 361]

[AJZE71] Ajzen, I. (1971): Attribution of Dispositions to an Actor: Effects of Perceived Decision Freedom and Behavioral Utilities. Journal of Personality and Social Psychology 18, 144–156. [JAME*: 287, 293]

[ALLE78] Allen, J.F. and C.R. Perrault (1978): Participating in Dialogues: Understanding via Plan Deduction. AI-Memo 78-4, Dept. of Computer Science, Univ. of Toronto, Canada.

[ALLE79] Allen, J.F. (1979): A Plan-Based Approach to Speech Act Recognition. TR 131/79, Dept. of Computer Science, Univ. of Toronto, Canada. [WAHL*: 5, 19; MORI*: 370, 374f., 382]

[ALLE80] Allen, J.F. and C.R. Perrault (1980): Analyzing Intention in Utterances. Artificial Intelligence 15, 143–178. [WAHL*: 12, 32; KOBS*: 57, 60; CHIN*: 78; CARB*: 135ff., 153; FINI*: 414]

[ALLE82a] Allen, J.F. (1982): ARGOT: A System Overview. TR101, Computer Science Dept., Univ. of Rochester, NY. [WAHL*: 19]

[ALLE82b] Allen, J.F., M.A. Frisch and D.J. Litman (1982): ARGOT: The Rochester Dialogue System. In: Proc. AAAI-82, 66–70. [FINI*: 413]

[ALLE83a] Allen, J.F. (1983): ARGOT: A System Overview. In: Computers and Mathematics with Applications 9, 97–109. [KOBS*: 58]

[ALLE83b] Allen, J.F. (1983): Recognizing Intentions from Natural Language Utterances. In: M. Brady and R.C. Berwick, eds.: Computational Models of Discourse. Cambridge, MA: MIT Press. [WAHL*: 19; KOBS*: 53, 57, 59ff.]

[ALLE86] Allen, J.F. and D.J. Litman (1986): Plans, Goals, and Natural Language. In: Proceedings of the IEEE 74, 939–947. [WAHL*: 6]

[ALLE87] Allen, J.F. (1987): Understanding Natural Language. Menlo Park, CA: Benjamin/Cummings. [WAHL*: 6]

[ALLG89] Allgayer, J., K. Harbusch, A. Kobsa, C. Reddig, N. Reithinger and D. Schmauks (forthcoming): XTRA: A Natural-Language Access System for Expert Systems. International Journal of Man-Machine Studies. [WAHL*: 5, 18f., 33]

[ALVA86] Alvarado, S., M. Dyer and M. Flowers (1986): Editorial Comprehension in OpED through Argument Units. In: Proc. AAAI-86, 250–256. [QUIL*: 112]

[AMLI88] Amling, S. (1988): Die Möglichkeit der Benutzermodellierung bei der Entwicklung von Expertensystemen unter Verwendung einer Schale am Beispiel von Expert System Environment. Master Thesis, Berufsakademie Stuttgart, Fed. Rep. of Germany.

[AMME86] Ammersbach, K. (1986): Benutzermodelle für Information-Retrieval-Systeme. Master Thesis, Dept. of Computer Science, Technical University of Darmstadt, Fed. Rep. of Germany.

[ANDJ85a] Anderson, J. R., C. F. Boyle and B. J. Reiser (1985): Intelligent Tutoring Systems. Science 228, 456–462. [KASS*: 389]

[ANDJ85b] Anderson, J. R., C. F. Boyle and G. Yost (1985): The Geometry Tutor. In: Proc. IJCAI-85, 1–7. [CHIN*: 77, 100; QUIL*: 116, 125; KASS*: 393]

[ANDN81] Anderson, N. H. (1981): Integration Theory Applied to Cognitive Responses and Attitudes. In: R. E. Petty, T. M. Ostrom and T. C. Brock, eds.: Cognitive Responses in Persuasion. Hillsdale, NJ: Erlbaum. [JAME*: 269]

[ANDR77] Anderson, R. C., R. J. Spiro and M. C. Anderson (1977): Schemata as Scaffolding for the Representation of Information in Connected Discourse. Technical Report 24, Center for the Study of Reading, Urbana, IL. [PARI*: 212]

[APPE82a] Appelt, D. E. (1982): Planning Natural-Language Utterances. In: Proc. AAAI-82, 59–62. [PARI*: 201]

[APPE82b] Appelt, D. E. (1982): Planning Natural-Language Utterances to Satisfy Multiple Goals. Technical Note 259, SRI International, Menlo Park, CA. [WAHL*: 5, 19; MORI*: 375]

[APPE85] Appelt, D. E. (1985): Planning English Sentences. Cambridge etc.: Cambridge Univ. Press. [WAHL*: 5, 16, 19; KOBS*: 53, 58ff., 62f.; PARI*: 201, 203]

[BAKE57] Baker, D. L., R. B. Brownlee and R. W. Fuller (1957): Elements of Physics. Newton, MA: Allyn and Bacon. [PARI*: 205]

[BALL84] Ballard, B., J. Lusth and N. Tinkham (1984): LDC-1: A Transportable, Knowledge-Based Natural Language Processor for Office Environments. ACM Trans. on Office Information Systems 2, 1–25. [WAHL*: 30]

[BARB82] Barber, G. R. (1982): Office Semantics. Ph.D. Thesis, Dept. of Electrical Engineering, MIT, Cambridge, MA. [MORI*: 374]

[BARN86] Barnden, J. (1986): Interpreting Propositional Attitude Reports: Towards Greater Freedom and Control. In: Proc. of the 7th European Conference on Artificial Intelligence, Brighton, England, 398–412. [KOBS*: 55]

[BARR77] Barr, A. (1977): Meta-Knowledge and Memory. Working Paper HPP-77-37, Heuristic Programming Project, Stanford Univ., Stanford, CA. [CHIN*: 98]

[BELK84] Belkin, N. J. (1984): Cognitive Models and Information Transfer. Social Science Information Studies 4, 111–129. [SPAR*: 360, 362]

[BENY87] Benyon, D., P. Innocent and D. Murray (1987): System Adaptivity and the Modelling of Stereotypes. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North-Holland.

[BOBR77] Bobrow, J. S. and T. Winograd (1977): An Overview of KRL, a Knowledge Representation Language. Cognitive Science 1, 3–46. [MCCO*: 245]

[BOGG85] Boggs, W. M., J. G. Carbonell, M. Kee and I. Monarch (1985): The DYPAR-I Tutorial and Reference Manual. Technical Report, Computer Science Dept., Carnegie-Mellon Univ., Pittsburgh, PA. [LEHM*: 167]

[BOGU85] Boguraev, B. K. (1985): User Modelling in Cooperative Natural Language Front Ends. In: G. N. Gilbert and C. Heath, eds.: Social Action and Artificial Intelligence. London: Gower Press. [WAHL*: 5]

[BONA87] Bonarini, A. (1987): User Modeling in Person-Machine Conversation: A Computational Approach. In: Proc. of COGNITIVA-87, Paris, France, 377–382. [WAHL*: 28, 33]

[BOTM87] Botman, A. M., A. J. Kok and L. Siklossy (1987): Using User Models and Automatic Expertise Acquisition in Man-Machine Interaction. In: Proc. of the 7th International Workshop on Expert Systems and their Applications, Avignon, France, 389–398. [WAHL*: 9]

[BRAC85] Brachman, R. J. and J. G. Schmolze (1985): An Overview of the KL-ONE Knowledge Representation System. Cognitive Science 9, 171–216. [FINI*: 422]

[BREC88] Brecht, B. and M. Jones (1988): Student Models: The Genetic Graph Approach. International Journal of Man-Machine Studies 28, 483–504.

[BRIT63] Britannica Junior Encyclopedia. Chicago: Encyclopedia Britannica Inc., 1963. [PARI*: 201, 205, 209]

[BRIT84] The New Encyclopedia Britannica. Chicago: Encyclopedia Britannica Inc., 1984. [PARI*: 201, 205]

[BROO85] Brooks, H. M., P. J. Daniels and N. J. Belkin (1985): Problem Description and User Models: Developing an Intelligent Interface for Document Retrieval Systems. In: Informatics 8: Advances in Intelligent Retrieval. London: Aslib. [SPAR*: 362]

[BROO86] Brooks, H. M. (1986): An Intelligent Interface for Document Retrieval Systems: Developing the Problem Description and Retrieval Strategy Components. Ph.D. Thesis, Dept. of Information Science, City Univ., London, England. [SPAR*: 362f.]

[BROW75] Brown, J. S., R. Burton, M. Miller, J. DeKleer, S. Purcell, C. Hauseman and R. Bobrow (1975): Steps towards a Theoretical Foundation for Complex, Knowlege-Based CAI. Technical Report 3135, Bolt, Beranek and Newman, Cambridge, MA. [KASS*: 387, 389]

[BROW76] Brown, J. S., R. Rubinstein and R. Burton (1976): A Reactive Learning Environment for Computer Assisted Electronics Instruction. Technical Report 3314, Bolt, Beranek and Newman, Cambridge, MA. [KASS*: 388]

[BROW77] Brown, J. S., R. Burton, C. Hauseman, I. Goldstein, B. Huggins and M. Miller (1977): Aspects of a Theory for Automated Student Modelling. Technical Report 3549, Bolt, Beranek and Newman, Cambridge, MA. [KASS*: 394]

[BROW78] Brown, J. S. and R. R. Burton (1978): Diagnostic Models for Procedural Bugs in Basic Mathematical Skills. Cognitive Science 2, 155–192. [CHIN*: 77, 100; QUIL*: 116, 125; MCCO*: 237; KASS*: 392, 394; FINI*: 414]

[BROW80] Brown, J. S. and K. VanLehn (1980): Repair Theory: A Generative Theory of Bugs in Procedural Skills. Cognitive Science 4, 379–426. [KASS*: 405]

[BROW82] Brown, J. S., R. R. Burton and J. DeKleer (1982): Pedagogical, Natural Language and Knowledge Engineering Techniques in Sophie 1, 2 and 3. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 388f.]

[BRUC75] Bruce, B. C. (1975): Belief Systems and Language Understanding. Technical Report No. 2973, Bolt, Beranek and Newman, Cambridge, MA.

[BUCH84] Buchanan, B. and E. Shortliffe (1984): Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA etc.: Addison-Wesley. [COHE*: 325; SPAR*: 352]

[BURG78] Burge, T. (1978): Belief *De Re*. Journal of Philosophy 74, 338–362. [KOBS*: 55]

[BURT76] Burton, R. R. and J. S. Brown (1976): A Tutoring and Student Modelling Paradigm for Gaming Environments. In: Proc. of the 1st Symposium on Computer Science and Education, Anaheim, CA, 236–246. [CHIN*: 77]

[BURT82] Burton, R. R. (1982): Diagnosing Bugs in a Simple Procedural Skill. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [QUIL*: 116, 125; KASS*: 394]

[CANT76] Cantor, N. and W. Mischel (1976): Traits as Prototypes: Effects on Recognition Memory. Journal of Personality and Social Psychology 35, 38–48. [RICH*: 36]

[CARB83] Carberry, S. (1983): Tracking User Goals in an Information-Seeking Environment. In: Proc. AAAI-83, 59–63. [WAHL*: 12; KOBS*: 59f.; CHIN*: 78; CARB*: 155; PARI*: 201, 203; FINI*: 413f.]

[CARB84] Carberry, S. (1984): Understanding Pragmatically Ill-Formed Input. In: Proc. of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Stanford Univ., Stanford, CA, 200–206. [MCCO*: 243]

[CARB85a] Carberry, S. (1985): A Pragmatics Based Approach to Understanding Intersentential Ellipsis. In: Proc. of the 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, IL, 188–197.

[CARB85b] Carberry, S. (1985): Pragmatic Modeling in Information System Interfaces. Ph.D. Thesis, Dept. of Computer and Information Sciences, Univ. of Delaware, Newark, DE. [KOBS*: 59f.]

[CARB86a] Carberry, S. (1986): TRACK: Toward a Robust Natural Language Interface. In: Proc. of the 6th Canadian Conference on Artificial Intelligence, Montreal, Canada, 84–88.

[CARB86b] Carberry, S. (1986): User Models: The Problem of Disparity. In: Proc. of the 11th International Conference on Computational Linguistics, Bonn, Fed. Rep. of Germany, 29–34.

[CARB88] Carberry, S. (1988): Modeling the User's Plans and Goals. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), 23–37. [PARI*: 201; CARB*: 133; FINI*: 414]

[CARB*] Carberry, S.: Plan Recognition and its Use in Understanding Dialog. In this volume. [WAHL*: 12; RICH*: 39; KOBS*: 59; CHIN*: 78; QUIL*: 129; PARI*: 201; MCCO*: 251; JAME*: 256, 260, 278f.; SPAR*: 361f.; MORI*: 375f.; FINI*: 414]

[CARN70a] Carbonell, J. R. (1970): AI in CAI: An Artificial Intelligence Approach to Computer-Aided Instruction. IEEE Transactions on Man-Machine Systems 11, 190–202. [KASS*: 390]

[CARN70b] Carbonell, J. R. and A. M. Collins (1970): Mixed-Initiative Systems for Training and Decision-Aid Applications. Technical Report ESD-TR-70-373, Electronics

Systems Division, Laurence G. Hanscom Field, US Air Force, Bedford, MA. [MCCO*: 244, 248]

[CARN81] Carbonell, J. G. and P. J. Hayes (1981): Dynamic Strategy Selection in Flexible Parsing. In: Proc. of the 19th Annual Meeting of the Association for Computational Linguistics, Stanford, CA, 143–147. [LEHM*: 167]

[CARN83a] Carbonell, J. G. (1983): Discourse Pragmatics in Task – Oriented Natural Language Interfaces. In: Proc. of the 21st Annual Meeting of the Association for Computational Linguistics, Cambridge, MA, 164–168. [LEHM*: 164]

[CARN83b] Carbonell, J. G. (1983): The Role of User Modelling in Natural Language Interface Design. Report CMU-CS-83-115, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA.

[CARN83c] Carbonell, J. G., W. M. Boggs, M. L. Mauldin and P. G. Anick (1983): The XCALIBUR Project: A Natural Language Interface to Expert Systems. In: Proc. IJCAI-83, 653–656. [WAHL*: 5, 30; FINI*: 413]

[CARN83d] Carbonell, J. G. and P. J. Hayes (1983): Recovery Strategies for Parsing Extragrammatical Language. American Journal of Computational Linguistics 9, 123–146. [LEHM*: 163f.]

[CARO83] Carlson, L. (1983): Dialogue Games: An Approach to Discourse Analysis. Dordrecht, The Netherlands: Reidel. [JAME*: 285]

[CARR77] Carr, B. and I. Goldstein (1977): Overlays: A Theory of Modelling for Computer Aided Instruction. AI Memo 406, MIT, Cambridge, MA. [WAHL*: 7; CHIN*: 77; KASS*: 388, 390, 400]

[CARV84] Carver, N. F., V. R. Lesser and D. L. McCue (1984): Focusing in Plan Recognition. In: Proc. AAAI-84, 42–48. [CARB*: 153]

[CECI84] Ceci, S. J. and D. Peters (1984): Letters of Reference: A Naturalistic Study of the Effects of Confidentiality. American Psychologist 39, 29–31. [JAME*: 276]

[CHAN78] Chang, C. L. (1978): Finding Missing Joins for Incomplete Queries in Relational Databases. Technical Report RJ 2145, IBM Research Laboratory, San Jose, CA. [CARB*: 146]

[CHAR83] Charniak, E. (1983): The Bayesian Basis of Common Sense Medical Diagnosis. In: Proc. AAAI-83, 70–73. [RICH*: 47]

[CHEI89] Cheikes, B. A. (1989): The Architecture of a Cooperative Respondent. Technical Report MS-CIS-89-13, Dept. of Computer and Information Science, Univ. of Pennsylvania, PA.

[CHEV78] Chevrolet Service Manual. Detroit, MI: General Motors Corporation, 1987. [PARI*: 201, 205]

[CHI81] Chi, M. T. H., R. Glaser and E. Rees (1981): Expertise in Problem Solving. In: R. J. Sternberg, ed.: Advances in the Psychology of Human Intelligence. Hillsdale, NJ: Lawrence Erlbaum. [PARI*: 211]

[CHIN83] Chin, D. N. (1983): A Case Study of Knowledge Representation in UC. In: Proc. IJCAI-83, 388–390. [CHIN*: 82]

[CHIN86] Chin, D. N. (1986): User Modelling in UC, the UNIX Consultant. In: Proc. of the CHI'86, Boston, MA, 24–28. [KOBS*: 57, 65; CHIN*: 74; PARI*: 202, 229; COHE*: 325, 330; FINI*: 414]

[CHIN87] Chin, D. N. (1987): Intelligent Agents as a Basis for Natural Language Interfaces. Ph.D. Thesis, Dept. of Computer Science, Univ. of California, Berkeley, CA. Also Available as Technical Report UCB/CSD 88/396, Division of Computer Science, Univ. of California, Berkeley, CA. [CHIN*: 75, 91, 104]

[CHIN88a] Chin, D. N. (1988): User Models and Discourse Models. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 86–87. [RICH*: 39; SPAR*: 362]

[CHIN88b] Chin, D. N. (1988): Exploiting User Expertise in Answer Expression. In: Proc. AAAI-88, 756–760.

[CHIN*] Chin, D. N.: KNOME: Modeling What the User Knows in UC. In this volume. [WAHL*: 5; RICH*: 36, 51; KOBS*: 57, 60f., 65; QUIL*: 128, 130; CARB*:

160; PARI*: 229; MCCO*: 235; JAME*: 256, 278, 307; SPAR*: 362f.; MORI*: 375, 377; FINI*: 414]

[CLAN79] Clancey, W. J. (1979): Transfer of Rule-Based Expertise through a Tutorial Dialogue. Report STAN-CS-769, Computer Science Dept., Stanford Univ., Stanford, CA. [WAHL*: 7]

[CLAN81] Clancey, W. J. and R. Letsinger (1981): NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application to Teaching. In: Proc. IJCAI-81, 829–836. [KASS*: 398]

[CLAN82] Clancey, W. J. (1982): Tutoring Rules for Guiding a Case Method Dialogue. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 388f., 398]

[CLAN83] Clancey, W. J. (1983): The Epistemology of a Rule-Based System: A Framework for Explanation. Artificial Intelligence 20, 215–251. [KASS*: 398]

[CLAN86a] Clancey, W. J. (1986): From Guidon to Neomycin and Heracles in Twenty Short Lessons. In: AI Magazine 7, 40–60. [KASS*: 398]

[CLAN86b] Clancey, W. J. (1986): Qualitative Student Models. Annual Review of Computer Science 1, 381–450. [SPAR*: 360f.]

[CLHH81] Clark, H. H. and C. R. Marshall (1981): Definite Reference and Mutual Knowledge. In: A. K. Joshi, B. L. Webber and I. A. Sag, eds.: Elements of Discourse Understanding. Cambridge, MA: Cambridge Univ. Press. [KOBS*: 62; JAME*: 277]

[CLHH83] Clark, H. H., R. Schreuder and S. Buttrick (1983): Common Ground and the Understanding of Demonstrative Reference. Journal of Verbal Learning and Verbal Behavior 22, 245–258. [KOBS*: 62]

[CLKL78] Clark, K. L. (1978): Negation as Failure. In: J. Minker and H. Gallaire, eds.: Logic and Databases. New York: Plenum Press. [FINI*: 424]

[CLOW85] Clowes, I., I. Cole, F. Arshad, C. Hopkins and A. Hockley (1985): User Modelling Techniques for Interactive Systems. In: P. Johnson and S. Cook, eds.: People and Computers: Designing the Interface. Cambridge: Cambridge Univ. Press. [WAHL*: 9]

[COBO86] Cobourn, T. F. (1986): An Evaluation of Cleopatra, a Natural Language Interface for CAD. Master's Thesis, Dept. of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA. [LEHM*: 188]

[COHE78] Cohen, P. R. (1978): On Knowing What to Say: Planning Speech Acts. TR-118, Dept. of Computer Science, Univ. of Toronto, Canada. [WAHL*: 17]

[COHE79] Cohen, P. R. and C. R. Perrault (1979): Elements of a Plan-Based Theory of Speech Acts. Cognitive Science 3, 177–212. [KOBS*: 53, 59, 61]

[COHE81] Cohen, P. R., C. R. Perrault and J. F. Allen (1981): Beyond Question Answering. In: W. Lehnert and M. Ringle, eds.: Strategies for Natural Language Processing. Hillsdale, NJ: Lawrence Erlbaum. [CARB*: 157]

[COHE85] Cohen, R. (1985): The Need for Pragmatics in Natural Language Understanding. In: Proc. of the Theoretical Issues in Natural Language Understanding Workshop, Dalhousie Univ., Halifax, Nova Scotia. [COHE*: 326]

[COHE88] Cohen, R. (1988): On the Relationship between User Models and Discourse Models. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 88–90. [RICH*: 39; SPAR*: 362]

[COHE*] Cohen, R. and M. Jones: Incorporating User Models into Expert Systems for Educational Diagnosis. In this volume. [WAHL*: 19; CARB*: 161; PARI*: 231; MCCO*: 250; SPAR*: 362]

[COLB82] Colbourn (Jones), M. J. (1982): Computer-Guided Diagnosis of Learning Disabilities: A Prototype. M. ED. Thesis, Dept. for the Education of Exceptional Children, Univ. of Saskatchewan, Canada; also Report 82-8, Dept. of Computational Science, Univ. of Saskatchewan; also ERIC Report No. ED 222-032. [COHE*: 314]

[COLB83] Colbourn (Jones), M. J. and J. McLeod (1983): The Potential and Feasibility of an Expert System for Educational Diagnosis. In: Proc. of the IFIP 9th World Computer Congress, Paris, France, 891–896. [COHE*: 314]

[COLB84] Colbourn (Jones), M. J. and J. McLeod (1984): Computer-Guided Education Diagnosis: A Prototype Expert System. Journal of Special Education Technology 6, 30–39. [COHE*: 314]

[COLL62] Collier's Encyclopedia. New York: Crowell-Collier, 1962. [PARI*: 201, 205, 207]

[COMM71] Committee on Teacher Education and Professional Standards for Educators of Exceptional Children in Canada. The National Institute for Mental Retardation, Downswiew, Canada, 1971. [COHE*: 314]

[DANI86a] Daniels, P. J. (1986): Cognitive Models in Information Retrieval: An Evaluative Overview. Journal of Documentation 42, 217–328. [SPAR*: 360]

[DANI86b] Daniels, P. J. (1986): The User Modelling Function of an Intelligent Interface for Document Retrieval Systems. In: B. Brookes, ed.: Intelligent Information Systems for the Information Society. Amsterdam: North-Holland. [SPAR*: 363]

[DAVD82] Davidson, J. (1982): Natural Language Access to Data Bases: User Modelling and Focus. In: Proc. of the 4th CSCSI/SCEIO Conference, Saskatoon, Saskatchewan, Canada, 204–211.

[DAVI77] Davis, R. and B. G. Buchanan (1977): Meta-Level Knowledge: Overview and Applications. In: Proc. IJCAI-77, 920–927. [CHIN*: 98]

[DAVS84] Davison, A. (1984): Readibility Formulas and Comprehension. Comprehension Instruction, Perspectives and Suggestions. New York etc.: Longman. [PARI*: 201]

[DE86] De, S. (1986): Providing Effective Decision Support: Modelling Users and their Requirements. Decision Support Systems 2, 309–319.

[DEDE86] Dede, C. (1986): A Review and Synthesis of Recent Research in Intelligent Computer-Assisted Instruction. International Journal of Man-Machine Studies 24, 329–353. [WAHL*: 7; KOBS*: 54]

[DEKL86] DeKleer, J. (1986): An Assumption-Based TMS. Artificial Intelligence 28, 127–162. [RICH*: 47; FINI*: 429]

[DESM87] Desmarais, M. C. and M. Pavel (1987): User Knowledge Evaluation: An Experiment with UNIX. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North-Holland.

[DOLA85] Dolan, C. and M. Dyer (1985): Learning Planning Heuristics through Observation. In: Proc. IJCAI-85, 600–602. [QUIL*: 131]

[DOLA86] Dolan, C. and M. Dyer (1986): Encoding Planning Knowledge for Recognition, Construction, and Learning. In: Proc. of the 8th Annual Cognitive Science Society, Amherst, MA, 488–499. [QUIL*: 126, 131]

[DOYL79] Doyle, J. (1979): A Truth Maintenance System. Artificial Intelligence 12, 231–272. [WAHL*: 28; RICH*: 42, 47]

[DOYL80] Doyle, J. and P. London (1980): A Selected Descriptor-Indexed Bibliography to the Literature on Belief Revision. SIGART Newsletter 71, 7–23. [WAHL*: 28]

[DOYL83] Doyle, J. (1983): The Ins and Outs of Reason Maintenance. In: Proc. IJCAI-83, 352–354. [WAHL*: 28]

[DURH83] Durham, I., D. D. Lamb and J. B. Saxe (1983): Spelling Correction in User Interfaces. Communications of the ACM 26, 764–773. [LEHM*: 170]

[DYER83] Dyer, M. (1983): In-Depth Understanding: A Computer Model of Narrative Comprehension. Cambridge, MA: MIT Press. [QUIL*: 111f., 126]

[DYER88] Dyer, M., M. Flowers and J. F. Reeves (1988): Recognizing Situational Ironies: A Computer Model of Irony Recognition and Narrative Understanding. Advances in Computing and the Humanities 1 (in press). [QUIL*: 126]

[EAGL81] Eagly, A. H., S. Chaiken and W. Wood (1981): An Attribution Analysis of Persuasion. In: J. H. Harvey, W. Ickes and F. Kidd, eds.: New Directions in Attribution Research 3. Hillsdale, NJ: Erlbaum. [JAME*: 292]

[ENCY82] The New Encyclopedia of Science. Milwaukee, WI: Raintree Publishers, 1982. [PARI*: 201, 205]

[FAGI85] Fagin, R. and J. Y. Halpern (1985): Belief, Awareness and Limited Reasoning: Preliminary Report. In: Proc. IJCAI-85, 491–501. [WAHL*: 17]

[FAIN85] Fain, J., J. G. Carbonell, P. J. Hayes and S. N. Minton (1985): MULTIPAR: A Robust Entity-Oriented Parser. In: Proc. of the 7th Cognitive Science Society Conference, Irvine, CA, 110–119. [LEHM*: 164f.]

[FARR84] Farrell, R. G., J. R. Anderson and B. J. Reiser (1984): An Interactive Computer Based Tutor for LISP. In: Proc. AAAI-84, 106–109. [KASS*: 388]

[FIKE71] Fikes, R. E. and N. J. Nilsson (1971): STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2, 189–208. [QUIL*: 111; CARB*: 136]

[FINI83] Finin, T. W. (1983): Providing Help and Advice in Task Oriented Systems. Proc. IJCAI-83, 176–178. [WAHL*: 32; MORI*: 375; FINI*: 411, 415]

[FINI86] Finin, T. W. and D. Drager (1986): GUMS $_1$: A General User Modeling System. In: Proc. of the 6th Canadian Conference on Artificial Intelligence, Montreal, Canada, 24–29. [WAHL*: 28, 33; KOBS*: 57; COHE*: 330]

[FINI*] Finin, T. W.: GUMS: A General User Modeling Shell. In this volume. [WAHL*: 28, 33; KOBS*: 57, 65; CHIN*: 106; MCCO*: 235; JAME*: 256, 294]

[FLOW82] Flowers, M., R. McGuire and L. Birnbaum (1982): Adversary Arguments and the Logic of Personal Attacks. In: W. Lehnert and M. Ringle, eds.: Strategies for Natural Language Processing. Hillsdale, NJ: Lawrence Erlbaum. [QUIL*: 112]

[GAZD79] Gazdar, G. (1979): Pragmatics: Implicature, Presupposition, and Logical Form. New York: Academic Press. [JAME*: 261, 301]

[GENE82] Genesereth, M. (1982): The Role of Plans in Intelligent Teaching Systems. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [RICH*: 39]

[GENT83] Gentner, D. and A. L. Stevens (1983): Mental Models. Hillsdale, NJ: Lawrence Erlbaum. [WAHL*: 9]

[GEOR83] Georgeff, M. (1983): Communication and Interaction in Multi-Agent Planning Systems. Proc. AAAI-83, 125–129. [WAHL*: 7; KOBS*: 54]

[GERS81] Gershman, A. (1981): Finding Out What the User Wants: Steps Toward an Automated Yellow Pages Assistant. In: Proc. IJCAI-81, 423–425. [FINI*: 413]

[GILB87] Gilbert, G. N. (1987): Cognitive and Social Models of the User. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North-Holland.

[GINS86] Ginsberg, M. L. (1986): Multi-Valued Logics. In: Proc. AAAI-86, 243–247. [FINI*: 425]

[GOFF59] Goffman, E. (1959): The Presentation of Self in Everyday Life. Garden City, NY: Doubleday. [JAME*: 298]

[GOLD82] Goldstein, I. P. (1982): The Genetic Graph: A Representation for the Evolution of Procedural Knowledge. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 388, 400]

[GOOD84] Good, M. D., J. A. Whiteside, D. R. Wixon and J. J. Jones (1984): Building a User-Derived Interface. Communications of the ACM 27, 1032–1043. [LEHM*: 183, 188]

[GOOM86] Goodman, B. A. (1986): Miscommunication and Plan Recognition. Paper presented at the International Workshop on User Modeling, Maria Laach, Fed. Rep. of Germany. [QUIL*: 129; SPAR*: 362]

[GREE88] Green, T. R. G., F. Schiele and S. J. Payne (forthcoming): Formalizable Models of User Knowledge. In: T. R. G. Green, J.-M. Hoc, D. M. Murray and G. C. Van der Veer, eds.: Working with Computers: Theory versus Outcome. New York: Academic Press.

[GREI66] Greimas, A. J. (1966): Semantique Structurale: Recherche de Methode. Paris: Larousse. [MORI*: 382f.]

[GREW78] Grewendorf, G. (1978): Zur Semantik von Wertäußerungen. Germanistische Linguistik 2, 155. [MORI*: 377]

[GREW83] Grewendorf, G. (1983): What Answers Can be Given? In: F. Kiefer, ed.: Questions and Answers. Dordrecht, The Netherlands: Reidel. [JAME*: 262]

[GRIC57] Grice, H. P. (1957): Meaning. Philosophical Review 56, 377–388. [CARB*: 146]

[GRIC69] Grice, H. P. (1969): Utterer's Meaning and Intentions. Philosophical Review 68, 147–177. [CARB*: 146]

[GRIC75] Grice, H. P. (1975): Logic and Conversation. In: P. Cole and J. L. Morgan, eds.: Syntax and Semantics 3: Speech Acts. New York: Academic Press. [CARB*: 146, 156; PARI*: 201, 212; JAME*: 261, 272, 301; COHE*: 321]

[GRIM75] Grimes, J. E. (1975): The Thread of Discourse. The Hague, The Netherlands: Mouton. [PARI*: 206]

[GROS77a] Grosz, B. J. (1977): The Representation and Use of Focus in a System for Understanding Dialogs. Proc. IJCAI-77, 67–76. [CARB*: 135, 139]

[GROS77b] Grosz, B. J. (1977): The Representation and Use of Focus in Dialogue Understanding. Technical Report 151, SRI International, Menlo Park, CA. [MCCO*: 245]

[GROS81] Grosz, B. J. (1981): Focusing and Description in Natural Language Dialogues. In: B. Webber, A. Joshi and I. Sag, eds.: Elements of Discourse Understanding. Cambridge, England: Cambridge Univ. Press. [CARB*: 152; MCCO*: 244; MORI*: 375, 382]

[GROS83] Grosz, B. J., A. K. Joshi and S. Weinstein (1983): Providing a Unified Account of Definite Noun Phrases in Discourse. In: Proc. of the 21st Annual Meeting of the Association for Computational Linguistics, Cambridge, MA, 44–50. [MCCO*: 244]

[GROS85] Grosz, B. J. and C. Sidner (1985): Discourse Structure and the Proper Treatment of Interruptions. In: Proc. IJCAI-85, 832–839. [MCCO*: 252]

[GROS86] Grosz, B. J. and C. Sidner (1986): Attention, Intentions, and the Structure of Discourse. Computational Linguistics 12, 175–204. [MORI*: 376]

[HAAS86] Haas, A. R. (1986): A Syntactic Theory of Belief and Action. Artificial Intelligence 28, 245–292. [KOBS*: 58]

[HALA83] Halasz, F. G. and T. P. Moran (1983): Mental Models and Problem Solving in Using a Calculator. In: Proc. of the CHI'83 Conference on Human Factors in Computing Systems, Boston, MA, 212–216.

[HALP85a] Halpern, J. Y. and Y. O. Moses (1985): A Guide to the Modal Logics of Knowledge and Belief: Preliminary Draft. In: Proc. IJCAI-85, 480–490. [WAHL*: 17; KOBS*: 62, 67]

[HALP85b] Halpern, J. Y. and Y. O. Moses (1985): Knowledge and Common Knowledge in a Distributed Environment. Report RJ 4421, IBM Research Laboratory, San Jose, CA. [KOBS*: 54, 63]

[HALP86] Halpern, J. Y. (1986): Reasoning about Knowledge: An Overview. In: Proc. of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, 1–17. [KOBS*: 67]

[HAMI76] Hamilton, D. L. and R. K. Gifford (1976): Illusory Correlation in Interpersonal Perception: A Cognition Basis of Stereotypic Judgements. Journal of Experimental Social Psychology 12, 392–407. [RICH*: 36]

[HARE52] Hare, R. M. (1952): The Language of Morals. Oxford: Clarendon Press. [MORI*: 377]

[HARR63] Harrah, D. (1963): Communication: A Logical Model. Cambridge, MA: MIT Press. [JAME*: 262, 285]

[HAYE76] Hayes, P. J. and M. A. Rosner (1976): ULLY: A Program for Handling Conversations. In: Proc. of the 1976 AISB Summer Conference, Edinburgh, Scotland, 137–147. [WAHL*: 5, 18]

[HAYE84] Hayes, P. J. (1984): Entity-Oriented Parsing. In: Proc. of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Stanford Univ., Stanford, CA, 212–217. [LEHM*: 165]

[HECK88] Hecking, M., C. Kemke, E. Nessen, D. Dengler, M. Gutmann and G. Hector (1988): The SINIX Consultant: A Progress Report. Memo No. 28, Dept. of Computer Science, Univ. of Saarbrücken, Fed. Rep. of Germany. [WAHL*: 5]

[HEIN85] Heines, J. M. and T. O'Shea (1985): The Design of a Rule-Based CAI Tutorial. International Journal of Man-Machine Studies 23, 1–26. [KASS*: 389]

[HERR87] Herrmann, T. (1987): Sprachpsychologische Beiträge zur Partnerbezogenheit des Sprechens. Research Group on Language and Cognition, Dept. of Psychology, Univ. of Mannheim, Fed. Rep. of Germany.

[HINT62] Hintikka, J. (1962): Knowledge and Belief. Ithaca, NY: Cornell Univ. Press. [WAHL*: 16]

[HINT75] Hintikka, J. (1975): Answers to Questions. In: H. Hiz, ed.: Questions. Dordrecht, The Netherlands: Reidel. [JAME*: 262]

[HOBB78a] Hobbs, J. (1978): Coherence and Coreference. Technical Note 168, SRI International, Menlo Park, CA. [PARI*: 206]

[HOBB78b] Hobbs, J. (1978): Why is a Discourse Coherent? Technical Report 176, SRI International, Menlo Park, CA. [PARI*: 206]

[HOBB80] Hobbs, J. and D. Evans (1980): Conversation as Planned Behavior. Cognitive Science 4, 349–377. [PARI*: 206]

[HOEP83a] Hoeppner, W. and K. Morik (1983): Was haben Hotels, Straßenkreuzungen und Fische gemeinsam? – Mit HAM-ANS spricht man darüber. Report ANS-20, Research Unit for Information Science and Artificial Intelligence, Univ. of Hamburg, Fed. Rep. of Germany. [KOBS*: 63]

[HOEP83b] Hoeppner, W., T. Christaller, H. Marburger, K. Morik, B. Nebel, M. O'Leary and W. Wahlster (1983): Beyond Domain-Independence: Experience with the Development of a German Language Access System to Highly Diverse Background Systems. In: Proc. IJCAI-83, 588–594. [WAHL*: 10, 30; SPAR*: 362; MORI*: 364, 370; FINI*: 415]

[HOEP84a] Hoeppner, W., S. Busemann, T. Christaller, H. Marburger, K. Morik and B. Nebel (1984): Dialoging HAM-ANS: Commented Terminal Sessions with a Natural Language System. Memo ANS-23, Research Unit for Information Science and Artificial Intelligence, Univ. of Hamburg, Fed. Rep. of Germany. [MORI*: 379]

[HOEP84b] Hoeppner, W., K. Morik and H. Marburger (1984): Talking it Over: The Natural Language Dialog System HAM-ANS. Technical Report ANS-26, Research Unit for Information Science and Artificial Intelligence, Univ. of Hamburg, Fed. Rep. of Germany. [CHIN*: 78; PARI*: 202]

[HOLL86] Holland, J. H., K. J. Holyoak, R. E. Nisbett and P. R. Thagard (1986): Induction: Processes of Inference, Learning, and Discovery. Cambridge, MA: MIT Press. [JAME*: 294]

[HORA87] Horacek, H., H. Kaindl and M. Wagner (1987): Probabilities in Game-Playing: Possible Meanings and Applications. In: B. Buchberger and J. Retti, eds.: 3. Österreichische Artificial-Intelligence-Tagung. Berlin: Springer. [WAHL*: 8]

[HORT87] Horton, D. L. (1987): Incorporating Agents' Beliefs in a Model of Presupposition. Technical Report CSRI-201, Computer Systems Research Institute, Univ. of Toronto, Canada.

[HORN78] Horn, L. R. (1978): Remarks on Neg-Raising. In: P. Cole, ed.: Syntax and Semantics 9: Pragmatics. New York etc.: Academic Press. [KOBS*: 57]

[HOVY86] Hovy, E. H. (1986): Putting Affect into Text. In: Proc. of the 8th Annual Conference of the Cognitive Science Society, Amherst, MA, 669–675. [JAME*: 271]

[HOWE84] Howe, A. and T. Finin (1984): Using Spreading Activation to Identify Relevant Help. In: Proc. of the 5th CSCSI/SCEIO Conference, London, Ontario, 25–27. [FINI*: 411]

[JAME80] Jameson, A., W. Hoeppner and W. Wahlster (1980): The Natural Language System HAM-RPM as a Hotel Manager: Some Representational Prerequisites. In: R. Wilhelm, ed.: GI – 10. Jahrestagung Saarbrücken. Berlin: Springer. [WAHL*: 5, 10; MORI*: 382]

[JAME82] Jameson, A. and W. Wahlster (1982): User Modelling in Anaphora Generation: Ellipsis and Definite Description. In: Proc. of the 1982 European Conference on Artificial Intelligence, Orsay, France, 222–227. [WAHL*: 22; JAME*: 274; MORI*: 371]

[JAME83] Jameson, A. (1983): Impression Monitoring in Evaluation-Oriented Dialog: The Role of the Listener's Assumed Expectations and Values in the Generation of Informative Statements. In: Proc. IJCAI-83, 616–620. [WAHL*: 25; JAME*: 258, 311; MORI*: 375]

[JAME84] Jameson, A. (1984): Interpreting and Predicting Belief Statements. In: Proc. of the 6th European Conference on Artificial Intelligence, Pisa, Italy, 419–422. [JAME*: 294]

[JAME86] Jameson, A. (1986): Some Psychological Literature of Relevance to User Modeling. Unpublished Memo, Psychological Laboratory, Univ. of Nijmegen, The Netherlands.

[JAME87] Jameson, A. (1987): How to Appear to be Conforming to the 'Maxims' even if You Prefer to Violate them. In: G. Kempen, ed.: Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics. Dordrecht, The Netherlands: Nijhoff. [JAME*: 311]

[JAME*] Jameson, A.: But What Will the Listener Think? Belief Ascription and Image Maintenance in Dialog. In this volume. [WAHL*: 14, 25; SPAR*: 361f.]

[JOHN84] Johnson, W. L. and E. Soloway (1984): Intention-Based Diagnostics of Programming Errors. In: Proc. AAAI-84, 162–168. [CHIN*: 77f., 100; KASS*: 393, 403; FINI*: 414]

[JOHN85] Johnson, W. L. (1985): Intention-Based Diagnosis of Errors in Novice Programs. Ph.D. Thesis, Dept. of Computer Science, Yale Univ., New Haven, CT. [KASS*: 403]

[JOHP83] Johnson-Laird, P. N. (1983): Mental Models. Cambridge: Cambridge Univ. Press.

[JONE65] Jones, E. E. and K. E. Davis (1965): From Acts to Dispositions: The Attribution Process in Person Perception. In: L. Berkowitz, ed.: Advances in Experimental Social Psychology 2. New York: Academic Press. [JAME*: 287]

[JONE86] Jones, M. J. and J. Tubman (1986): Expert Systems for Educational Diagnosis: An Experiment with THEORIST. In: Proc. of the 6th International Workshop on Expert Systems and their Applications, Avignon, France, 1265–1283. [COHE*: 314]

[JONE87] Jones, M. J., J. McLeod, G. Robertson, J. Tubman and B. Wasson (1987): The CGD Project: An Educational Diagnostic System Based on THEORIST. Technical Report CS-87-26, Dept. of Computer Science, Univ. of Waterloo, Canada. [COHE*: 314, 322]

[JOSH82] Joshi, A. K. (1982): Mutual Beliefs in Question-Answer Systems. In: N. V. Smith, ed.: Mutual Knowledge. London etc.: Academic Press. [KOBS*: 62; CARB*: 152, 158f.; JAME*: 256]

[JOSH84a] Joshi, A. K., B. Webber and R. M. Weischedel (1984): Living up to Expectations: Computing Expert Responses. In: Proc. AAAI-84, 169–175. [CHIN*: 104; QUIL*: 126; JAME*: 256; COHE*: 329; FINI*: 414]

[JOSH84b] Joshi, A. K., B. Webber and R. M. Weischedel (1984): Preventing False Inferences. In: Proc. of the 10 International Conference on Computational Linguistics, Stanford, CA, 134–138. [CHIN*: 104; FINI*: 414]

[JOYC83] Joyce, C. and L. Wingerson (1983): Can We Adjust to Computer Culture? In: New Scientist, April 1983, 72–73. [WAHL*: 14]

[KAPD69] Kaplan, D. (1969): Quantifying In. In: D. Davidson and J. Hintikka, eds.: Words and Objections: Essays on the Work of W. V. Quine. Dordrecht, The Netherlands: Reidel. [KOBS*: 58]

[KAPS79] Kaplan, S. J. (1979): Cooperative Responses from a Portable Natural Language Data Base Query System. Ph.D. Thesis, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia. Also available as: Report HPP-79-19, Computer Science Dept., Stanford Univ., Stanford, CA. [WAHL*: 19, 26; KOBS*: 58; CARB*: 146, 152; MCCO*: 237]

[KAPS82] Kaplan, S. J. (1982): Cooperative Responses from a Portable Natural Language Query System. Artificial Intelligence 19, 165–188. [WAHL*: 19, 26; PARI*: 201; FINI*: 416]

[KAPS83] Kaplan, S. J. (1983): Cooperative Responses from a Portable Natural Language Database Query System. In: M. Brady and R. C. Berwick, eds.: Computational Models of Discourse. Cambridge, MA: MIT Press. [WAHL*: 19, 26; KOBS*: 58; CHIN*: 100]

[KARL85] Karlin, R. (1985): Romper Mumbles. Technical Report 85-41, Univ. of Pennsylvania, Philadelphia, PA. [MCCO*: 236]

[KASS86] Kass, R. (1986): The Role of User Modelling in Intelligent Tutoring Systems. Technical Report MS-CIS-86-58, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA.

[KASS87a] Kass, R. (1987): Implicit Acquisition of User Models in Cooperative Advisory Systems. Technical Report MS-CIS-87-05, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA. [WAHL*: 12, 18, 33; PARI*: 229; KASS*: 409; FINI*: 416]

[KASS87b] Kass, R. and T. Finin (1987): Rules for the Implicit Acquisition of Knowledge about the User. In: Proc. AAAI-87, 295–300. [KASS*: 409; FINI*: 416]

[KASS87c] Kass, R. and T. Finin (1987): The Need for User Models in Generating Expert System Explanations. Technical Report MS-CIS-87-86, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA.

[KASS88a] Kass, R. (1988): Acquiring a Model of the User's Beliefs from a Cooperative Advisory Dialog. Ph.D. Thesis, Dept. of Computer and Information Science, University of Pennsylvania, PA.

[KASS88b] Kass, R. and T. Finin (1988): Modeling the User in Natural Language Systems. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), 5–22. [FINI*: 412]

[KASS88c] Kass, R. and T. Finin (1988): The Need for User Models in Generating Expert System Explanations. Technical Report MS-CIS-88-37, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA. [WAHL*: 5]

[KASS*] Kass, R.: Student Modeling in Intelligent Tutoring Systems: Implications for User Modeling. In this volume. [WAHL*: 7, 15; CHIN*: 77; FINI*: 412, 414]

[KAUT86] Kautz, H. and J. Allen (1986): Generalized Plan Recognition. In: Proc. AAAI-86, 423–427. [QUIL*: 129]

[KAYD85] Kay, D. S. and J. B. Black (1985): The Evolution of Knowledge Representations with Increasing Expertise in Using Systems. In: Proc. of the 7th Annual Conference of the Cognitive Science Society, Irvine, CA, 140–149. [CHIN*: 77]

[KAYM79] Kay, M. (1979): Functional Grammar. In: Proc. of the 5th Meeting of the Berkeley Linguistics Society, Berkeley, CA, 142–158. [PARI*: 223]

[KEMK86] Kemke, C. (1986): The SINIX Consultant: Requirements, Design, and Implementation of an Intelligent Help System for a UNIX Derivative. Report No. 11, AI Laboratory, Dept. of Computer Science, Univ. of Saarbrücken, Fed. Rep. of Germany. [WAHL*: 5, 32; QUIL*: 128]

[KIMB82] Kimball, R. (1982): A Self-Improving Tutor for Symbolic Integration. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 400]

[KOBS84] Kobsa, A. (1984): Three Steps in Constructing Mutual Belief Models from User Assertions. In: Proc. of the 6th European Conference on Artificial Intelligence, Pisa, Italy, 423–427. [WAHL*: 11; KOBS*: 62; FINI*: 415]

[KOBS85a] Kobsa, A. (1985): Benutzermodellierung in Dialogsystemen. Berlin: Springer. [WAHL*: 7, 13, 16ff., 32; KOBS*: 53, 57, 60f., 67; JAME*: 278; MORI*: 375, 382]

[KOBS85b] Kobsa, A. (1985): User Modelling in Dialogue Systems: Potentials and Hazards. Report 85-01, Dept. of Medical Cybernetics and Artificial Intelligence, Univ. of Vienna, Austria. [CHIN*: 85]

[KOBS85c] Kobsa, A. (1985): Using Situation Descriptions and Russellian Attitudes for Representing Beliefs and Wants. In: Proc. IJCAI-85, 513–515. [KOBS*: 58, 67]

[KOBS85d] Kobsa, A. (1985): VIE-DPM: A User Model in a Natural-Language Dialogue System. In: J. Laubsch, ed.: GWAI-84, 8th German Workshop on Artificial Intelligence. Berlin: Springer. [WAHL*: 11, 18; KOBS*: 53; CHIN*: 85]

[KOBS86a] Kobsa, A. (1986): Generating a User Model from Wh-Questions in the VIE-LANG System. In: P. Hellwig und H. Lehmann, eds.: Trends in der Linguistischen Datenverarbeitung. Hildesheim, Fed. Rep. of Germany etc.: Olms. [WAHL*: 11; KOBS*: 60; CHIN*: 85]

[KOBS86b] Kobsa, A., J. Allgayer, C. Reddig, N. Reithinger, D. Schmauks, K. Harbusch and W. Wahlster (1986): Combining Deictic Gestures and Natural Language for Referent Identification. In: Proc. of the 11th International Conference on Computational Linguistics, Bonn, Fed. Rep. of Germany, 356–361. [WAHL*: 5]

[KOBS88] Kobsa, A. (1988): User Models and Dialog Models: United They Stand... In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 91–95. [WAHL*: 18, 33; RICH*: 39; SPAR*: 362]

[KOBS89] Kobsa, A. (1989): User Modeling in Dialog Systems: Potentials and Hazards. In: Proc. of the IFIP/GI Conference on Opportunities and Risks of Artificial Intelligence Systems, Hamburg, Fed. Rep. of Germany (forthcoming).

[KOBS*] Kobsa A.: A Taxonomy of Beliefs and Goals for User Models in Dialog Systems. In this volume. [JAME*: 277; SPAR*: 360, 362; KASS*: 406]

[KOLO84] Kolodner, J. L. (1984): Retrieval and Organizational Strategies in Conceptual Memory. Hillsdale, NJ: Lawrence Erlbaum. [QUIL*: 125]

[KOLO86] Kolodner, J. L. and R. E. Cullingford (1986): Towards a Memory Architecture that Supports Reminding. In: Proc. of the 8th Annual Cognitive Science Society, Amherst, MA, 467–477. [QUIL*: 125]

[KONO80] Konolige, K. and N. J. Nilsson (1980): Multiple Agent Planning Systems. In: Proc. AAAI-80, 138–141. [WAHL*: 7; KOBS*: 54]

[KONO81] Konolige, K. (1981): A First-Order Formalisation of Knowledge and Action for a Multi-Agent Planning System. In: J. E. Hayes, D. Michie and Y.-H. Pao, eds.: Machine Intelligence 10. Chichester: Ellis Horwood. [WAHL*: 16; KOBS*: 58, 63]

[KONO84] Konolige, K. (1984): Belief and Incompleteness. Report CSLI-84-4, Center for the Study of Language and Information, Stanford, CA. [MORI*: 369]

[KONO85] Konolige, K. (1985): User Modelling, Common-Sense Reasoning and the Belief-Desire-Intention Paradigm. In: Proc. IJCAI-85, 1300.

[KOWA79] Kowalski, R. (1979): Logic for Problem Solving. New York: North-Holland. [FINI*: 424]

[KRAU83] Kraut, R. (1983): There are no *De Dicto* Attitudes. Synthese 54, 275–294. [KOBS*: 55]

[LANC87] Lancaster, J. S. and J. L. Kolodner (1987): Problem Solving in a Natural Task as a Function of Experience. Technical Report, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA. [PARI*: 211]

[LEBO83] Lebowitz, M. (1983): RESEARCHER: An Overview. In: Proc. AAAI-83, 232–235. [PARI*: 202]

[LEBO85] Lebowitz, M. (1985): RESEARCHER: An Experimental Intelligent Information System. In: Proc. IJCAI-85, 858–862. [PARI*: 202]

[LEBO86] Lebowitz, M. (1986): An Experiment in Intelligent Information Systems: RESEARCHER. In: R. Davies, ed.: Intelligent Library and Information Systems. London: Ellis Horwood. [PARI*: 202]

[LEHD84] Lehman, D. (1984): Knowledge, Common Knowledge and Related Puzzles. In: Proc. of the 3rd Annual ACM Conference on Principles of Distributed Computing, Vancouver, Canada, 62–67. [KOBS*: 54, 62]

[LEHM*] Lehman, J. F. and J. G. Carbonell: Learning the User's Language: A Step Towards Automated Creation of User Models. In this volume. [CHIN*: 107; CARB*: 160; SPAR*: 363; MORI*: 365]

[LEHN77] Lehnert, W. (1977): A Conceptual Theory of Question Answering. In: Proc. IJCAI-77, 158–164. [PARI*: 202]

[LEVE84] Levesque, H. (1984): A Logic of Implicit and Explicit Belief. In: Proc. AAAI-84, 198–202. [COHE*: 326]

[LEVI77] Levin, J. A. and J. A. Moore (1977): Dialogue-Games: Metacommunication Structures for Natural Language Interaction. Cognitive Science 1, 395–420. [WAHL*: 30]

[LEWI69] Lewis, D. K. (1969): Convention: A Philosophical Study. Cambridge: Harvard Univ. Press. [KOBS*: 62]

[LITM84] Litman, D. J. and J. F. Allen (1984): A Plan Recognition Model for Clarification Subdialogues. In: Proc. of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Stanford Univ., Stanford, CA, 302–311. [CHIN*: 78]

[LITM85] Litman, D. J. (1985): Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogue. Ph.D. Thesis, Report TR70, Dept. of Computer Science, Univ. of Rochester, NY. [SPAR*: 362; MORI*: 370, 374f., 382]

[LITM86] Litman, D. J. (1986): Linguistic Coherence: A Plan-Based Alternative. In: Proc. of the 24th Annual Meeting of the Association for Computational Linguistics, New York, NY, 215–223. [CARB*: 135]

[LNR74] The LNR Research Group (1974): MEMOD: A Model for the Representation of Language and Semantic Structures in Human Long-Term Memory. Unpublished Memo, Center for Human Interaction Processing, Univ. of California at San Diego, CA.

[LOND82] London, B. and W. J. Clancey (1982): Plan Recognition Strategies in Student Modeling: Prediction and Description. In: Proc. AAAI-82, 335–338. [KASS*: 391, 398]

[MACL82] Maclaran, R. (1982): Comments on Joshi's Paper. In: N. V. Smith, ed.: Mutual Knowledge. London: Academic Press. [JAME*: 256]

[MACM83] Macmillan, S. A. (1983): User Models to Personalize an Intelligent Agent. Ph.D. Thesis, School of Education, Stanford Univ., Stanford, CA. [CHIN*: 84, 106]

[MAND88] Mandl, H. and A. Lesgold, eds. (1988): Learning Issues for Intelligent Tutoring Systems. New York: Springer.

[MANN84] Mann, W. C. (1984): Discourse Structure for Text Generation. Technical Report ISI/RR-84-127, Information Sciences Institute, Marina del Rey, CA. [PARI*: 206]

[MARB83] Marburger, H. and W. Wahlster (1983): Case Role Filling as a Side Effect of Visual Search. In: Proc. of the 1st Conference of the European Chapter of the Association for Computational Linguistics, Pisa, Italy, 188–195. [WAHL*: 23; MORI*: 371]

[MARB87] Marburger, H. (1987): A Strategy for Producing Cooperative NL Reactions in a Database Interface. In: P. Jorrand and V. Sgurev, eds.: Artificial Intelligence II: Methodology, Systems, Applications. Amsterdam etc.: North-Holland. [WAHL*: 26; CHIN*: 101; MORI*: 370]

[MARB88] Marburger, H. (1988): Generierung kooperativer natürlichsprachlicher Antworten in einem Dialogsystem mit Zugang zu relationalen Datenbanken. Ph. D. Thesis, Dept. of Computer Science, Univ. of Saarbrücken, Fed. Rep. of Germany. [WAHL*: 26; CHIN*: 101]

[MARH87] Mark, H. F., D. F. Othmer, C. G. Overberger and G. T. Seaborg, eds.; Executive Editor: M. Grayson; Associate Editor: D. Eckroth (1987): Encyclopedia of Chemical Technology. New York: John Wiley and Sons. [PARI*: 213]

[MARJ85] Martin, J. H. (1985): Knowledge Acquisition through Natural Language Dialogue. In: Proc. of the 2nd IEEE Conference on Artificial Intelligence Applications, Miami, FL, 582–586. [CHIN*: 88]

[MARP83] Martin, P., D. Appelt and F. Pereira (1983): Transportability and Generality in a Natural-Language Interface System. In: Proc. IJCAI-83, 573–581. [WAHL*: 30]

[MARS83] Martins, J. P. (1983): Reasoning in Multiple Belief Spaces. Ph.D. Thesis, Dept. of Computer Science, State Univ. of New York at Buffalo, Amherst, NY. [KOBS*: 56, 59]

[MASS88] Masson, M. E. J., W. C. Hill, J. Conner and R. Guindon (1988): Misconceived Misconceptions? In: Proc. of the CHI'88 Conference on Human Factors in Computing Systems, Washington, D.C., 151–156.

[MAST89] Mastaglio, T. (1989): A User Modelling Approach for a Computer-Based Critic. Dissertation Research Proposal, Dept. of Computer Science, University of Colorado, Boulder, CO.

[MATT86] Matthews, M. and G. Biswas (1986): USCSH: An Active Assistance Interface for UNIX. Technical Report USC-CS 86-003, Dept. of Computer Science, Univ. of South Carolina, Columbia, SC. [CHIN*: 78]

[MATZ82] Matz, M. (1982): Towards a Process Model for High School Algebra. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 397]

[MAYS80a] Mays, E. (1980): Correcting Misconceptions about Database Structure. In: Proc. of the 3rd CSCSI/SCEIO Conference, Victoria, B.C., 123–128. [MCCO*: 237]

[MAYS80b] Mays, E. (1980): Failures in Natural Language Systems: Applications to Data Base Query Systems. In: Proc. AAAI-80, 327–330. [WAHL*: 26; CHIN*: 101; CARB*: 146]

[MCCA80] McCauley, C., C. L. Stitt and M. Segal (1980): Stereotyping: From Prejudice to Prediction. Psychological Bulletin 87, 195–208. [RICH*: 35]

[MCCA87] McCalla, G. I. and J. E. Greer (1987): The Practical Use of Artificial Intelligence in Automated Tutoring: Current Status and Impediments to Progress. Research Report 87-2, Dept. of Computational Science, Univ. of Saskatchewan, Saskatoon, Canada.

[MCCL78] McCawley, J. D. (1978): Conversational Implicature and the Lexicon. In: P. Cole, ed.: Syntax and Semantics 9: Pragmatics. New York: Academic Press. [JAME*: 261]

[MCCO83] McCoy, K. F. (1983): Correcting Misconceptions: What to Say When the User is Mistaken. In: Proc. of the CHI'83 Conference on Human Factors in Computing Systems, Boston, MA, 197–201. [PARI*: 201]

[MCCO85a] McCoy, K. F. (1985): Correcting Object-Related Misconceptions. Ph.D. Thesis, MS-CIS-85-57, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA. [WAHL*: 27; CHIN*: 101; MCCO*: 245; FINI*: 416]

[MCCO85b] McCoy, K. F. (1985): The Role of Perspective in Responding to Property Misconceptions. In: Proc. IJCAI-85, 791–793. [WAHL*: 27]

[MCCO86] McCoy, K. F. (1986): The ROMPER System: Responding to Object-Related Misconceptions Using Perspective. In: Proc. of the 24th Annual Meeting of the Association for Computational Linguistics, New York, NY, 97–105. [PARI*: 201f.; COHE*: 325, 330]

[MCCO87] McCoy, K. F. (1987): Contextual Effects on Responses to Misconceptions. In: G. Kempen, ed.: Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics. Dordrecht/Boston: Kluwer. [MCCO*: 242]

[MCCO88] McCoy, K. F. (1988): Reasoning on a Dynamically Highlighted User Model to Respond to Misconceptions. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), 52–63. [PARI*: 201f.; MCCO*: 233; SPAR*: 362]

[MCCO*] McCoy, K. F.: Highlighting a User Model to Respond to Misconceptions. In this volume. [WAHL*: 7, 27; CHIN*: 101; QUIL*: 126; CARB*: 152, 161; PARI*: 201; JAME*: 260; MORI*: 375; FINI*: 416]

[MCDO80] McDonald, D. D. (1980): Natural Language Production as a Process of Decision Making under Constraint. Ph.D. Thesis, Dept. of Computer Science and Electrical Engineering, MIT, Cambridge, MA. [MCCO*: 236]

[MCKE82] McKeown, K. R. (1982): Generating Natural Language Text in Response to Questions about Database Structure. Ph.D. Thesis, Univ. of Pennsylvania, Philadelphia, PA. [MCCO*: 243]

[MCKE85a] McKeown, K. R. (1985): Discourse Strategies for Generating Natural-Language Text. Artificial Intelligence 27, 1–41. [CARB*: 139, 155; FINI*: 415]

[MCKE85b] McKeown, K. R. (1985): Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text. Cambridge: Cambridge Univ. Press. [PARI*: 202f., 206, 211, 223; JAME*: 279; COHE*: 325]

[MCKE85c] McKeown, K. R., M. Wish and K. Matthews (1985): Tailoring Explanations for the User. In: Proc. IJCAI-85, 794–798. [WAHL*: 12; PARI*: 201, 203; MCCO*: 251; FINI*: 415]

[MCKE87] McKeown, K. R. and C. L. Paris (1987): Functional Unification Grammar Revisited. In: Proc. of the 25th Annual Meeting of the Association for Computational Linguistics, Stanford Univ., Stanford, CA, 97–103. [PARI*: 223]

[MCLE82] McLeod, J. (1982): Institute of Child Guidance and Development at the Univ. of Saskatchewan. Journal of Learning Disabilities 15, 290–293. [COHE*: 314]

[MCLE85] McLeod, J. and M. Jones (1985): The Development of an Expert System for Computer-Guided Diagnosis of Children's Learning Problems: Some Emerging Problems. In: Proc. of the International Conference on the Computer as an Aid for Those with Special Needs, Sheffield, England, 68–78. [COHE*: 314]

[MCLE86] McLeod, J., M. Jones, G. Robertson and B. Wasson (1986): Computer-Guided Diagnosis: A Progress Report on a Canadian Project. Fifth Canadian Symposium on Instructional Technology, Ottawa, Canada, 101–105. [COHE*: 314]

[MILL82] Miller, M. L. (1982): A Structured Planning and Debugging Environment for Elementary Programming. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 388]

[MINS75] Minsky, M. (1975): A Framework for Representing Knowledge. In: P. Winston, ed.: The Psychology of Computer Vision. New York: McGraw-Hill. [RICH*: 37]

[MINT85] Minton, S. N., P. J. Hayes and J. E. Fain (1985): Controlling Search in Flexible Parsing. In: Proc. IJCAI-85, 785–787. [LEHM*: 165]

[MITC86] Mitchell, T. M., R. M. Keller and S. T. Kedar-Cabelli (1986): Explanation-Based Learning: A Unifying View. Machine Learning 1, 47–80. [LEHM*: 193]

[MOOR80] Moore, R. C. (1980): Reasoning about Knowledge and Action. Technical Note 191, AI Center, SRI International, Menlo Park, CA. [WAHL*: 16, 18; KOBS*: 62]

[MOOR82] Moore, R. C. and G. G. Hendrix (1982): Computational Models of Belief and the Semantics of Belief Sentences. In: S. Peters and E. Saarinen, eds.: Processes, Beliefs, and Questions. Dordrecht etc.: Reidel. [KOBS*: 55]

[MORI82] Morik, K. (1982): Überzeugungssysteme der Künstlichen Intelligenz: Validierung vor dem Hintergrund linguistischer Theorien über implizite Äußerungen. Tübingen: Niemeyer. [MORI*: 372, 383]

[MORI84] Morik, K. (1984): Partnermodellierung und Interessenprofile bei Dialogsystemen der Künstlichen Intelligenz. In: C.-R. Rollinger, ed.: Probleme des (Text-) Verstehens: Ansätze der Künstlichen Intelligenz. Tübingen: Niemeyer. [WAHL*: 19; MORI*: 365]

[MORI85a] Morik, K. (1985): User Modeling, Dialog Structure, and Dialog Strategy in HAM-ANS. In: Proc. of the 2nd Conference of the European Chapter of the Association for Computational Linguistics, Geneva, Switzerland, 268–273. [WAHL*: 21; KOBS*: 53, 59; MORI*: 373, 375f., 379]

[MORI85b] Morik, K. and C.-R. Rollinger (1985): The Real Estate Agent: Modeling Users by Uncertain Reasoning. AI Magazine 6, 44–52. [WAHL*: 13, 21; KOBS*: 53; CHIN*: 78, 83; CARB*: 161; FINI*: 414ff.]

[MORI88] Morik, K. (1988): Discourse Models, Dialog Memories, and User Models. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 95–97. [RICH*: 39; SPAR*: 362; MORI*: 366]

[MORI*] Morik, K.: User Models and Conversational Settings: Modeling the User's Wants. In this volume. [WAHL*: 19; RICH*: 36; KOBS*: 53, 59; CHIN*: 78; MCCO*: 251; JAME*: 256, 278, 294; SPAR*: 343, 360, 362; FINI*: 415]

[MOTR86] Motro, A. (1986): Query Generalization: A Method for Interpreting Null Answers. In: L. Kerschberg, ed.: Expert Database Systems. Menlo Park, CA: Benjamin/Cummings. [FINI*: 418]

[MURR87a] Murray, D. M. (1987): A Survey of User Cognitive Modelling. Report DITC 92/87, National Physics Laboratory, Teddington, England. [WAHL*: 9]

[MURR87b] Murray, D. M. (1987): Embedded User Models. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North Holland. [WAHL*: 9]

[NADA83] Nadathur, G. and A. K. Joshi (1983): Mutual Beliefs in Conversational Systems: Their Role in Referring Expressions. In: Proc. IJCAI-83, 603–605. [KOBS*: 62]

[NEAL87] Neal, L. R. (1987): User Modeling for Syntax-Directed Editors. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North Holland.

[NESS87] Nessen, E. (1987): SC-UM: User Modeling in the SINIX Consultant. Memo No. 18, Dept. of Computer Science, Univ. of Saarbrücken, Fed. Rep. of Germany. [WAHL*: 5; CHIN*: 92; QUIL*: 128]

[NEUM83] Neumann, B. and H.-J. Novak (1983): Event Models for Recognition and Natural Language Description of Events in Real-World Image Sequences. In: Proc. IJCAI-83, 724–726. [MORI*: 370]

[NEW67] The New Book of Knowledge: The Children's Encyclopedia. New York: Grolier Inc., 1967. [PARI*: 201, 205]

[NILS83] Nilsson, N. (1983): Artificial Intelligence Prepares for 2001. The AI Magazine 4(4), 7–14. [WAHL*: 31]

[NIKU85] Nisbett, R. E. and Z. Kunda (1985): Perception of Social Distributions. Journal of Personality and Social Psychology 48, 297–311. [JAME*: 294, 308]

[NIRO80] Nisbett, R. E. and L. Ross (1980): Human Inference: Strategies and Shortcomings of Social Judgment. Englewood Cliffs, NJ: Prentice-Hall. [JAME*: 285]

[NORM86a] Norman, D. A. (1986): Cognitive Engineering. In: D. A. Norman and S. W. Draper, eds.: User Centered System Design: New Perspectives on Human-Computer Interaction. Hillsdale, NJ: Erlbaum. [JAME*: 255]

[NORM86b] Norman, D. A. and S. W. Draper, eds. (1986): User Centered System Design: New Perspectives on Human-Computer Interaction. Hillsdale, NJ: Erlbaum. [WAHL*: 9]

[NOTH84] Nothdurft, W. (1984): äh folgendes Problem äh: Die interaktive Ausarbeitung 'des Problems' in Beratungsgeprächen. In: R. Wimmer and G. Zifonun,

eds.: Forschungsberichte des Instituts für Deutsche Sprache. Tübingen: Narr. [MORI*: 372]

[NOVA84] Novak, H.-J. and B. Neumann (1984): Szenenbeschreibung und Imagination in NAOS. In: C.-R. Rollinger, ed.: Probleme des (Text-)Verstehens. Tübingen: Niemeyer. [WAHL*: 24]

[OSHE82] O'Shea, T. (1982): A Self-Improving Quadratic Tutor. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 400]

[PARI84] Paris, C. L. (1984): Determining the Level of Expertise. In: Proc. of the 1st Annual Workshop on Theoretical Issues in Conceptual Information Processing, Atlanta, GA, 128–135. [PARI*: 212, 229]

[PARI85a] Paris, C. L. (1985): Description Strategies for Naive and Expert Users. In: Proc. of the 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, IL, 238–246. [PARI*: 225; COHE*: 330]

[PARI85b] Paris, C. L. and T. Kwee (1985): Guide to the Unification Process and its Implementation: Progress Report on Extending the Grammar. Technical Report, Computer Science Dept., Columbia Univ., NY. [PARI*: 223]

[PARI87a] Paris, C. L. (1987): The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise. Ph.D. Thesis, Dept. of Computer Science, Columbia Univ., NY. [PARI*: 211, 223f.]

[PARI87b] Paris, C. L. and K. R. McKeown (1987): Discourse Strategies for Describing Complex Physical Objects. In: G. Kempen, ed.: Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology and Linguistics. Dordrecht/Boston: Kluwer. [PARI*: 224]

[PARI88] Paris, C. L. (1988): Tailoring Object Descriptions to a User's Level of Expertise. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), 64–78. [FINI*: 415]

[PARI*] Paris, C. L.: The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise. In this volume. [CHIN*: 107; QUIL*: 130; CARB*: 161; JAME*: 256, 260]

[PART70] Partee, B. H. (1970): The Semantics of Belief-Sentences. In: J. Hintikka, J. Moravcsik and P. Suppes, eds.: Approaches to Natural Languages. Dordrecht etc.: Reidel. [KOBS*: 57]

[PERR78] Perrault, C. R., J. Allen and P. Cohen (1978): Speech Acts as a Basis for Understanding Dialogue Coherence. In: Proc. of Theoretical Issues in Natural Language Processing 2, Urbana-Champaign, IL, 125–132. [COHE*: 329]

[PERR80] Perrault, C. R. and J. F. Allen (1980): A Plan-Based Analysis of Indirect Speech Acts. American Journal of Computational Linguistics 6, 167–182. [WAHL*: 19; CARB*: 135]

[PERR81] Perrault, C. R. and P. R. Cohen (1981): It's for your Own Good: A Note on Inaccurate Reference. In: A. K. Joshi, B. Webber and I. A. Sag, eds.: Elements of Discourse Understanding. Cambridge: Cambridge Univ. Press. [KOBS*: 62]

[PERR88] Perrault, C. R. (1988): An Application of Default Logic to Speech Act Theory. In: P. Cohen, J. Morgan and M. Pollack, eds.: Plans and Intentions in Communication and Discourse. Cambridge, MA: MIT Press. [FINI*: 416]

[POLL82a] Pollack, M. E., J. Hirschberg and B. Webber (1982): User Participation in the Reasoning Process of Expert Systems. MS CIS-82-9, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA. [WAHL*: 5]

[POLL82b] Pollack, M. E., J. Hirschberg and B. Webber (1982): User Participation in the Reasoning Process of Expert Systems. In: Proc. AAAI-82, 358–361. [CARB*: 152]

[POLL83] Pollack, M. E. (1983): Generating Expert Answers through Goal Inference. Technical Note 316, SRI International, Stanford, CA. [MORI*: 372]

[POLL85] Pollack, M. E. (1985): Information Sought and Information Provided. In: Proc. of the CHI'85 Conference on Human Factors in Computing Systems, San Francisco, CA, 155–160. [FINI*: 418]

[POLL86a] Pollack, M. E. (1986): A Model of Plan Inference that Distinguishes between the Beliefs of Actors and Observers. In: Proc. of the 24th Annual Meeting of the Association for Computational Linguistics, New York, NY, 207–214. [WAHL*: 12; QUIL*: 127; CARB*: 136, 143, 153; SPAR*: 362]

[POLL86b] Pollack, M. E. (1986): Inferring Domain Plans in Question-Answering. Ph.D. Thesis, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA. [WAHL*: 12; QUIL*: 112, 127]

[POLL87] Pollack, M. E. (1987): Some Requirements for a Model of the Plan-Inference Process in Conversation. In: R. Reilly, ed.: Communication Failure in Dialogue. Amsterdam etc.: North Holland. [CARB*: 151ff.]

[POLS88] Polson, M. and J. Richardson, eds. (1988): Foundations of Intelligent Tutoring Systems. Hillsdale, NJ: Lawrence Erlbaum.

[POOL86] Poole, D. L., R. Aleliunas and R. Goebel (1986): Theorist: A Logical Reasoning System for Default and Diagnosis. Technical Report CS-86-06, Dept. of Computer Science, Univ. of Waterloo, Canada. [COHE*: 322]

[POWE79] Power, R. (1979): The Organisation of Purposeful Dialogues. Linguistics 17, 107–152. [WAHL*: 7; KOBS*: 54]

[PRAT87] Pratt, J. M. (1987): The Social Impact of User Models. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North Holland.

[PSOT88] Psotka, J., D. Massey and S. Mutter, eds. (1988): Intelligent Tutoring Systems: Lessons Learned. Hillsdale, NJ: Lawrence Erlbaum.

[QUIL85] Quilici, A. (1985): Human Problem Understanding and Advice Giving: A Computer Model. Technical Report 85-69, Computer Science Dept., Univ. of California, Los Angeles, CA. [QUIL*: 113]

[QUIL86] Quilici, A., M. Dyer and M. Flowers (1986): AQUA: An Intelligent UNIX Advisor. In: Proc. of the 7th European Conference on Artificial Intelligence, Brighton, England, 33–38.

[QUIL88] Quilici, A., M. Dyer and M. Flowers (1988): Recognizing and Responding to Plan-Oriented Misconceptions. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), 38–51. [QUIL*: 108; PARI*: 201]

[QUIL89] Quilici, A. (1989): The Correction Machine: Formulating Explanations for User Misconceptions. Proc. IJCAI-89 (forthcoming).

[QUIL*] Quilici, A.: Detecting and Responding to Plan-Oriented Misconceptions. In this volume. [WAHL*: 7, 27; RICH*: 39; CHIN*: 101; CARB*: 152, 161; PARI*: 201; MCCO*: 238; JAME*: 260, 307]

[QUIN56] Quine, W. V. O. (1956): Quantifiers and Propositional Attitudes. Journal of Philosophy 53, 177–187. Reprinted in: Quine, W. V. O. (1966): The Ways of Paradox. New York: Random House. [KOBS*: 55]

[QUIN60] Quine, W. V. O. (1960): Word and Object. Cambridge, MA: MIT Press.

[QUIN86] Quinn, L. and D. M. Russell (1986): Intelligent Interfaces: User Models and Planners. In: Proc. of the CHI'86 Conference on Human Factors in Computing Systems, Boston, MA, 314–320. [WAHL*: 33]

[RAMS87] Ramshaw, L. A. (1987): Pragmatic Knowledge for Resolving Ill-Formedness. Technical Report, Dept. of Computer and Information Sciences, Univ. of Delaware, Newark, DE. [CARB*: 161]

[RAPA84] Rapaport, W. J. and S. C. Shapiro (1984): Quasi-Indexial Reference in Propositional Semantic Networks. In: Proc. of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Stanford Univ., Stanford, CA, 65–70. [KOBS*: 55]

[REIC78] Reichman, R. (1978): Conversational Coherency. Cognitive Science 2, 283–327. [JAME*: 279]

[REIS85] Reiser, B. J., J. R. Anderson and R. G. Farrell (1985): Dynamic Student Modelling in an Intelligent Tutor for LISP Programming. In: Proc. IJCAI-85, 8–14. [KOBS*: 56; CHIN*: 77, 100; KASS*: 388f., 393]

[REIT78] Reiter, R. (1978): On Closed World Data Bases. In: H. Gallaire and J. Minker, eds.: Logic and Data Bases. New York and London: Plenum Press. [CHIN*: 97; CARB*: 151; FINI*: 424]

[REIT80] Reiter, R. (1980): A Logic for Default Reasoning. Artificial Intelligence 13, 81–132. [RICH*: 41; FINI*: 416]

[RICH79a] Rich, E. (1979): Building and Exploiting User Models. Ph.D. Thesis, Computer Science Dept., Carnegie-Mellon Univ., Pittsburgh, PA. [WAHL*: 5, 11, 13f.; KOBS*: 53, 67; COHE*: 329; MORI*: 373–377]

[RICH79b] Rich, E. (1979): User Modelling Via Stereotypes. Cognitive Science 3, 329–354. [WAHL*: 5, 11, 13f.; RICH*: 36; KOBS*: 53, 67; CHIN*: 78f., 83; PARI*: 201, 229; SPAR*: 342, 362; FINI*: 414ff.]

[RICH82] Rich, E. (1982): Programs as Data for their Help Systems. In: Proc. of the National Computer Conference, Houston, TX, 481–485. [WAHL*: 32]

[RICH83] Rich, E. (1983): Users are Individuals: Individualizing User Models. In: International Journal of Man-Machine Studies 18, 199–214. [WAHL*: 5, 20; RICH*: 38; CHIN*: 78f., 89; SPAR*: 360; FINI*: 414]

[RICH84] Rich, E. (1984): Natural Language Interfaces. Computer 17, 39–47. [LEHM*: 173]

[RICH*] Rich, E.: Stereotypes and User Modeling. In this volume. [WAHL*: 13; CHIN*: 78f.; PARI*: 201; MCCO*: 235; JAME*: 256, 294; COHE*: 329; MORI*: 376; FINI*: 416, 422]

[RIVE89] Rivers, R. (1989): Embedded User Models: Where Next? Interacting with Computers 1, 14–30.

[ROBI80] Robinson, A. E., D. E. Appelt, B. J. Grosz, G. G. Hendrix and J. J. Robinson (1980): Interpreting Natural Language Utterances in Dialogs about Tasks. Technical Report TR210, SRI International, Menlo Park, CA. [CARB*: 135]

[ROBI81] Robinson, A. E. (1981): Determing Verb Phrase Referents in Dialogs. American Journal of Computational Linguistics 7, 1–18. [CARB*: 135]

[ROLL80] Rollinger, C. R. (1980): Readtime-Inferenzen für semantische Netze. In: C. R. Rollinger and H.-J. Schneider, eds.: Inferenzen in natürlichsprachlichen Systemen der Künstlichen Intelligenz. Berlin: Einhorn.

[ROSC77] Rosch, E. (1977): Human Categorization. In: N. Warren, ed.: Studies in Cross Cultural Psychology I. London: Academic Press. [CHIN*: 79, 105]

[ROSC78] Rosch, E. (1978): Principles of Categorization. In: E. Rosch and B. B. Lloyd, eds.: Cognition and Categorization. Hillsdale, NJ: Lawrence Erlbaum. [CHIN*: 79, 105]

[ROSC83] Rosch, E. (1983): Prototype Classification and Logical Classification: The Two Systems. In: E. Scholnick, ed.: New Trends in Cognitive Representation: Challenges to Piaget's Theory. Hillsdale, NJ: Erlbaum. [CHIN*: 79, 105]

[ROSE82] Rosenschein, J. S. (1982): Synchronization of Multi-Agent Plans. In: Proc. AAAI-82, 115–119. [WAHL*: 7; KOBS*: 54]

[ROSE85] Rosenschein, J. S. (1985): Rational Interaction: Cooperation among Intelligent Agents. Ph.D. Thesis, Stanford Univ., CA. [WAHL*: 7; KOBS*: 54]

[ROSN81] Rosner, M. (1981): Some Thoughts about Actions in Conversation. In: Proc. of the Workshop on Models of Dialogue: Theory and Application, Linköping, Sweden, 105–116. [WAHL*: 26, 32]

[ROSS85] Ross, P., J. Jones and M. Millington (1985): User Modeling in Command-Driven Systems. DAI Research Paper No. 264, Dept. of Artificial Intelligence, Univ. of Edinburgh, Scotland.

[ROWE84] Rowe, N. C. (1984): Modeling Degrees of Item Interest for a General Database Query System. International Journal of Man-Machine Studies 20, 421–443. [JAME*: 264]

[RUME73] Rumelhart, D. E. and A. A. Abrahamson (1973): A Model for Analogical Reasoning. Cognitive Psychology 5, 1–28. [MCCO*: 243]

[RUME86] Rumelhart, D. E., J. L. McClelland and the PDP Research Group (1986): Parallel Distributed Processing. Cambridge, MA: MIT Press. [RICH*: 50]

[SACE74] Sacerdoti, E. (1974): Planning in a Hierarchy of Abstraction Spaces. Artificial Intelligence 5, 115–135. [QUIL*: 111]

[SACE77] Sacerdoti, E. (1977): A Structure for Plans and Behavior. Amsterdam: North-Holland. [KASS*: 394]

[SAMP82] Sampson, G. (1982): The Economics of Conversation: Comments on Joshi's Paper. In: N. V. Smith, ed.: Mutual Knowledge. London: Academic Press. [JAME*: 259]

[SCHG79] Schank, G. (1979): Zum Ablaufmuster von Kurzberatungen: Beschreibung einer Gesprächsstruktur. In: J. Dittmann, ed.: Arbeiten zur Konversationsanalyse. Tübingen: Niemeyer [MORI*: 372]

[SCHI72] Schiffer, S. R. (1972): Meaning. Oxford: Clarendon Press. [KOBS*: 62]

[SCHM78] Schmidt, C. F., N. S. Sridharan and J. L. Goodson (1978): The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence. Artificial Intelligence 11, 45–83. [CARB*: 152f.]

[SCHR77] Schank, R. C. and R. Abelson (1977): Scripts, Plans, Goals, and Understanding. Hillsdale, NJ: Lawrence Erlbaum. [QUIL*: 111]

[SCHR82] Schank, R. C. (1982): Dynamic Memory. Cambridge, MA: Cambridge Univ. Press. [QUIL*: 125]

[SCHR86] Schank, R. C. (1986): Explanation Patterns: Understanding Mechanically and Creatively. Hillsdale, NJ: Lawrence Erlbaum. [QUIL*: 111]

[SCHU83a] Schuster, E. (1983): Custom-Made Responses: Maintaining and Updating the User Model. MS-CIS-83-13, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA.

[SCHU83b] Schuster, E. and T. W. Finin (1983): Understanding Misunderstandings. MS-CIS-83-12, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA. [WAHL*: 7]

[SCHU85] Schuster, E. (1985): Grammars as User Models. In: Proc. IJCAI-85, 20–22. [LEHM*: 164]

[SCHU86a] Schuster, E. (1986): The Role of Native Grammars in Correcting Errors in Second Language Learning. Computational Intelligence 2, 93–98.

[SCHU86b] Schuster, E. and T. Finin (1986): VP2: The Role of User Modeling in Correcting Errors in Second Language Learning. In: A. G. Cohn and J. R. Thomas, eds.: Artificial Intelligence and its Applications. Chichester etc.: Wiley. [FINI*: 411, 418]

[SCHU88] Schuster, E. (1988): Establishing the Relationship between Discourse Models and User Models. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 82–85. [RICH*: 39; SPAR*: 362]

[SCHW79] Schwitalla, J. (1979): Nonresponsive Antworten. Deutsche Sprache 7, 193–211. [JAME*: 279]

[SEAR69] Searle, J. R. (1969): Speech Acts: An Essay in the Philosophy of Language. Cambridge, England: Cambridge Univ. Press. [KOBS*: 61; CHIN*: 85]

[SELF74] Self, J. A. (1974): Student Models in Computer-Aided Instruction. International Journal of Man-Machine Studies 6, 261–276. [KASS*: 387]

[SELF88] Self, J. A. (1988): Bypassing the Intractable Problem of Student Modelling. In: ITS-88: Intelligent Tutoring Systems, Montreal, Canada, 18–24.

[SHAF76] Shafer, G. (1976): A Mathematical Theory of Evidence. Princeton, NJ: Princeton Univ. Press. [RICH*: 47]

[SHAN79] Shanon, B. (1979): Where Questions. In: Proc. of the 17th Annual Meeting of the Association Computational Linguistics, La Jolla, CA, 73–75.

[SHEP26] Sheperd, H. R. (1926): The Fine Art of Writing. New York: Macmillan. [PARI*: 206]

[SHNE87] Shneiderman, B. (1987): Designing the User Interface: Strategies for Effective Human-Computer Interaction. Reading, MA: Addison-Wesley. [JAME*: 257, 308]

[SHOR76] Shortliffe, E. H. (1976): Computer-Based Medical Consultation: MYCIN. New York: Elsevier. [RICH*: 47; COHE*: 329]

[SHRA82] Shrager, J. and T. Finin (1982): An Expert System that Volunteers Advice. In: Proc. AAAI-82, 339–340. [KASS*: 388; FINI*: 411]

[SIDN81a] Sidner, C. L. (1981): Focusing for Interpretation of Pronouns. American Journal of Computational Linguistics 7, 217–231. [CARB*: 139, 155]

[SIDN81b] Sidner, C. L. and D. J. Israel (1981): Recognizing Intended Meaning and Speakers' Plans. In: Proc. IJCAI-81, 203–208. [FINI*: 415]

[SIDN83a] Sidner, C. L. (1983): Focusing in the Comprehension of Definite Anaphora. In: M. Brady and R. Berwick, eds.: Computational Models of Discourse. Cambridge, MA: MIT Press. [MCCO*: 244]

[SIDN83b] Sidner, C. L. (1983): What the Speaker Means: The Recognition of Speaker's Plans in Discourse. Computers and Mathematics with Applications 9, 71–82. [WAHL*: 12; CARB*: 135, 156]

[SIDN85] Sidner, C. L. (1985): Plan Parsing for Intended Response Recognition of Speaker's Plans in Discourse. Computational Intelligence 1, 1–10. [CARB*: 135]

[SIKL78] Siklossy, L. (1978): Impertinent Question-Answering Systems: Justification and Theory. In: Proc. of the 1978 ACM National Conference, Washington, D.C., 39–44. [JAME*: 279]

[SIMM84] Simmons, R. F. (1984): From Menus to Intentions in Man-Machine Dialogue. Technical Report TR-84-33, AI-TR-84-04, Dept. of Computer Sciences, Univ. of Texas at Austin, TX.

[SLAT86] Slator, B. M., M. P. Anderson and W. Conley (1986): Pygmalion at the Interface. Communications of the ACM 29, 499–514. [LEHM*: 184]

[SLEE77] Sleeman, D. H. (1977): A System which Allows Students to Explore Algorithms. In: Proc. IJCAI-77, 780–786. [KASS*: 388]

[SLEE81] Sleeman, D. H. and M. J. Smith (1981): Modeling Student's Problem Solving. Artificial Intelligence 16, 171–187. [CHIN*: 77, 100; KASS*: 392, 396]

[SLEE82a] Sleeman, D. H. (1982): Assessing Aspects of Competence in Basic Algebra. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 396; FINI*: 414]

[SLEE82b] Sleeman, D. H. (1982): Inferring (Mal) Rules From Pupil's Protocols. In: Proc. of the 6th European Conference on Artificial Intelligence, Orsay, France, 160–164. [PARI*: 211; MCCO*: 237]

[SLEE82c] Sleeman, D. H. and J. S. Brown (1982): Intelligent Tutoring Systems. New York: Academic Press. [WAHL*: 7; KOBS*: 54; CHIN*: 77; COHE*: 329; SPAR*: 360; KASS*: 387]

[SLEE82d] Sleeman, D. H. and R. J. Hendley (1982): ACE: A System which Analyzes Complex Explanations. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [KASS*: 388]

[SLEE83] Sleeman, D. H. (1983): A Rule Directed Modeling System. In: R. Michalski, J. Carbonell and T. M. Mitchell, eds.: Machine Learning. Los Altos, CA: Tioga Press. [KASS*: 393, 396]

[SLEE84] Sleeman, D. H. (1984): An Attempt to Understand Students' Understanding of Basic Algebra. Cognitive Science 8, 387–412. [KASS*: 396f.]

[SLEE85] Sleeman, D. H. (1985): UMFE: A User Modeling Front End Subsystem. International Journal of Man-Machine Studies 23, 71–88. [WAHL*: 15; RICH*: 36; KOBS*: 53, 56f.; PARI*: 229; SPAR*: 361; KASS*: 408; FINI*: 416]

[SMIB82] Smith, B. (1982): Reflection and Semantics in a Procedural Language. Ph.D. Thesis, Technical Report MIT/LCS/TR-272, MIT, Cambridge, MA. [KOBS*: 62; FINI*: 413]

[SMID83] Smith, D. E. and M. R. Genesereth (1983): Finding All of the Solutions to a Problem: Serious Applications of Meta-Level Reasoning II. Memo HPP-83-21, Stanford Univ. Heuristic Programming Project, Stanford, CA. [CHIN*: 98]

[SMIN82] Smith, N. V. (1982): Mutual Knowledge. London etc.: Academic Press.

[SNYD79] Snyder, M. (1979): Self-Monitoring Processes. In: L. Berkowitz, ed.: Advances in Experimental Social Psychology 12. New York: Academic Press. [JAME*: 298]

[SOUL84] Soulhi, S. (1984): Representing Knowledge about Knowledge and Mutual Knowledge. In: Proc. of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Stanford Univ., Stanford, CA, 194–199. [WAHL*: 18]

[SOWA76] Sowa, J. F. (1976): Conceptual Graphs for a Database Interface. IBM Journal of Research and Development 20, 336–357. [CARB*: 146]

[SPAR84] Sparck Jones, K. (1984): User Models and Expert Systems. Technical Report No. 61, Computer Laboratory, Univ. of Cambridge, England. [WAHL*: 5, 20; KOBS*: 61; SPAR*: 344, 350, 352; MORI*: 366f., 374]

[SPAR86] Sparck Jones, K. (1986): Issues in User Modeling for Expert Systems. In: A. Cohn and J. Thomas, eds.: Artificial Intelligence and its Applications. Chichester etc.: Wiley. [RICH*: 39; SPAR*: 350]

[SPAR88] Sparck Jones, K. (1988): User Models, Discourse Models, and Some Others. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 98–100. [RICH*: 39; SPAR*: 362]

[SPAR*] Sparck Jones, K.: Realism about User Modeling. In this volume. [WAHL*: 20; KOBS*: 61; JAME*: 294; MORI*: 374]

[SPOH85] Spohrer, J. C., E. Soloway and E. Pope (1985): A Goal/Plan Analysis of Buggy Pascal Programs. Technical Report YALEU/CSD/RR 392, Dept. of Computer Science, Yale Univ., New Haven, CT. [KASS*: 403]

[SPOH89] Spohrer, J. C. and E. Soloway (1989): Simulating Student Programmers. Proc. IJCAI-89 (forthcoming).

[SPRE87] Sprenger, M. (1987): Konzeption des Partnermodells in WISBER. Technical Report No. 19, Project WISBER, Computer Science Dept., Univ. of Hamburg, Fed. Rep. of Germany. [WAHL*: 13]

[SPRE88] Sprenger, M. (1988): Interpretation von Modalverben zur Konstruktion von Partnermodelleinträgen. Memo No. 18, Project WISBER, Computer Science Dept., Univ. of Hamburg, Fed. Rep. of Germany. [WAHL*: 13]

[SRID85] Sridharan, N. S. (1985): User Modelling and Plan Recognition. In: Proc. IJCAI-85, 1301–1302.

[STEV79] Stevens, A., A. Collins and S. E. Goldin (1982): Misconceptions in Student's Understanding. International Journal of Man-Machine Studies 11, 145–156. [CHIN*: 77, 100; MCCO*: 237]

[STEV80] Stevens, A. and A. Collins (1980): Multiple Conceptual Models of a Complex System. In: P.-A. Federico, R. E. Snow and W. E. Montague, eds.: Aptitude, Learning, and Instruction. Hillsdale, NJ: Erlbaum. [MCCO*: 237]

[STEV82] Stevens, A., A. Collins and S. E. Goldin (1982): Misconceptions in Student's Understanding. In: D. H. Sleeman and J. S. Brown, eds.: Intelligent Tutoring Systems. New York: Academic Press. [QUIL*: 116, 125]

[STIC83] Stich, S. P. (1983): From Folk Psychology to Cognitive Science: The Case against Belief. Cambridge, MA: MIT Press. [KOBS*: 54f.]

[STRE86] Streitz, N. A. (1986): Mental Models and Methaphors: Implications for the Design of Adaptive User-System Interfaces. In: H. Mandl and A. Lesgold, eds.: Learning Issues for Intelligent Tutoring Systems. New York: Springer.

[SUTC87] Sutcliffe, A. G. and A. C. Old (1987): Do Users Know They Have User Models? Some Experiences in the Practise of User Modeling. In: H.-J. Bullinger and B. Shackel, eds.: Human-Computer Interaction: INTERACT'87. Amsterdam: North Holland.

[SWAR83] Swartout, W. R. (1983): XPLAIN: A System for Creating and Explaining Expert Consulting Programs. Artificial Intelligence 21, 285–325. [COHE*: 325, 329; FINI*: 415]

[TANG89] Tang, H., N. Major and R. Rivers (1989): From Users to Dialogues: Enabling Authors to Build an Adaptive, Intelligent System. In: Proc. of the 5th Con-

ference of British Computer Society Humen-Computer Interaction Specialist Group, Nottingham, England (forthcoming).

[TAUB88] Tauber, M. J. (1988): On Mental Models and the User Interface. In: T. R. G. Green, J.-M. Hoc, D. M. Murray and G. C. Van der Veer, eds.: Working with Computers: Theory versus Outcome. New York: Academic Press.

[TAYL78] Taylor, S. E., S. T. Fiske, N. L. Etcoff and A. J. Rudderman (1978): Categorical and Contextual Bases of Person Memory and Stereotyping. Journal of Personality and Social Psychology 36, 778–793. [RICH*: 36]

[TENN78] Tennant, H. (1978): The Evaluation of Natural Language Question Answerers. Technical Report, Univ. of Illinois at Urbana-Champaign, IL. [PARI*: 203]

[TETL85] Tetlock, P. E. and A. S. R. Manstead (1985): Impression Management versus Intrapsychic Explanations in Social Psychology: A Useful Dichotomy? Psychological Review 92, 59–77. [JAME*: 298]

[THMA83] Thomas, J. C. (1983): Psychological Issues in the Design of Database Query Languages. In: M. E. Sime and M. J. Coombs, eds.: Designing for Human-Computer Communication. London: Academic Press. [JAME*: 257, 276]

[THOM80] Thompson, B. H. (1980): Linguistic Analysis of Natural Language Communication with Computers. In: Proc. IJCAI-80, 190–201. [CARB*: 145]

[TOU82] Tou, F., M. Williams, R. Fikes, A. Henderson and T. Malone (1982): Rabbit: An Intelligent Database Assistant. In: Proc. AAAI-82, 314–317. [MCCO*: 245]

[TOUR86] Touretzky, D. (1986): The Mathematics of Inheritance Systems. Palo Alto, CA: Morgan Kaufmann. [RICH*: 43]

[TREN86] Trenouth, J. (1986): A Review and Classification of User Modelling Systems and Methods. KBET Research Report, Computer Science Dept., Univ. of Exeter, England.

[TREN88] Trenouth, J. (1988): A Pragmatic Architecture for Embedded User Modelling: "U.M. in K.B.E.T." Research Report No. 156, Computer Science Dept., Univ. of Exeter, England.

[TUBM86] Tubman, J. (1986): An Expert System for Educational Diagnosis Using THEORIST. M. Math. Thesis, Technical Report CS-86-32, Dept. of Computer Science, Univ. of Waterloo, Canada. [COHE*: 314, 332]

[TVER77] Tversky, A. (1977): Features of Similarity. Psychological Review 84, 327–352. [MCCO*: 243]

[VANA87] Van Arragon, P. (1987): User Modeling Bibliography. Research Report CS-87-22, Dept. of Computer Science, Univ. of Waterloo, Canada.

[VANB86a] van Beek, P. (1986): A Model for User Specific Explanation from Expert Systems. M. Math. Thesis, Technical Report CS-86-42, Dept. of Computer Science, Univ. of Waterloo, Canada. [COHE*: 326; SPAR*: 361f.]

[VANB86b] van Beek, P. and R. Cohen (1986): Towards User Specific Explanation Systems. In: Proc. of the 6th Canadian Conference on Artificial Intelligence, Montreal, Canada, 194–198. [COHE*: 326, 332; SPAR*: 361f.]

[VANB87] van Beek, P. (1987): A Model for Generating Better Explanations. In: Proc. of the 25th Annual Meeting of the Association for Computational Linguistics, Stanford, CA, 215–220.

[VANL88] VanLehn, K. (1988): Student Modeling. In: M. C. Polson and J. J. Richardson, eds.: Intelligent Tutoring Systems. Hillsdale, NJ: Lawrence Erlbaum.

[VAND81] van Dijk, T. A. (1981): Studies in the Pragmatics of Discourse. The Hague: Mouton. [JAME*: 285]

[VARD85] Vardi, M. Y. (1985): A Model-Theoretic Analysis of Monotonic Knowledge. In: Proc. IJCAI-85, 509–512. [WAHL*: 17]

[VERW62] Verwiebe, F. L., G. R. van Hooft and R. R. Suchy (1962): Physics: A Basic Science. Princeton, NJ: van Nostrand. [PARI*: 205]

[WAHL83] Wahlster, W., H. Marburger, A. Jameson and S. Busemann (1983): Over-Answering Yes-No-Questions: Extended Responses in a NL Interface to a Vision System. In: Proc. IJCAI-83, 643–646. [WAHL*: 19, 23; MORI*: 370, 375, 383]

[WAHL84] Wahlster, W. (1984): Cooperative Access Systems. Future Generation Computer Systems 1, 103–111. [WAHL*: 5]

[WAHL86a] Wahlster, W. (1986): Some Terminological Remarks on User Modeling. Paper presented at the International Workshop on User Modeling, Maria Laach, Fed. Rep. of Germany. [SPAR*: 360; MORI*: 366]

[WAHL86b] Wahlster, W. and A. Kobsa (1986): Dialog-Based User Models. In: Proc. of the IEEE-74 (Special Issue on Natural Language Processing), 948–960. [WAHL*: 4]

[WAHL88] Wahlster, W. (1988): Distinguishing User Models from Discourse Models. In: Computational Linguistics 14(3), Special Issue on User Modeling (A. Kobsa and W. Wahlster, eds.), Discussion Section, 101–103. [WAHL*: 6; RICH*: 39; SPAR*: 362]

[WAHL89] Wahlster, W. (1989): User and Discourse Models for Multimodal Communication. In: J. W. Sullivan and S. W. Tyler, eds.: Architectures for Intelligent Interfaces: Elements and Prototypes. Reading, MA: Addison-Wesley (forthcoming).

[WAHL*] Wahlster, W. and A. Kobsa: User Models in Dialog Systems. In this volume. [KOBS*: 52, 61, 67; CHIN*: 78; COHE*: 330; SPAR*: 360, 362; KASS*: 407]

[WALK87] Walker, A. (1987): Knowledge Systems and Prolog. Reading, MA: Addison Wesley. [FINI*: 428]

[WALL82] Wallis, J. W. and E. H. Shortliffe (1982): Explanatory Power for Medical Expert Systems: Studies in the Representation of Causal Relationships for Clinical Consultation. Technical Report STAN-CS-82-923, Heuristic Programming Project, Dept. of Medicine and Computer Science, Stanford Univ., Stanford, CA. [PARI*: 201f., 211; COHE*: 329; FINI*: 415]

[WALL84] Wallis, J. W. and E. H. Shortliffe (1984): Customized Explanations Using Causal Knowledge. In: B. G. Buchanan and E. H. Shortliffe, eds.: Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley. [KOBS*: 56]

[WALS66] Walster, E., E. Aronson and D. Abrams (1966): On Increasing the Persuasiveness of a Low Prestige Communicator. Journal of Experimental Social Psychology 2, 325–342. [JAME*: 292]

[WARD74] Ward, C. D. and E. McGinnies (1974): Persuasive Effects of Early and Late Mention of Credible and Noncredible Sources. Journal of Psychology 86, 17–23. [JAME*: 287]

[WASS83] Wasserman, K. and M. Lebowitz (1983): Representing Complex Physical Objects. Cognition and Brain Theory 6, 333–352. [PARI*: 222]

[WASS85] Wasserman, K. (1985): Unifying Representation and Generalization: Understanding Hierarchically Structured Objects. Ph.D. Thesis, Dept. of Computer Science, Columbia Univ., NY. [PARI*: 222]

[WEBB83] Webber, B. L. and E. Mays (1983): Varieties of User Misconceptions: Detection and Correction. In: Proc. IJCAI-83, 650–652. [WAHL*: 26; CHIN*: 101]

[WEBB84] Webber, B. L. and T. Finin (1984): In Response: Next Steps in Natural Language Interaction. In: W. Reitman, ed.: Artificial Intelligence Applications for Business. Norwood, NJ: Ablex Publ. Co. [FINI*: 411, 415, 418]

[WEBB86] Webber, B. L. (1986): Questions, Answers and Responses: Interacting with Knowledge Base Systems. In: M. Brodie and J. Mylopoulos, eds.: On Knowledge Base Management Systems. Berlin etc.: Springer. [CARB*: 158]

[WEIN84] Weiner, E. J. (1984): A Knowledge Representation Approach to Understanding Metaphors. Computational Linguistics 19, 1–14. [MCCO*: 243f., 248]

[WEIS73] Weissler, A. and P. Weissler (1973): A Woman's Guide to Fixing the Car. New York: Walker and Company. [PARI*: 201, 205]

[WENG87] Wenger, E. (1987): Artificial Intelligence and Tutoring Systems. Los Altos, CA: Morgan Kaufmann.

[WERN88] Werner, E. (1988): Toward a Theory of Communication and Cooperation for Multiagent Planning. In: Proc. of the 1988 Conference on Theoretical As-

pects of Reasoning about Knowledge, Monterey, CA (to appear). [WAHL*: 7; KOBS*: 54]

[WEYD79] Weyd, H. (1979): "Immerhin". In: H. Weyd, ed.: Die Partikeln der deutschen Sprache. Berlin: de Gruyter. [JAME*: 280]

[WILE83] Wilensky, R. (1983): Planning and Understanding. Reading, MA: Addison-Wesley. [QUIL*: 111]

[WILE84] Wilensky, R., Y. Arens and D. N. Chin (1984): Talking to UNIX in English: An Overview of UC. In: Communications of the ACM 27, 574–593. [WAHL*: 5, 19, 32; CHIN*: 74; QUIL*: 128]

[WILE86] Wilensky, R., J. Mayfield, A. Albert, D. N. Chin, C. Cox, M. Luria, J. Martin and D. Wu (1986): UC: A Progress Report. Technical Report UCB/CSD 87/303, Division of Computer Science, Univ. of California, Berkeley, CA. [WAHL*: 5, 19, 32; KOBS*: 60, 65; CHIN*: 74, 78, 82, 87f., 103; QUIL*: 128f.; SPAR*: 362; FINI*: 414f.]

[WILE87] Wilensky, R. (1987): Some Problems and Proposals for Knowledge Representation. Report No. UCB/CSD 87/351, Division of Computer Science, Univ. of California, Berkeley, CA. [CHIN*: 79, 105]

[WILK83] Wilks, Y. and J. Bien (1983): Beliefs, Points of View and Multiple Environments. Cognitive Science 7, 95–119. [WAHL*: 29, 32; SPAR*: 362]

[WILK87] Wilks, Y. and A. Ballim (1987): Multiple Agents and the Ascription of Belief. In: Proc. IJCAI-87, 119–124. [WAHL*: 32]

[WILL84] Williams, M. D. (1984): What Makes RABBIT Run. International Journal of Man-Machines Studies 21, 333–352. [RICH*: 36]

[WILS86] Wilson, P. T. and R. C. Anderson (1986): What They Don't Know Will Hurt Them: The Role of Prior Knowledge in Comprehension. In: J. Orasanu, ed.: Reading Comprehension: From Research to Practice. Hillsdale, NJ: Erlbaum. [PARI*: 201, 212]

[WINO77] Winograd, T. (1977): A Framework for Understanding Discourse. In: P. A. Carpenter and M. A. Just, eds.: Cognitive Processes in Comprehension. Hillsdale, NJ: Erlbaum. [JAME*: 261]

[WINO86] Winograd, T. and F. Flores (1986): Understanding Computers and Cognition: A New Foundation for Design. Norwood, NJ: Ablex. [JAME*: 256]

[WOOD73] Woods, W. (1973): An Experimental Parsing System for Transition Network Grammars. In: R. Rustin, ed.: Natural Language Processing. New York : Algorithmics Press. [PARI*: 223]

[WOOL83] Woolf, B. and D. McDonald (1983): Human-Computer Discourse in the Design of a Pascal Tutor. In: Proc. of the CHI'83 Conference on Human Factors in Computing Systems, Boston, MA, 230–234. [MCCO*: 237]

[WOOL84a] Woolf, B. (1984): Context Dependent Planning in a Machine Tutor. Ph.D. Thesis, Univ. of Massachusetts, Amherst, MA. [MCCO*: 237]

[WOOL84b] Woolf, B. and D. D. McDonald (1984): Building a Computer Tutor: Design Issues. Computer 17, 61–73. [KASS*: 388]

[WYER74] Wyer, R. S., Jr. (1974): Cognitive Organization and Change: An Information-Processing Approach. Hillsdale, NJ: Erlbaum. [JAME*: 269]

[YOUN88] Young, R. M. and A. MacLean (1988): Choosing between Methods: Analysing the User's Decision Space in Terms of Schemas and Linear Models. In: Proc. of the CHI'88 Conference on Human Factors in Computing Systems, Washington, D.C., 139–143.

[ZADE65] Zadeh, L. A. (1965): Fuzzy Sets. Information and Control 8, 338–353. [CHIN*: 83]

[ZADE82] Zadeh, L. A. (1982): A Computational Approach to Fuzzy Quantifiers in Natural Language. Memorandum UCB/ERL M82/36 (Revised), Division of Computer Science, Univ. of California, Berkeley, CA. [CHIN*: 83]

[ZILL82] Zillig, W. (1982): Bewerten: Sprechakttypen der bewertenden Rede. Tübingen: Niemeyer. [MORI*: 377]

[ZISS85] Zissos, A. Y. and I. H. Witten (1985): User Modeling for a Computer Coach: A Case Study. International Journal of Man-Machine Studies 23, 729–750. [KASS*: 391)]

Subject Index

acceptability SPAR*: 347, 349ff., 356f.
acquisition of user models WAHL*: 10–14; CHIN*: 83f., 86–89, 91–94; QUIL*: 129; MCCO*: 235, 254; I4: 337, 339; FINI*: 416
– system-driven vs. user driven MORI*: 375
– system-driven MORI*: 374, 377, 379
– user-driven MORI*: 374, 383, 385
ACT* KASS*: 404
action-achieving dialog, *see* dialog, action-achieving
active path CARB*: 138, 140
actor model, *see* agent model
adaptive parsing, *see* parsing, adaptive
addition change type MORI*: 373
advice WAHL*: 5, 27; I2: 70f.; CHIN*: 74; QUIL*: 108, 126; JAME*: 267, 294; I4: 338; KASS*: 400, 407
agent WAHL*: 6f., 20; KOBS*: 54, 56, 60; CHIN*: 75, 104; I4: 336; SPAR*: 343f.
– definition by Sparck Jones I4: 337
– model WAHL*: 7, 20; COHE*: 330f.; I4: 337; MORI*: 366; KASS*: 406
– modeled KOBS*: 54, 58
– modeling KOBS*: 54, 58; I4: 337
– role SPAR*: 343f., 346–354, 356, 360f.
analogy MCCO*: 241
answer CARB*: 158
– expression CHIN*: 103
anticipation feedback loop WAHL*: 22f., 25f.; I3: 198; JAME*: 274; KASS*: 407
– global WAHL*: 24
– local WAHL*: 22
applicability conditions CARB*: 136
apprentice-expert dialog, *see* dialog, apprentice-expert
appropriate query assumption CARB*: 143, 151, 153
AQUA WAHL*: 27; I2: 71; CHIN*: 101; QUIL*: 128
ARGOT WAHL*: 12, 19, 32; KOBS*: 53, 57, 59f., 63–66; I2: 72
assumptions about the user WAHL*: 5f., 9, 11–15, 20, 27f., 32; I2: 70–73; CHIN*: 94, COHE*: 313; I4: 336ff.; KASS*: 408
ATMS RICH*: 47
attitude change JAME*: 287, 292
attribute salience MCCO*: 243–249, 251, 253
attribution of beliefs and goals KOBS*: 53f.

background COHE*: 313–317, 319–328, 331ff.
backward chaining FINI*: 428
Bayes' theorem RICH*: 47
Bayesian inference JAME*: 293, 303
belief
– discrepancy MCCO*: 235
– report KOBS*: 55, 57
– representation WAHL*: 7, 15ff.; KOBS*: 53, 58, 65, 67; QUIL*: 111–115; KASS*: 406
– revision WAHL*: 28, 33; JAME*: 285, 287; COHE*: 325–328
– types I1: 3; KOBS*: 53f., 56ff., 60–65, 67; CHIN*: 82
– verification QUIL*: 121, 131
beliefs WAHL*: 4f., 7–13, 16f., 26f., 29–33; KOBS*: 52ff., 57–65; I2: 70ff.; QUIL*: 109–112, 126, 129; CARB*: 136, 144ff., 151ff., 157; I3: 197; PARI*: 201, 206, 212; MCCO*: 236; I4: 339; SPAR*: 356, 362; MORI*: 376; KASS*: 391, 407, 409; FINI*: 415
– basic KOBS*: 56f., 59, 61
– collective WAHL*: 4, 29
– complex WAHL*: 32; KOBS*: 54, 56f., 60–63; JAME*: 277, 293
– contradictory WAHL*: 7; I2: 72; QUIL*: 115–121
– domain-specific QUIL*: 115, 123f.
– embedded, *see* beliefs, complex
– infinite-reflexive WAHL*: 7, 11; KOBS*: 54, 61ff.

beliefs
– inconsistent I4: 339
– meta- KOBS*: 61; I2: 71
– mutual WAHL*: 7, 11, 18, 31; KOBS*: 54, 61–64; CARB*: 152; JAME*: 277, 293f.
– saturated KOBS*: 56–59
– unsaturated KOBS*: 58
beneficiary SPAR*: 348
BGP-MS WAHL*: 18, 33; I4: 339
bias JAME*: 269, 309
– negative JAME*: 270, 291f.
black box KASS*: 388, 395
BORIS QUIL*: 112
bug library KASS*: 392, 400, 403f.
BUGGY KASS*: 386, 394f.
bugs KASS*: 392, 395, 403ff.

canonical user, *see* user, canonical
canonical user model, *see* user model, canonical
capabilities FINI*: 415
categories CHIN*: 79
certainty measures RICH*: 43, 46ff.
CGD COHE*: 313–317, 322f., 325f., 331f.
CHAMP I2: 73; LEHM*: 170
change types MORI*: 372–375
classification of stereotypes, *see* stereotypes, classification
closed world assumption CHIN*: 97f.; QUIL*: 121; CARB*: 151; I4: 339; FINI*: 424
CO-OP WAHL*: 19, 26; CHIN*: 100f.; MCCO*: 237f.
coherence MORI*: 382; PARI*: 203, 206
comment, in evaluation-oriented dialog
– definition JAME*: 261
– desirability JAME*: 262, 266f., 270f., 279, 282, 300
– representation in IMP JAME*: 264
communication module KASS*: 388
computer-aided instruction KASS*: 387
conflict resolution RICH*: 42f.
constituency schema I3: 196; PARI*: 205–208, 216–221, 228
consultation WAHL*: 5, 19f., 29; KOBS*: 59; CHIN*: 75, 77f., 83f., 89, 91, 105; QUIL*: 108; COHE*: 314–317; MORI*: 372
contents of user models I1: 3; KOBS*: 52–53; PARI*: 203; I4: 339
context KOBS*: 58, 60; CHIN*: 88, 90; MCCO*: 234, 244
context model CARB*: 134f., 138–146, 148ff., 152f., 155–161; COHE*: 313
– enriched CARB*: 152, 157, 160; COHE*: 313
contextual effects MCCO*: 234f., 242ff.
contextual information MCCO*: 236
contradiction FINI*: 429
– resolution I1: 3; RICH*: 41
conversational
– implicature JAME*: 272, 301
– roles WAHL*: 5, 18; CHIN*: 88; MORI*: 366f., 370, 375
– – in evaluation-oriented dialog JAME*: 258, 278
– setting WAHL*: 4f., 18, 30; I4: 337; MORI*: 367, 370f., 375, 383; MORI*: 367, 375, 383
cooperation SPAR*: 361
cooperative dialog CARB*: 137, 139, 152
cooperative response MCCO*: 235
cooperative system FINI*: 414
cooperativeness WAHL*: 4ff., 8, 18f., 28f.; I2: 70f.; CHIN*: 100; QUIL*: 109; PARI*: 212; MCCO*: 235; JAME*: 256, 261, 272
correction
– schemas MCCO*: 252
– strategies MCCO*: 234, 240
– response CARB*: 161

database WAHL*: 6, 9, 26, 29; PARI*: 203; MCCO*: 233–237; FINI*: 412, 418
– completed FINI*: 424
– deductive FINI*: 417
– distributed KOBS*: 63, 65
– query system FINI*: 418
de dicto KOBS*: 55, 57, 59, 61
de re KOBS*: 55
DEBUGGY KASS*: 392, 394ff.
decision properties JAME*: 294; I4: 336f.; SPAR*: 345f., 348–354, 356ff., 360
deep model KASS*: 395, 409
default
– assumptions WAHL*: 10; RICH*: 35, 45; COHE*: 317–323, 326f., 330–333
– inferences CHIN*: 76ff., 81, 88
– reasoning I1: 3; RICH*: 35, 41, 43, 47; CARB*: 153, 156, 158; I4: 339; FINI*: 411f., 416
– rules FINI*: 420
definitions in a stereotype RICH*: 45; FINI*: 421
Dempster-Shafer model RICH*: 47
description of beliefs and goals KOBS*: 54f., 57–62, 67
destination of modeling information SPAR*: 347
development change type MORI*: 374
deviation LEHM*: 165, 168, 170, 177f.
diagnostic model KASS*: 394, 406

dialog CARB*: 134, 137, 144, 150; MCCO*: 236, 242, 244ff.; SPAR*: 353f., 358, 361
– action-achieving MORI*: 368, 371, 376
– apprentice-expert, *see* dialog, expert-novice
– cooperative CARB*: 137, 139, 152
– contributions WAHL*: 6, 11–14, 19, 26, 32; KOBS*: 63ff.; I2: 70, 72; I3: 198
– expert-novice CARB*: 135, 139, 152
– evaluation-oriented I3: 198, JAME*: 258
– focus MCCO*: 236
– goals CARB*: 155
– information-seeking CARB*: 133–139, 159; MORI*: 367, 370, 375
– markers CARB*: 136, 156
– memory I4: 337; MORI*: 365f., 370
– mixed-initiative WAHL*: 5
– model RICH*: 39; SPAR*: 362; MORI*: 366, 376
– planning I1: 3; WAHL*: 6, 19–24, 31f.; KOBS*: 53, 60–65; I3: 196f.; I4: 336
– strategies PARI*: 200f., 206, 212, 221, 223, 230f.; COHE*: 313
– structure MORI*: 379
– task-oriented WAHL*: 5, 13; JAME*: 269
– update- MORI*: 368
differential modeling KASS*: 390–393, 398ff., 408
difficulty levels CHIN*: 79f., 87, 106
discourse, *see* dialog
disparate models CARB*: 158
disparate plans, *see* plans, disparate
domain
– goals CARB*: 136f.
– highlighting MCCO*: 248
– knowledge PARI*: 200–206, 211ff., 215f., 229, 231; COHE*: 313; FINI*: 415
– model MCCO*: 240, 248f.
– plans CARB*: 136
double stereotypes, *see* stereotypes, double
dynamic SPAR*: 344
– plan inference, *see* plan inference, dynamic
– property SPAR*: 344f., 348–351, 354, 361

educational diagnosis COHE*: 313–320, 322f., 329–333
effectiveness SPAR*: 346, 349ff., 354, 356f.
efficiency SPAR*: 347, 349ff., 356
effort, in evaluation-oriented dialog
– consequences for pragmatic interpretation JAME*: 285, 301f.
– contributing factors JAME*: 262, 279
– representation in IMP JAME*: 264
– rule in selection of comments JAME*: 261f., 267
ellipsis
– intersentential CARB*: 134, 145
– generation WAHL*: 22
– resolution WAHL*: 22
error messages JAME*: 257
evaluation criteria MORI*: 377–380, 382
evaluation standards
– complex beliefs concerning JAME*: 293
– definition JAME*: 261
– evidence concerning JAME*: 278, 293
– examples used by IMP JAME*: 294
– representation in IMP JAME*: 262
– uncertainty in JAME*: 293
evaluator JAME*: 258, 278
evidential reasoning FINI*: 416
expert PARI*: 200–206, 211ff., 223, 229; COHE*: 313
– model KASS*: 388–92, 395, 398–401, 404, 408
– module KASS*: 388
– systems WAHL*: 7; PARI*: 231; MCCO*: 233ff.; COHE*: 313–321, 325f., 329–333; I4: 340; KASS*: 388, 408; FINI*: 418
expertise WAHL*: 5, 15, 27; KOBS*: 52; I2: 70; CHIN*: 74f., 78–85, 90–94, 97, 106; CARB*: 160f.; I3: 196; PARI*: 200–206, 211–223, 228–231; COHE*: 313; I4: 337
– levels CHIN*: 83
– local PARI*: 214
explanation COHE*: 313–318, 320f., 325f., 329–333; FINI*: 415
– for mistaken belief QUIL*: 110, 115
– how to find QUIL*: 120f.
– organization QUIL*: 123
– patterns QUIL*: 115
– in story understanding QUIL*: 111
– verification of QUIL*: 116, 121
expressive power WAHL*: 14, 16ff.; KOBS*: 53, 65

failure as negation FINI*: 420, 427
flexibility
– increments LEHM*: 168
– level LEHM*: 169
focus MCCO*: 236, 244
– of attention CARB*: 137f., 140, 142ff., 146, 150, 155f., 158
– of dialog, *see* dialog focus
– of question JAME*: 278, 295

focused goal
– candidate CARB*: 137f., 140ff.
– current CARB*: 138ff., 147
focused plan
– candidate CARB*: 138–144
– current CARB*: 138–144
focusing CARB*: 139
– constraints CARB*: 139
– heuristics CARB*: 137, 139, 142ff., 150, 153, 156, 158
foreground MORI*: 382
forms LEHM*: 188
frames RICH*: 37
fuzzy logic CHIN*: 83

game playing WAHL*: 7, 8
generalization hierarchy RICH*: 37, 40f., 44, 46; CARB*: 150
generation system PARI*: 201ff., 206, 211f., 213, 221, 226, 230f.; MCCO*: 239
genetic graph KASS*: 401f.
Geometry Tutor KASS*: 393, 403ff., 409
glass box KASS*: 388
goal
– analysis CHIN*: 88, 103, 105
– recognition KOBS*: 64
– representation WAHL*: 17f.; KOBS*: 53, 65, 67; KASS*: 403–406
– types I1: 3; KOBS*: 53f., 59f., 63f., 67
goals WAHL*: 4f., 9–13, 16f., 20, 25ff., 29, 32f.; KOBS*: 53, 57, 59ff., 64f.; I2: 70ff.; CHIN*: 78, 88; QUIL*: 114, 116–120, 126, 129; PARI*: 201, 203, 206, 211, 230; MCCO*: 233, 236, 251; COHE*: 313–318, 320ff., 326ff., 330–333; I4: 339; MORI*: 376f., 382, 385; KASS*: 403f., 407, 409, 414
– basic KOBS*: 57, 59ff.
– complex KOBS*: 54, 57, 63f.
– embedded, *see* goals, complex
– non-primitive CARB*: 136
– saturated KOBS*: 59
– unsaturated KOBS*: 60
grammatical deviance (deviation) LEHM*: 163ff., 167
GREATERP KASS*: 388f., 393, 403ff., 409
GRUNDY WAHL*: 5, 11, 13f., 19f.; RICH*: 36, 41, 48; KOBS*: 53; CHIN*: 78, 80, 83; PARI*: 229; MORI*: 374f., 377; FINI*: 416
GUIDON KASS*: 386, 388f., 391, 394, 398ff., 402
GUMS WAHL*: 28, 33; KOBS*: 65f.; CHIN*: 106; I4: 339; FINI*: 412, 420

habitable interface, *see* interface, habitable
HAM-ANS WAHL*: 5, 19–22, 26, 30; RICH*: 36; KOBS*: 53, 63, 65f.; CHIN*: 78, 101; PARI*: 202; I4: 337f.; MORI*: 364f., 367–379, 383
help systems WAHL*: 7; QUIL*: 108, 126ff.; PARI*: 231; JAME*: 257, 307; FINI*: 411, 415
highlighted user model, *see* user model, highlighted
human-computer interaction WAHL*: 9; JAME*: 255, 307
hypothesize-and-revise paradigm CARB*: 152f.; COHE*: 313

ideal and buggy model KASS*: 404
ideal student, *see* student, ideal
IDEBUGGY KASS*: 394
idiosyncratic language LEHM*: 163, 165, 170, 176f., 183, 185
if-detected daemons CHIN*: 91
image maintenance JAME*: 275f., 280, 283, 285, 298, 301, 304–307, 310
IMP WAHL*: 20, 22, 25; I3: 198; JAME*: 258, 308f., 311
– behavior in employment interview situation JAME*: 258, 267f., 270, 273, 276, 283, 289f., 296f., 301
– plausibility of performance JAME*: 269
impasse KASS*: 405
implicature, *see* conversational implicature
impression
– change JAME*: 262, 264, 280, 282
– definition JAME*: 262
– empirical research JAME*: 269
– expected value JAME*: 264
– graphic representation JAME*: 265, 309
– uncertainty in JAME*: 262, 264
imprecise utterances CARB*: 151
indirect speech acts WAHL*: 6, 19; I2: 71; CHIN*: 85, 88, 103
informant JAME*: 258
information integration theory JAME*: 269
information-seeking dialog, *see* dialog, information-seeking
inheritance RICH*: 40, 43; CHIN*: 79
intelligent computer-aided instruction, *see* intelligent tutoring systems
intelligent tutoring systems WAHL*: 7, 28; KOBS*: 54; CHIN*: 77, 84, 89, 100; MCCO*: 237; I4: 338; KASS*: 386–390, 393, 395, 398, 408, 410; FINI*: 418
intended meaning CARB*: 146
intended recognition CARB*: 157f.

intentions CARB*: 136, 146f., 150, 153
interface
– habitable LEHM*: 164
– natural-language WAHL*: 4ff., 19, 26, 29f.; CHIN*: 75; KASS*: 407
– – learning LEHM*: 163, 165, 167ff., 171ff., 175–184, 187–193
intermediate person MORI*: 367
introspection FINI*: 425
isotopy MCCO*: 251; JAME*: 256; I4: 338; MORI*: 382f., 385
item JAME*: 261
– examples used by IMP JAME*: 263, 296
– representation in IMP JAME*: 263, 268, 281, 296, 300, 305
ITS, *see* intelligent tutoring systems

justification RICH*: 41, 47; MCCO*: 233f., 238–242

KAMP WAHL*: 5; KOBS*: 53, 59, 63–66; MORI*: 375
kernel grammar LEHM*: 164, 170, 172, 184
keyhole recognition CARB*: 157
KL-ONE FINI*: 422
KNOME CHIN*: I2: 70; 74–82, 85–88, 90–94, 97–107; QUIL*: 128; CARB*: 160f.
knowledge WAHL*: 5ff., 10ff.; RICH*: 37, 50; KOBS*: 55, 62ff., 67; I2: 70, 72; CHIN*: 74–78, 82–89, 91, 97–101, 104ff.; I3: 196; PARI*: 200–205, 211f., 214, 217f., 222, 229ff.; MCCO*: 235; COHE*: 313, 318f.; I4: 339; FINI*: 415, 426
– about domain, *see* domain knowledge
– bases FINI*: 412
– conceptual MORI*: 380
– difficulty levels CHIN*: 80f.
– meta- CHIN*: 74, 98f., 100–103
– referential MORI*: 380
– representation FINI*: 413
– semantic LEHM*: 165, 173, 175
– spectrum PARI*: 201f., 204f., 221; COHE*: 313
– syntactic LEHM*: 165, 167f., 173, 175f.
KODIAK CHIN*: 79f., 82, 91, 93, 99, 105

language stabilization LEHM*: 186, 190
LDC WAHL*: 30
learning RICH*: 48
– domain LEHM*: 176
– natural language, *see* interface, natural-language learning
least-deviant-first LEHM*: 165, 167ff., 178f., 182
letters of recommendation JAME*: 261, 276
likelihood ratings CHIN*: 85, 91–94, 97
linguistic particles WAHL*: 13
linguistics KOBS*: 54, 62; PARI*: 206, 224; JAME*: 262, 279f., 285
LISP CHIN*: 105
listener, in evaluation-oriented dialog JAME*: 258
LMS KASS*: 386, 392ff., 396ff., 405, 408f.
levels of expertise, *see* expertise levels
logic
– epistemic WAHL*: 16; KOBS*: 54, 58
– fuzzy, *see* fuzzy logic
– multi-valued FINI*: 425
long-term user model RICH*: 39

magnification, of benefits or costs of making a comment JAME*: 279f., 282
maintenance, of user model FINI*: 416
mal-rules MCCO*: 237; KASS*: 392, 396ff.
MAP CHIN*: 84
maxim
– of quality, revised CARB*: 159
– of relation CARB*: 146, 156
– of relevance CARB*: 134
MBR KOBS*: 59, 66
mental model WAHL*: 5, 9; PARI*: 203, 211f.; JAME*: 255, 307
– functional PARI*: 203
– conditions furthering accuracy JAME*: 310
– of system's dialog motivation JAME*: 256
– of system's user model JAME*: 256
meta-beliefs, *see* beliefs, meta-
meta-interpreter FINI*: 428
meta-knowledge, *see* knowledge, meta-
meta-rules CHIN*: 98
misattribution MCCO*: 236, 240f., 252
misclassification MCCO*: 234, 236, 238, 240
miscommunication CARB*: 152
misconceptions WAHL*: 7, 15, 26, 28; KOBS*: 52; I2: 71f.; CHIN*: 74f., 97f., 100–105; QUIL*: 109ff., 114–122, 125–131; CARB*: 150–153, 161; I3: 196f.; PARI*: 201; MCCO*: 233–238, 240–243, 249–252; COHE*: 313; KASS*: 395, 406f.
misgeneralization KASS*: 397
missing joins CARB*: 145, 147
modal verbs WAHL*: 13
model disparity CARB*: 159
model tracing KASS*: 404, 409
modeling factor SPAR*: 342, 351f.

modeling function SPAR*: 346–351, 354, 357, 360
modification change type MORI*: 373, 377
monolog JAME*: 260
most recently considered subgoal CARB*: 138ff., 142, 144
motivation, of dialog system JAME*: 256
multiple-agent planning systems WAHL*: 7; KOBS*: 54, 63, 65
MUMBLE I3: 197; MCCO*: 236
MYCIN RICH*: 47; COHE*: 329f.; KASS*: 388, 398

NAOS WAHL*: 22, 24
natural language LEHM*: 163, 169f.; SPAR*: 352f.; FINI*: 418
negation as failure I4: 339; FINI*: 424
negotiation CARB*: 152, 158f.
NEOMYCIN KASS*: 388f., 391, 398f.
non-decision properties I4: 336; SPAR*: 345–351, 353–358, 360, 362
norms, social JAME*: 275, 304f., 307
novel plans CARB*: 151
novice, *see* user, naive

object
– importance MCCO*: 244–247, 251
– misclassification WAHL*: 27
– perspective MCCO*: 233, 235f., 243–249, 251–254
– similarity MCCO*: 236, 239–246, 252ff.
– taxonomy MCCO*: 235f., 248
– in evaluation-oriented dialog JAME*: 258
objective facts about the user WAHL*: 13; KOBS*: 52f.; I2: 72; I4: 336; SPAR*: 344f., 348–351, 354, 362
obstacles in the user's plan WAHL*: 6
open world models CHIN*: 98
ORBIT JAME*: 260
over-answering WAHL*: 19; JAME*: 278, 282f.; MORI*: 370, 379f., 383, 385
overlay technique WAHL*: 7, 15; CHIN*: 77; KASS*: 390–393, 401f., 408

PAGAN CHIN*: 88
parsing LEHM*: 177, 180, 182
– adaptive CARB*: 161; LEHM*: 163ff., 167–173, 175–184, 188–193
– strategy LEHM*: 167f., 178
partition approach WAHL*: 17
path expansion heuristics CARB*: 148ff.
patient role I4: 336f.; SPAR*: 343f., 346–354, 356, 360f.
personality traits WAHL*: 13, 15
perspective RICH*: 36; CARB*: 161; MCCO*: 233
perturbation modeling KASS*: 390, 392f., 395f., 408f.
plan
– body CARB*: 136
– disparity CARB*: 135, 152, 155–160; COHE*: 313
– effects CARB*: 136
– hierarchical CARB*: 135f.
– identification heuristics CARB*: 137
– inference KASS*: 403
– – dynamic CARB*: 134, 137, 144, 150
– recognition FINI*: 414
– representation KASS*: 404f.
planfor CHIN*: 82
planning failures QUIL*: 113, 123, 125ff., 130
plans WAHL*: 5–8, 12, 27, 31; KOBS*: 52ff.; I2: 70ff.; CHIN*: 78, 82, 87ff., 101; QUIL*: 109–121, 127ff.; CARB*: 136, 147–161; PARI*: 201; COHE*: 313; I4: 339; KASS*: 403, 407ff.; FINI*: 414
– disparate CARB*: 152, 154
– invalid CARB*: 151, 153
– organization QUIL*: 122ff., 131
– retrieval QUIL*: 124
politeness JAME*: 256
posited superordinate MCCO*: 239f.
possible-world approach WAHL*: 7, 16
pragmatic interpretation JAME*: 272
– anticipation by speaker JAME*: 272, 274ff., 295f., 299, 304f.
– consequences for impressions of listener JAME*: 273, 276, 285f., 291f., 298
– in human-computer interaction JAME*: 309f.
– relation to conversational implicature JAME*: 272
pragmatically ill-formed queries CARB*: 145f., 148, 161
pragmatically ill-formed utterances CARB*: 134, 145
preconditions CARB*: 136, 142; CARB*: 147f.
predicates PARI*: 206–209, 216, 223ff., 228
predictions I1: 2; CHIN*: 77f., 80, 92, 104
– from user model I1: 2f.; RICH*: 35, 49; KASS*: 396, 402
preferences COHE*: 317f., 320ff., 326–330; FINI*: 415

preparatory expressions JAME*: 280
– dependent on projected ascriptions JAME*: 296
– conditions for use JAME*: 280, 285
– dependent on projected ascriptions JAME*: 295, 300f., 304f.
– pragmatic functions JAME*: 280, 302
presumption failures MCCO*: 237
presuppositions WAHL*: 26; KOBS*: 57f.
previous discourse MCCO*: 234f.
privacy WAHL*: 11, 14, 27; KOBS*: 64
projected ascriptions JAME*: 294ff., 298–305, 307
projected bias JAME*: 274–277, 286ff., 290–294, 298, 302, 307
PROLOG FINI*: 412, 430
process schema I3: 196; PARI*: 205, 208ff., 217–221, 228
property substitution heuristic CARB*: 147f.
protagonist model MORI*: 366
prototypes CHIN*: 79
PROUST KASS*: 393, 403f.
psychological model KASS*: 397, 399, 401–404, 406f., 410

question answering WAHL*: 4, 6, 9, 14, 19, 25; I2: 70; PARI*: 200–203; COHE*: 313; I4: 337
questions JAME*: 282
– attention focusing JAME*: 278, 282
– wh- WAHL*: 11, 13; KOBS*: 57f., 60; COHE*: 316f., 326ff.
– why- COHE*: 316f., 326ff.
– y/n- KOBS*: 57f., 60, 62, 64; JAME*: 282

RABBIT RICH*: 36
real estate agent CHIN*: 78, 83
recommendations WAHL*: 19; KOBS*: 52; MORI*: 379
recovery
– strategies LEHM*: 165, 168, 177f., 179
– actions LEHM*: 168ff., 177
referential knowledge, *see* knowledge, referential
relevance CARB*: 146f., 150, 155
repair theory KASS*: 403, 405, 409
replacement change type MORI*: 373
representation of user models, *see* user models, representation
resource-limited processes WAHL*: 4, 31
response CARB*: 158; MCCO*: 233–241, 249, 251–254
– cooperative QUIL*: 109ff., 126, 130
– schemas MCCO*: 236
– strategies MCCO*: 233, 236, 238–242, 253f.
– tailoring QUIL*: 128, 130
revision change type MORI*: 373f., 376
revision of user models I4: 337, 373f., 376
rhetorical predicates PARI*: 206
rigid designator KOBS*: 58
roles, conversational, *see* conversational roles
ROMPER WAHL*: 27; QUIL*: 126; MCCO*: 236, 242f., 251f.

salience of attributes I3: 197; MCCO*: 246
SC WAHL*: 5; QUIL*: 128
SC-UM CHIN*: 92
schemas PARI*: 206ff., 211–220, 223–226, 228, 231; COHE*: 324ff., 332f.
search LEHM*: 180, 182
security WAHL*: 11, 14, 27
self-bounded language LEHM*: 170, 182, 185, 188, 190
shell systems, *see* user modeling shell systems
short-term user model, *see* user model, short-term
shortcomings of UM components I1: 2; CHIN*: 105, 107; JAME*: 256; I4: 336
silence JAME*: 264, 266, 284, 301, 305
similarity CARB*: 147, 150
– metric MCCO*: 243f., 247, 249, 253f.
similar object MCCO*: 234
situation KOBS*: 56f., 59, 61
skill lattice KASS*: 394f.
social implications WAHL*: 4, 11, 14, 27; JAME*: 255, 307
social psychology JAME*: 269, 276, 285, 287, 292ff., 298, 308
SOPHIE KASS*: 389
source
– of misconception MCCO*: 233, 235, 239
– of modeling information SPAR*: 347
speech act MORI*: 371f., 375ff., 379, 383
– generation MORI*: 379, 385
– recognition I1: 3; KOBS*: 59, 65
SPIRIT QUIL*: 112, 127, 131
standard name KOBS*: 58
static property SPAR*: 344f., 348–351, 354, 361
stereotype
– activation RICH*: 38, 46
– adaptation I1: 3; RICH*: 43, 48
– approach I1: 2f.; QUIL*: 128; MORI*: 373; JAME*: 256
– hierarchy RICH*: 37, 40f., 44, 46
– techniques, *see* stereotype approach

stereotypes I1: 2; WAHL*: 13, 20f.; RICH*: 35–39, 41, 43–50; I2: 70f.; CHIN*: 74, 78ff., 104f.; CARB*: 161; PARI*: 200ff., 229; COHE*: 313, 317ff., 325–331; I4: 339; SPAR*: 356, 363; MORI*: 377ff.; FINI*: 414, 418ff.
– classification of FINI*: 422
– double CHIN*: 74f., 79f., 105f.
strategies LEHM*: 177; PARI*: 200f., 205, 208, 211–216, 218–226, 228, 230f.; COHE*: 313
student
– ideal KASS*: 389, 404
– model WAHL*: 7; COHE*: 318–322, 329–332; KASS*: 386–410
– modeling I4: 338
subjective property SPAR*: 344f., 348–351, 354, 361f.
submodel SPAR*: 351f., 354
substitution heuristics CARB*: 147, 150
support for misconception MCCO*: 234, 238ff., 242, 254
syntactic knowledge, *see* knowledge, syntactic
system canonical form LEHM*: 170, 175ff., 188

TAILOR CHIN*: 107; CARB*: 161; I3: 197
talkativeness JAME*: 266, 271, 277
task MORI*: 367
task domain LEHM*: 173, 175
task-oriented dialog, *see* dialog, task-oriented
task-related goal CARB*: 136
task-related plan CARB*: 134, 138f., 144, 147–152, 159, 161
taxonomy
– of beliefs and goals I1: 3; KOBS*: 52f., 65, 67
– of potential explanations QUIL*: 116–122, 128, 130f.
TEAM WAHL*: 30
TEIRESIAS CHIN*: 98
terminology MORI*: 364f., 374
text PARI*: 205
– comprehension WAHL*: 7; QUIL*: 109, 129
– generation WAHL*: 7
– natural language PARI*: 204
– naturally occurring PARI*: 201, 206, 211, 213
THEORIST COHE*: 322f., 332f.
theory of meaning CARB*: 134, 146
TMS, *see* truth maintenance system
topic
– common JAME*: 279, 281
– conceptualization of JAME*: 278, 281
– importance of JAME*: 279, 282f.
– of question JAME*: 278, 283
TRACK WAHL*: 12; KOBS*: 59f., 66; CARB*: 134, 136, 140, 142, 144; MCCO*: 251
transmutable systems WAHL*: 30f.
transportable systems WAHL*: 30
triggers RICH*: 37, 40, 43, 46f., 49; FINI*: 422
TRUMP WAHL*: 28, 33
truth maintenance system RICH*: 42, 47; FINI*: 429
tutoring component KASS*: 388
tutoring systems QUIL*: 116, 125, 131

UC WAHL*: 5, 19; RICH*: 36, 51; KOBS*: 60, 66; CHIN*: 74ff., 78f., 81f., 85ff., 89–94, 97–106; QUIL*: 128; CARB*: 161; PARI*: 202
UCEgo CHIN*: 75, 102, 104
UCExpress CHIN*: 103f.
UCTeacher CHIN*: 88
ULLY WAHL*: 5, 18
UMFE WAHL*: 15; RICH*: 36; KOBS*: 53, 65f.; PARI*: 229; KASS*: 408; FINI*: 416
uncertainty KOBS*: 53, 56ff., 61, 63; CHIN*: 77, 82, 91–94; LEHM*: 169, 172; I3: 198; FINI*: 426
UNIX CHIN*: 74f., 78, 81, 89, 106
unprovability FINI*: 424
unsolicited dialog contributions WAHL*: 14
update dialog MORI*: 368
USCSH CHIN*: 78
user
– adaptability LEHM*: 184, 191
– canonical MCCO*: 236
– canonical form LEHM*: 171f., 175, 188
– frequent LEHM*: 164, 170, 183
– naive PARI*: 201f., 204ff., 211ff., 229; COHE*: 313
– model WAHL*: 6, 9; RICH*: 38; QUIL*: 124, 128; CARB*: 134, 139, 144, 150, 160f.; LEHM*: 171, 192; PARI*: 202–205, 212–221, 223, 226, 228ff.; MCCO*: 235f., 238, 254; COHE*: 313f., 316–321; 325–333; MORI*: 366; KASS*: 386, 406f., 409f.
– – acquisition, *see* acquisition of user models
– – canonical RICH*: 38f.
– – highlighted MCCO*: 233f., 236, 239–243, 253f.

– – individual RICH*: 38ff.; CHIN*: 79, 81, 83; LEHM*: 164; FINI*: 414
– – initial WAHL*: 10f., 14; QUIL*: 110, 129
– – representation WAHL*: 14–17, 24ff.; KOBS*: 53, 65, 67; CHIN*: 80, 99; I4: 339
– – short-term RICH*: 39
– modeling
– – component I1: 3; KOBS*: 53, 67; WAHL*: 4, 6, 10; KOBS*: 59, 63; QUIL*: 114; I4: 339
– – dynamic MORI*: 374
– – shell systems I1: 3; WAHL*: 33; KOBS*: 53, 65; CHIN*: 106; I4: 339; FINI*: 412
– – static MORI*: 374
– property SPAR*: 344ff., 348–351, 353, 356ff., 362
– role SPAR*: 343f., 346ff., 350, 360

vagueness JAME*: 261, 271, 273, 276, 289, 291f., 302
valid plan assumption CARB*: 151
VIE-DPM WAHL*: 11, 32; KOBS*: 53, 57–60, 62–66; CHIN*: 85; MORI*: 375

wants, *see* goals
WEST KASS*: 389, 391
wh-questions WAHL*: 11
WUSOR KASS*: 386, 388, 390f., 394, 400ff.

XCALIBUR WAHL*: 30
XTRA WAHL*: 5, 19

List of Contributors

Dr. Sandra Carberry
Dept. of Computer Science, Univ. of Delaware, Newark, Delaware 19711, USA

Sandra Carberry received a BA degree in Mathematics from Cornell University and an MS degree in Computer Science from Rice University, and was a Member of Technical Staff at Bell Telephone Laboratories. She is now an Assistant Professor of Computer and Information Sciences at the University of Delaware, where she received her Ph.D. in 1985. Portions of her contribution are based on this research. Her current interests are robust models of plan recognition and the application of user models to the generation of helpful responses.

Dr. Jaime G. Carbonell
Carnegie-Mellon Univ., Dept. of Computer Science, Schenley Park, Pittsburgh, PA 15213, USA

Jaime G. Carbonell is Professor of Computer Science at Carnegie-Mellon University, founder and director of Carnegie Group Inc. and director of the Center for Machine Translation at CMU. He received his B.S. degrees in Physics and in Mathematics from MIT and his M.S. and Ph.D. degrees in Computer Science from Yale University. Dr. Carbonell has worked in various areas of artificial intelligence, including machine learning, natural language processing, planning and problem solving, machine translation, analogical reasoning and knowledge representation. He has authored over a hundred technical papers and has edited several books.

Dr. David N. Chin
Department of Information and Computer Sciences, University of Hawaii at Manoa, 2565 The Mall, Honolulu, HI 96822, USA

David N. Chin is Assistant Professor of Information and Computer Sciences at the University of Hawaii at Manoa. He conducted his Ph.D. studies at the University of California at Berkeley, where he worked on treating UC, the UNIX Consultant, as an intelligent agent which has its own goals, plans, and model of the user. His contribution summarizes the user modeling portion of his dissertation work.

Dr. Robin Cohen
Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L3G1

Robin Cohen is Assistant Professor of Computer Science at the University of Waterloo, working in the area of natural language understanding. She completed her Ph.D. in 1983 at the University of Toronto, under the supervision of Ray Perrault, in the area of natural language pragmatics and discourse. The joint research with Marlene Jones described in this volume reflects an interest in investigating the development of specific explanations (discourse) to users based on their background and goals (pragmatics).

Jill Fain Lehman
Carnegie-Mellon Univ., Dept. of Computer Science, Schenley Park, Pittsburgh, PA 15213, USA

Jill Fain Lehman is a Ph.D. candidate in Computer Science at Carnegie-Mellon University. She received her B.S. degree in Computer Science from Yale University in 1981, and her M.S. degree in the same field from Carnegie-Mellon in 1987. Her primary interests lie in the areas of natural language processing and machine learning, although previous research has included work in expert systems for the RAND Corporation. Her paper summarizes the current state of her thesis work on adaptive parsing.

Dr. Timothy Finin
UNISYS, Paoli Research Center, Route 252 and Central Ave., Paoli, PA 19311, USA

Timothy W. Finin is a technical director at the UNISYS Paoli Research Center. He received a master's degree in Electrical Engineering from M.I.T., and MSc. and Ph.D. degrees in Computer Science from the University of Illinois at Urbana-Champaign. He did research at M.I.T., in the area of robotics and computer vision, and was an Assistant Professor in the Department of Computer and Information Science at the University of Pennsylvania from 1980 to 1987. His current research interests include knowledge representation, expert systems and computational linguistics, as well as their applications to intelligent user interfaces. His paper describes recent work on creating a user modeling shell in order to support the development of various intelligent interactive applications.

Anthony Jameson
Psychologisch Laboratorium, Katholieke Universiteit Nijmegen, Montessorilaan 3, Postbus 9104, NL-6500 HE Nijmegen, Netherlands

Anthony Jameson is affiliated with the Department of Experimental Psychology at the University of Nijmegen, The Netherlands. He is writing a dissertation on psychological and normative models of the way people ascribe knowledge and values to others in everyday interaction. The present paper incorporates some of the theoretical ideas that have emerged from this research, while building on work done several years ago in connection with the HAM-ANS project at the University of Hamburg, Fed. Rep. of Germany.

Dr. Marlene Jones
Alberta Research Council, 6815 8th Street NE, 3rd floor, Calgary, Alberta, Canada T2E 7H7

Marlene Jones is a senior researcher in Artificial Intelligence in the Alberta Research Council's Department of Advanced Technologies and an adjunct professor in the Department of Computational Science at the University of Sakatchewan. Prior to joining the Alberta Research Council, Dr. Jones was a tenured Associate Professor at the University of Waterloo. Much of her research concernes AI applications within the field of education, including expert systems for educational diagnosis, expert environments for curriculum and course development, and user modeling.

Dr. Robert Kass
Center for Machine Intelligence, 2001 Commenwealth Blvd., Ann Arbor, MI 48105, USA

Robert Kass is a Senior Research Engineer at the Center for Machine Intelligence in Ann Arbor, Michigan. He recently completed his Ph.D. at the Department of Computer and Information Science at the University of Pennsylvania, where his research focused on the implicit acquisition of user models from users' interactions with an advisory expert system. His contribution summarizes some of his research in representing and acquiring agent models.

Dr. Alfred Kobsa

SFB 314: AI - Knowledge-Based Systems, Dept. of Computer Science, University of Saarbrücken, D-6600 Saarbrücken 11, Fed. Rep. of Germany

Alfred Kobsa received master degrees in Computer Science and the Social and Economic Sciences from the University of Linz, Austria, and his Ph.D. in Computer Science from the Technical University of Vienna. He has also pursued non-degree studies in Cognitive Psychology, Linguistics, Formal Logic and the Philosophy of Science at the University of Salzburg. Since 1985 he has been a senior researcher and project coordinator of the XTRA project in the German Collaborative Research Program on AI and Knowledge-Based Systems at the University of Saarbrücken, where he works in the fields of user modeling, knowledge representation and expert systems. His contribution is based on experiences gained both from his Ph.D. work as well as his current research.

Dr. Kathleen McCoy

Dept. of Computer Science, Univ. of Delaware, Newark, Delaware 19711, USA

Kathleen McCoy is Assistant Professor of Computer and Information Sciences at the University of Delaware. She received her Ph.D. in computer science from the University of Pennsylvania in 1985. Portions of her contribution are based on this Ph.D. research. Her current interests center on contextual effects of previous discourse on representation of user models and generation of subsequent responses.

Dr. Katharina Morik

Technical University of Berlin, FR-5-8, Franklinstr. 28/29, D-1000 Berlin 10, Fed. Rep. of Germany

Katharina Morik is internal project leader of the KIT-Lerner project at the Technical University of Berlin, working in the fields of knowledge acquisition and machine learning. She completed her Ph.D. thesis on belief systems and natural language understanding at the University of Hamburg in 1981. From 1982 to 1984 she worked on the Hamburg Application-Oriented Natural-Language System (HAM-ANS) with a special interest in user modeling and dialog strategy. Her contribution is based on experiences gained from this work.

Dr. Cécile L. Paris

Information Sciences Institute, 4676 Admirality Way, Marina del Rey, CA 90292-6695, USA

Cécile Paris received her Bachelor degree from the University of California in Berkeley and has just completed her Ph.D. in Computer Science at Columbia University (New York). She is now working at the Information Sciences Institute, continuing her research on natural language generation and user modeling. Her thesis work has focused on how a system should tailor a response depending on how much the user knows about the domain under consideration. Her paper reports major results from this work.

Alexander E. Quilici

University of California, Computer Science Dept., Artificial Intelligence Lab., 3531 Boelter Hall, Los Angeles, CA 90024, USA

Alex Quilici is currently completing his Ph.D. studies at the University of California, Los Angeles. His research concerns the automatic detection and correction of plan-oriented misconceptions that often occur in argumentative dialogs. His contribution summarizes aspects of this research dealing with the misconceptions of novice computer users. His research interests include user modeling, intelligent tutoring systems and models of argumentation.

Dr. Elaine Rich
MCC, 9430 Research Boulevard, Austin, TX 78759, USA

Elaine Rich is a director of the artificial intelligence lab at Microelectronics and Computer Technology Corporation (MCC), where she has been working primarily in the area of knowledge-based natural language interfaces. Prior to joining MCC, she was an assistant professor of Computer Sciences at the University of Texas at Austin. Her paper surveys the use of stereotypes in user modeling, which was the topic of her Ph.D. dissertation work.

Dr. Karen Sparck Jones
University of Cambridge, Computer Laboratory, Corn Exchange Street, Cambridge CB2 3QG, England

Karen Sparck Jones is an Assistant Director of Research at the University of Cambridge Computer Laboratory. She has worked on language and information processing problems since the late fifties, concentrating in recent years on inquiry interfaces and hence on the meaning representations and entity models these require.

Dr. Wolfgang Wahlster
SFB 314: AI - Knowledge-Based Systems, Dept. of Computer Science, University of Saarbrücken, D-6600 Saarbrücken 11, Fed. Rep. of Germany

Wolfgang Wahlster received the diploma and the doctoral degree in Computer Science from the University of Hamburg, West Germany. He was a senior researcher and coordinator in the HAM-RPM and HAM-ANS natural language dialog projects, and is now Professor of Artificial Intelligence and Database Systems and head of the AI Laboratory at the University of Saarbrücken as well as director of the Saarbrücken branch of the German AI Center. His current research includes multimodal input and output, natural language access systems, user modeling, natural language generation and explanation components for expert systems.

N.J. Nilsson: Principles of Artificial Intelligence. XV, 476 pages, 139 figs., 1982

J.H. Siekmann, G. Wrightson (Eds.): Automation of Reasoning 1. Classical Papers on Computational Logic 1957–1966. XXII, 525 pages, 1983

J.H. Siekmann, G. Wrightson (Eds.): Automation of Reasoning 2. Classical Papers on Computational Logic 1967–1970. XXII, 638 pages, 1983

L. Bolc (Ed.): The Design of Interpreters, Compilers, and Editors for Augmented Transition Networks. XI, 214 pages, 72 figs., 1983

M.M. Botvinnik: Computers in Chess. Solving Inexact Search Problems. With contributions by A.I. Reznitsky, B.M. Stilman, M.A. Tsfasman, A.D. Yudin. Translated from the Russian by A.A. Brown. XIV, 158 pages, 48 figs., 1984

L. Bolc (Ed.): Natural Language Communication with Pictorial Information Systems. VII, 327 pages, 67 figs., 1984

R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.): Machine Learning. An Artificial Intelligence Approach. XI, 572 pages, 1984

A. Bundy (Ed.): Catalogue of Artificial Intelligence Tools. Second, revised edition. IV, 168 pages, 1986

C. Blume, W. Jakob: Programming Languages for Industrial Robots. XIII, 376 pages, 145 figs., 1986

J.W. Lloyd: Foundations of Logic Programming. Second, extended edition. XII, 212 pages, 1987

L. Bolc (Ed.): Computational Models of Learning. IX, 208 pages, 34 figs., 1987

L. Bolc (Ed.): Natural Language Parsing Systems. XVIII, 367 pages, 151 figs., 1987

N. Cercone, G. McCalla (Eds.): The Knowledge Frontier. Essays in the Representation of Knowledge. XXXV, 512 pages, 93 figs., 1987

G. Rayna: REDUCE. Software for Algebraic Computation. IX, 329 pages, 1987

D.D. McDonald, L. Bolc (Eds.): Natural Language Generation Systems. XI, 389 pages, 84 figs., 1988

L. Bolc, M.J. Coombs (Eds.): Expert System Applications. IX, 471 pages, 84 figs., 1988

C.-H. Tzeng: A Theory of Heuristic Information in Game-Tree Search. X, 107 pages, 22 figs., 1988

H. Coelho, J.C. Cotta: Prolog by Example. How to Learn, Teach and Use It. X, 382 pages, 68 figs., 1988

L. Kanal, V. Kumar (Eds.): Search in Artificial Intelligence. X, 482 pages, 67 figs., 1988

H. Abramson, V. Dahl: Logic Grammars. XIV, 234 pages, 40 figs., 1989

R. Hausser: Computation of Language. An Essay on Syntax, Semantics, and Pragmatics in Natural Man-Machine Communication. XVI, 425 pages, 1989

P. Besnard: An Introduction to Default Logic. XI, 201 pages, 1989

A. Kobsa, W. Wahlster (Eds.): User Models in Dialog Systems. XI, 471 pages, 113 figs., 1989