VOLUME 7, NUMBER 3, MARCH 1992

7(3) 195–292
ISSN 0884-8173

International Journal of Intelligent Systems

Ronald R. Yager *Editor*

A WILEY-INTERSCIENCE PUBLICATION

WILEY

Publishers Since 1807

NEW YORK / CHICHESTER / BRISBANE / TORONTO / SINGAPORE

Petitioner has added romanettes to pages (i) - (iii). Otherwise, it leaves the original page numbering.

MICROSOFT CORP.
EXHIBIT 1033

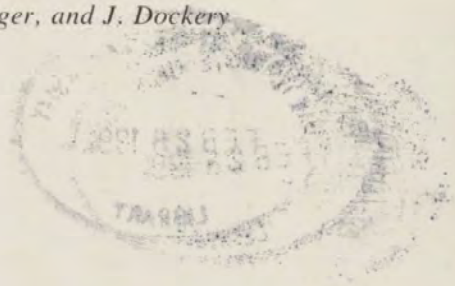
International Journal of Intelligent Systems

Volume 7, Number 3, March 1992

195 A Least Exception Logic Shell for Intelligent Systems
S. D. Post and A. P. Sage

245 TRUMP: A Transportable Language Understanding Program
P. S. Jacobs

277 Combat Modeling with Imprecise Data
V. Protopopescu, R. Yager, and J. Dockery



0884-8173(199203)7:3-7

VOLUME 7, N

Inter
of In

A WILEY

NEW YO

The *International Journal of Intelligent Systems* (ISSN: 0884-8173) is published monthly except in April, May, October, and November, one volume per year, by John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158.

Copyright © 1992 by John Wiley & Sons, Inc. All rights reserved. Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the U.S. Copyright Law without the permission of the copyright owner is unlawful. Second-class postage paid at New York, NY, and at additional mailing offices.

The code and copyright notice appearing at the bottom of the first page of an article in the journal indicate the copyright holder's consent that copies may be made for personal or internal use, or for the personal or internal use of specific clients, on the condition that the copier pay for copying beyond that permitted by Sections 107 and 108 of the United States Copyright Law. The per-copy fee for each item is $4.00 and is to be paid through the Copyright Clearance Center, Inc., 21 Congress St., Salem, MA 01970. This consent does not extend to other kinds of copying, such as copying for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. Such permission requests and other permission inquiries should be addressed to Permissions Dept.

Subscription price (1992): $400.00 per volume. Outside U.S.A.: $500.00 (air service and handling included). Please allow four weeks for processing a change of address. For subscription inquiries, please call customer service at (212) 850-6645 or write to the above address.

Claims for undelivered copies will be accepted only after the following issue has been received. Please enclose a copy of the mailing label or cite your subscriber reference number in order to expedite handling. Missing copies will be supplied when losses have been sustained in transit and where reserve stock permits.

Postmaster: Send address changes to *International Journal of Intelligent Systems*, Susan Malawski, Fulfillment Manager, Subscription Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158.

Advertising Sales: Inquiries concerning advertising should be forwarded to Roberta Frederick, Advertising Sales, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158, (212) 850-8832. Advertising Sales, European Contact: Michael Levermore, Advertising Sales Manager, John Wiley & Sons, Ltd., Baffins Lane, Chichester, Sussex PO19 1UD, England.

Other Correspondence: Address all other correspondence to: *International Journal of Intelligent Systems*, Diane Cerra, Professional, Reference, and Trade Group, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158.

The contents of this journal are indexed in the following: *Automatic Subject Citation Alert®*, *Compumath Citation Index®*, *Current Contents/Engineering, Technology & Applied Sciences*, **INSPEC,** and *SCISEARCH®*.

Manuscripts should be submitted to the Editor, Dr. Ronald R. Yager, *International Journal of Intelligent Systems*, Machine Intelligence Institute, Iona College, New Rochelle, NY 10801. **Information for contributors** appears in the first and last issue of each volume.

TRUMP: A Transportable Language Understanding Program

Paul S. Jacobs
Artificial Intelligence Laboratory, GE Research and Development Center, Schenectady, New York 12301

Transportability has perpetually been the nemesis of natural language processing systems, in both the research and commercial sectors. During the last 20 years, the technology has not moved much closer to providing robust coverage of everyday language, and has failed to produce commercial successes beyond a few specialized interfaces and application programs. The redesign required for each application has limited the impact of natural language systems. TRUMP (TRansportable Understanding Mechanism Package) is a natural language analyzer that functions in a variety of domains, in both interfaces and text processing. While other similar efforts have treated transportability as a problem in knowledge engineering, TRUMP instead relies mainly on a "core" of knowledge *about* language and a set of techniques for applying that knowledge within a domain. The information about words, word meanings, and linguistic relations in this generic knowledge base guides the conceptual framework of language interpretation in each domain. TRUMP uses this core knowledge to piece together a conceptual representation of a natural language input by combining generic and specialized information. The result has been a language processing system that is capable of performing fairly extensive analysis with a minimum of customization for each application.

I. INTRODUCTION

Natural language should be a component of most computer systems. It is the *natural* way people communicate, so clearly the reason language is absent from most computer software is not because any menu, icon, or command language is any substitute. It is because natural language is so *unnatural* for computers: Human-like communicative capabilities require a wealth of knowledge far beyond what machines will master in this century. Thirty years of natural language research has failed to produce much more than a series of small-scale demonstration programs.

The heart of the natural language problem seems to be in the difference between the way human beings use language and the way programs operate. People use essentially the same grammar, vocabulary, and basis for language regardless of the topic of discussion, while computer programs are brittle,

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 7, 245–276 (1992)
© 1992 John Wiley & Sons, Inc. CCC 0884-8173/92/030245-32$04.00

PORTLAND STATE UNIVERSITY LIBRARY

relying mainly on specific rules of communication. People take advantage of what they know in learning new words and phrases, while in computer programs a large knowledge base is often a disadvantage in adding new information. These differences explain why, in spite of quite a number of substantial natural language research efforts, almost nothing is shared from one natural language program to the next. The tools and techniques reported here have addressed this fundamental problem.

Winograd's SHRDLU program[1] was a milestone in showing both the capabilities and limitations of natural language programs. SHRDLU communicated effectively within a simple "toy" world, but almost none of its linguistic capability extended beyond this world. Natural language technology has since transcended the toy worlds to apply successfully to commercial databases,[2] but much of the brittleness remains. A number of recent efforts have tried to ease the transportability of such natural language database interfaces.[3–5] This work has been confined to databases, and has emphasized tools for building up the domain-specific information that each interface requires.

An important simplification in the database applications is that most words and phrases generally refer unequivocally to a database field or relation. This eliminates a major issue in transportability, that is, the problem of defining general word meanings. However, the simplification is also a barrier to transportability, because each system has its own largely independent lexicon. The construction of a lexicon is often the main task in customizing a database interface.

Beyond database interfaces, the simplified view of a lexicon as a simple lookup facility, linking words to entries, is insufficient. Text processing and cooperative dialogues require a greater depth of analysis, distinguishing nuances of word meaning and tying together complex linguistic structures. For these more challenging applications of natural language, transportability entails not only the use of a natural language program for multiple applications, but also the sharing of a knowledge base among applications.

TRUMP (TRansportable Understanding Mechanism Package)[6,7] is a natural language system designed to apply this "core" knowledge about language, as well as the language analysis program, to multiple domains. TRUMP shares its knowledge base with a language generation program called KING (Knowledge-INtensive Generator),[7] which is also transportable. The system relies on a knowledge representation framework in which specialized information derives as much as possible from general structures. An important component of language analysis then starts from a general interpretation of an input and refines this interpretation into a meaning representation.

A number of application programs use TRUMP, including SCISOR (System for Conceptual Information Summarization, Organization, and Retrieval)[8–10] and an organizational directory interface called DEUCE (Directory Extraction Using Common English).[11] TRUMP developed over a three-year period. The program is now part of a package called the GE NLTOOLSET which GE components are applying internally. In our experience, TRUMP is unequaled in the

degree of common benefit to text processing, language analysis, and generation.

The section that follows briefly covers some of the applications of TRUMP; the next sections deal with the essential elements of TRUMP's transportability: the "core" lexicon and knowledge base design and the semantic interpretation mechanism.

II. APPLICATIONS OF TRUMP

TRUMP currently serves as the cornerstone of a set of natural language tools used in a number of locations within the General Electric Company. Two experimental applications, an interface to an on-line directory and a text-based information retrieval system, illustrate the flexibility of the language processing mechanism.

A. DEUCE

The directory system, DEUCE (Directory Extraction Using Common English)[11] provides a simple natural language interface to an existing on-line telephone and organizational directory for the GE R&D Center. The installation of DEUCE required approximately six person-months, including porting the database from an IBM to a UNIX environment and integration of database code in the C language with natural language software written in common Lisp. The system is accessible through a shell command to most workstations on the local network. An example interchange is the following (user input is typed to the "DEUCE>" prompt):

```
% deuce
DEUCE> Who's the manager of the artificial intelligence branch?
ARTIFICIAL INTELLIGENCE BRANCH's manager is LARRY M SWEET
DEUCE> What's his phone number?
LARRY M SWEET's telephone number is 5362
DEUCE> Who is Sieter?
 no record for SIETER exists
IRA L SIDER is located at building K-1 room 5C40
IRA L SIDER's telephone number is 6888
STEPHEN O SUTTER is located at building K-1 room GATE
STEPHEN O SUTTER's telephone number is 6041
other closely related names:
 TIMOTHY L SAUTER, REINHARD SIDOR
DEUCE> In what lab does Sider work?
IRA L SIDER's laboratory is the INFORMATION SYSTEMS LABORATORY
```

This sort of exchange is not sensational in that a variety of commercially available products (cf. Ref. 22), as well as research systems (cf. Refs. 3–5), provide similar database front ends. The main unusual characteristic of DEUCE is that the natural language interface, including the semantic compo-

PORTLAND STATE UNIVERSITY LIBRARY

nent, is not designed for directories in particular or for databases in general. The customization of the interface was easy because TRUMP's knowledge base helped to control the multiple ways of expressing a query. For example, given the domain-specific relationship between the *concept* of "managing" and the database, the program could correctly handle requests like "the manager of Sider," "Who manages John?" and "The AI Branch manager."

B. SCISOR

SCISOR (System for Conceptual Information Summarization, Organization, and Retrieval) is an information retrieval system described in other papers (cf. Refs. 8–10). SCISOR provides access to information in natural language texts, including question answering, text retrieval, and automatic alert capabilities.

The following is an example of the operation of SCISOR on a simple news story:

```
   W ACQUISITION UPS BID FOR WARNACO
Warnaco received another merger offer, valued
at $36 a share, or $360 million. The buyout
offer for the apparel maker was made by the
W Acquisition Corporation of Delaware.
```

User: *Who took over Warnaco?*
System: W Acquisition offered $36 per share for Warnaco.
User: *What happened to Warnaco last Tuesday?*
System: Warnaco rose 2 ½ as a result of rumors.

SCISOR processes stories from the Dow Jones news service, handling roughly a day's worth of stories per hour (about ten seconds per story). The program deals only with stories about mergers and acquisitions (about 80 stories per day out of a total of several hundred news items), but the core text processing tools have adapted easily to a number of other domains as well. For example, the system, including TRUMP, adapted easily within weeks to the task of generating a database from naval operations reports, and performed with high throughput and accuracy.[13] The current direction of the project is to make the techniques of SCISOR still more generic, applying them to robust but more superficial processing. Information retrieval technology must ultimately address the problem of accessing information from arbitrary texts, while SCISOR to this point has dealt only with texts in limited domains.

One of the exceptional characteristics of SCISOR is that the same natural language processing and knowledge representation framework applies to text processing and question answering. TRUMP performs the analysis of both the texts and the queries, and KING generates the system's responses. SCISOR is, to our knowledge, the only such system that uses a single knowledge base for these tasks. The next section describes the design of this knowledge base and how it affects TRUMP.

III. THE CORE KNOWLEDGE BASE

The theme of TRUMP is that a powerful base of knowledge *about* language is the key to transportability in natural language. This knowledge must be in a declarative form, to apply to both language analysis and generation, and must be sufficiently general to cover linguistic constructs in different domains and forms. Because every application still requires significant domain-specific linguistic and conceptual information, the core knowledge base must also support the addition of specialized knowledge.

Several factors determine the power of a knowledge base for natural language processing. Among these variables are the size of the lexicon, or number of known words; the richness of information about each word; the treatment of lexical compounds, or phrases; the representation of grammatical relations; and the linking of linguistic and conceptual structures.

The size of the lexicon is certainly important, but the richness of information about each word tempers this measure. Also, since the meaning of each word depends on the context in which it appears, phrasal knowledge is at least as important as individual word entries. Such phrasal knowledge helps in determining the sense of a word and also in interpreting phrases whose conceptual content seems to be more than the combination of individual word meanings. Phrasal knowledge goes beyond stock phrases like "kick the bucket" and "piece of cake" to include more flexible constructs like "Chinese restaurant," "answer the door," and "give a kiss." To avoid redundancy, aid transportability, and ease knowledge acquisition, the phrasal lexicon must group these phrases with other similar constructs, such as "Italian restaurant," "answer the phone," and "get a kiss."

Knowledge about words is inextricably tied to knowledge about word meanings. Some of the phrases above illustrate these conceptual underpinnings of language; for instance, the relationship between "give a kiss" and "get a kiss" seems to be through some conceptualization of kissing as a transfer rather than through any direct tie between the words "give" and "get." Conceptual knowledge often determines the interpretation of general linguistic relationships as well; for example, "pay the waiter" and "pay the bill" have the same linguistic structure but differ semantically because of the difference between the waiter *concept* and the bill *concept*.

Most work in linguistics and natural language processing has concentrated on the grammar of language. Knowledge of linguistic structure, however, goes beyond grammar and word order. More general relations, such as the generic association between a word and a modifier, or between a verb and its complements, help to determine the correct syntactic interpretation of an input as well as its semantic content.

The critical task in language analysis is the production of a representation of the conceptual content of a natural language input. The linguistic structures described above, including lexical entries and linguistic relations, help to determine this conceptual interpretation. However, in order to produce a semantic

interpretation from a linguistic analysis, a program must also make use of relationships between language and meaning.

This demand for a rich linguistic representation with conceptual foundations, along with the motivation for a core knowledge base that the analyzer can share with a generator, are behind the following three most important aspects of TRUMP's knowledge base:

- A uniform representation for linguistic and conceptual knowledge
- A hierarchical phrasal lexicon.
- Explicit associations between linguistic and conceptual structures

These knowledge representation issues tie together in relating language to meaning, in such a way that a language analyzer can produce conceptual interpretations without overly specific knowledge. The rest of this section covers each of these elements of the core knowledge base in greater detail.

A. Uniform Knowledge Base Design

The design of the knowledge base shared by TRUMP and KING differs drastically from most other substantial natural language systems. Most programs use a grammar encoded in one formalism, a conceptual knowledge base in a different knowledge representation language, and a set of rules for mapping between the grammar and the knowledge representation. Moreover, most all such systems that perform both language analysis and generation have separate grammars for analysis and generation, as well as separate sets of mapping rules. This is disastrous for transportability, because a change to a new domain demands modifications to each of the many different components of the knowledge base.

By contrast, the TRUMP/KING knowledge base includes linguistic and conceptual knowledge as well as structures that associate language with meaning, all encoded using a common representational framework called Ace,[14] a derivative of the KODIAK knowledge representation,[15] which is similar to KL-ONE.[16] The entire knowledge base, including the grammar, conceptual knowledge, and associations, applies to both analysis and generation, although the analyzer (TRUMP) and generator (KING) use the knowledge very differently. The main practical reason for lack of uniformity in other natural language systems is that the individual components are designed by different individuals or groups, often before the conception of the total system. Designing the knowledge base before either of the two programs was the secret to uniformity in TRUMP/KING.

The shared knowledge base makes the system more portable by limiting the amount of work that must be reduplicated for each domain, but also qualitatively improves system capabilities. Because the analyzer and generator use the same knowledge, the language produced is consistent with the language understood. This allows the system more easily to incorporate part of a request in its response, and avoids the awkwardness of not understanding a word or construct that the program has itself used. Finally, the analyzer benefits from knowledge more typically included in the generation lexicon, and vice versa.

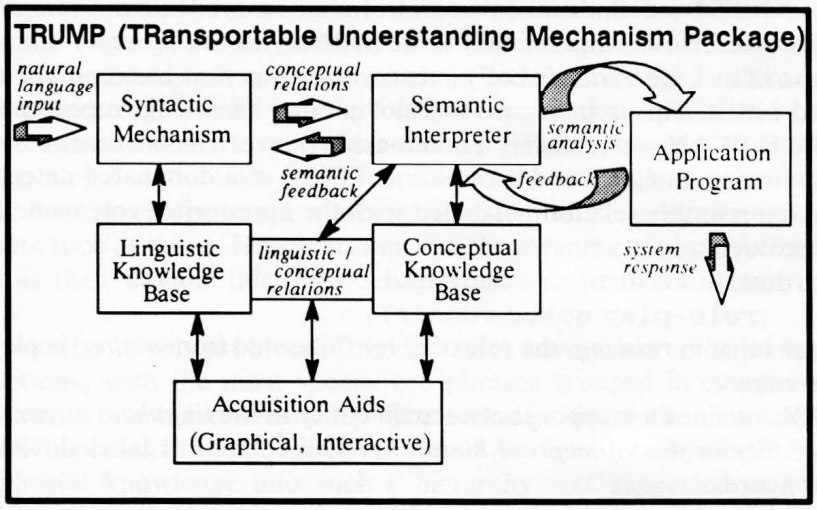


Figure 1. TRUMP system organization.

1. System Organization

Part of the design of TRUMP is to encode information within the knowledge base rather than as part of the program, wherever possible. Many natural language analyzers augment their knowledge of language with "mapping rules"—procedures that guide the production of conceptual structures from a linguistic input. A declarative representation in TRUMP replaces these mapping rules.

Figure 1 shows the overall structure of TRUMP. The parser and semantic interpreter share with the generator (not shown) a knowledge base consisting of grammatical knowledge, conceptual knowledge, and associations between linguistic and conceptual structures. The rest of this section deals with the nature of that shared knowledge base.

2. Basic Knowledge Representation Structures

TRUMP uses the Ace knowledge representation framework. Many of Ace's basic elements can be found in other knowledge representation schemes (e.g., *isa* links, slots, and inheritance).[15–17] but Ace has a unique ability to represent structured relationships among categories that contribute to linguistic expression (see the descriptions of REF and VIEW below). The connections in an Ace hierarchy include the following:

DOMINATE—defines as *isa* link between two categories. This relation is labeled with a **"D"** in the figures.
`(dominate action running)` means that *running* is an *action*—i.e., *action* dominates *running*.

MANIFEST—defines a constituent of a category. Unless a *role-play* applies (see below), this relation is labeled **"m"** in the figures.

`(manifest action actor)` means that an *action* has an *actor* associated with it. This is analogous to a slot in other knowledge representations.

ROLE-PLAY—establishes a relationship between a constituent (slot) of a dominating category and a constituent (slot) of a dominated category. In the figures, this relation is labeled with the appropriate role name for the constituent.

```
(dominate action running
  (role-play actor runner))
```

means that in *running*, the role of *actor* (inherited from *action*) is played by the *runner*.

REF—defines a mapping between an entity in the *linguistic hierarchy* and an entity in the *conceptual hierarchy*. This relation is labeled "REF" in the figures.

`ref lex-run running)` means that when the lexical category *lex-run* is invoked, the concept of *running* should be invoked as well. This is the main channel through which semantic interpretation is accomplished.

VIEW—defines a metaphorical mapping between two categories in the conceptual hierarchy.

```
(view transfer-event action
  (role-play source actor))
```

means that in certain cases, an *action* can be metaphorically VIEWed as a *transfer-events*, with the *actor* VIEWed as the *source* of the transfer.

In the rest of this article, diagrams will show sections of TRUMP's knowledge base graphically, rather than using the text form of the connections above. The graphical form makes it much easier to examine and manipulate a representation where knowledge is organized hierarchically.

TRUMP's grammar, lexicon, and conceptual knowledge base are all encoded using this representation. The grammar, consisting of about 150 rules, is similar enough to other feature systems[18] that it does not merit separate coverage here. The discussion that follows will present two components of TRUMP's knowledge base—the lexicon and the other relationships between language and meaning—making use of these knowledge structures.

B. The Hierarchical Phrasal Lexicon

Natural language processing systems must use a broad range of lexical knowledge to account for the syntactic use and the meaning of words and constructs. The problem of understanding is compounded by the fact that language is full of *nonproductive* constructs—expressions whose meaning is not fully determined by examining their parts. To handle these constructs, some systems use a phrasal lexicon,[19–23] a dictionary designed to make the representation of these specialized constructs easier.

The problem with phrasal lexicons is that they are too rigid: the phrasal

knowledge is entered in a way that makes it difficult to represent the many forms some expressions may take—without treating each form as a distinct "phrase." For example, expressions such as "send a message," "give a hug," "working directory," and "pick up" may be handled as specialized phrases, but this overlooks similar expressions such as "give a message," "get a kiss," "working area," and "take up." To perform the in-depth semantic analysis required for knowledge-based systems, a system must recognize specialized constructs such as these. However, much of the meaning of phrasal constructs, as well as their flexible linguistic behavior, may come from a more general level.

A solution to this problem of rigidity is to have a hierarchy of linguistic constructions, with the most specialized phrases grouped in categories with other phrases that behave similarly. The idea of a linguistic hierarchy is not novel, having roots in both linguistics[24,25] and Artificial Intelligence.[26] Incorporating phrasal knowledge into such a hierarchy was suggested in some AI work,[20] but the actual implementation of a hierarchical phrasal lexicon requires substantial extensions to this work.

TRUMP's hierarchical lexicon has gone through several development cycles while retaining its basic structure. Currently, the lexicon includes about 10,000 unique word roots, a set of word senses grouped into semantic categories using a core hierarchy of 1,000 concepts, and knowledge about sense preferences as well as the combinations described here. The design of the lexicon makes it easier to encode domain-specific phrasal knowledge within the framework of existing lexical entries. The lexical knowledge base has certain characteristics of both traditional and phrasal lexicons, all represented using the Ace knowledge structures outlined earlier. The lexicon includes information about simple *lexemes,* including words, abbreviations, and contractions (such as 's), and compound lexemes, or phrases. The phrasal portion of the lexicon includes the following four classes of entries:

- word sequences
- lexical relations
- linguistic relations
- linguistic/conceptual relations

The following considers each of these classes of phrasal entries, their encoding in TRUMP's knowledge base, and examples of phrases to which they apply.

1. Word Sequences

Word sequences are phrases such as "by and large" and "let alone" that must be treated as compound words because there is little hope in trying to determine their meaning by examining their components. Internally, these word sequences may or may not be grammatical (e.g., "kick and bucket" is internally grammatical, but "by and large" is not). Because this type of com-

pound lexeme is very specialized, word sequences do not benefit from the hierarchical lexicon organization, as do the other three types of relations covered next.

2. Lexical Relations

Lexical relations include compound lexical entites such as "pick up" and "sell out" that can appear in a variety of surface forms, but have some general relationship among their simple lexeme constituents. These relations also include many compound nomials, where a general lexical semantic relation can describe the behavior of a group of compound lexemes.

In contrast to the individual subcategories of word sequences, there are many entries that fall underneath each individual subcategory of lexical relations. Most of the entries under these subcategories, however, share constituents with other entries, which makes generalization possible. For example, Figure 2 shows how a lexical compound like "branch manager" can fall under a series of parent relations in the hierarchy. The *compound-noun* relation describes the syntactic behavior of the relation, while the *cn-v-head-o-tail* node groups compound nominals that semantically relate as *verb-object*. This allows TRUMP's semantic interpreter to determine that "branch manager" and "managing the branch" have the same semantic relationship. The *cn-xxx-manager* node allows the database package to treat the various manager relations uniformly.

As Figure 2 shows, knowledge in the lexical hierarchy may be redundant. The semantic interpreter could apply the *verb-object* relation to determine the relationship between "branch" and "manager" without the specific lexical entry for this combination. Our experience has shown, however, that both speed and accuracy of semantic interpretation improve when the hierarchy includes these specific phrases. But the hierarchy prevents having to encode all the syntactic and semantic properties of these more specific relations.

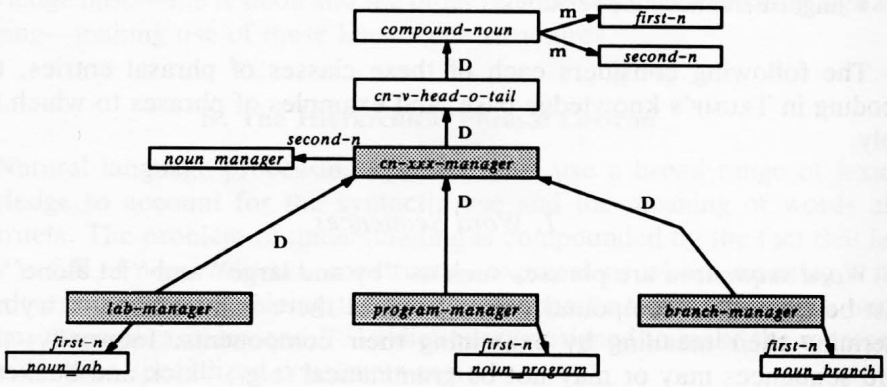


Figure 2. The compound lexemes cn-xxx-manager.

3. Linguistic Relations

Linguistic relations isolate certain linguistic structures from word order. These relations come into play particularly where different ways of using a word or word combination express a common meaning. TRUMP invokes relations according to constraints on their constituents, where the constituents may be simple lexemes, compound lexemes, or syntactic structures. An example occurs in the sentence "John was sold a book by Mary" where the object of the preposition is the main actor of the even described by the verb. This condition occurs only when the whole verb is in the passive form (constraint 1) and the preposition in the modifying prepositional phrase is *by* (constraint 2).

Linguistic relations are difficult to represent for two reasons: 1) Their constituents are not always simple lexemes and 2) usually there are additional constraints on each constituent. The relations, however, capture many generalities that prevent the need for excessive knowledge at the level of individual expressions. For example, TRUMP's knowledge base includes a linguistic relation for the class of the modifying prepositional phrases. In some instances, prepositional phrases modify noun phrases and verb phrases in almost the same way (e.g., "The man *on the hill* is a skier" and "We had a picnic *on the hill*"). In other cases prepositional phrases modify noun phrases and verb phrases in completely different ways (e.g., "The man *by the car* is my father." and "The boy was hit *by the car*.") TRUMP is able to represent both types of linguistic relation by having more than one level of generic representation. Figure 3 shows the general modifying relation (*mod-rel*) at the first level below *compound-lexeme*.

Prepositional phrases that are homogeneous across noun phrases and verb phrases are represented underneath this category. Below *mod-rel* in Figure 3 are the verb-adjunct (*va*) and noun-post-modifier (*npm*) categories, which make up the second level of generic representation. Prepositional phrases that modify verb phrases and noun phrases differently are represented underneath these categories.

As an example, in Figure 3, the *mod-rel* category has the more specific modifying relation *to-pmod-relation* underneath it, which is a modifying relation where the preposition in the modifier is *prep-to*. Example uses of this

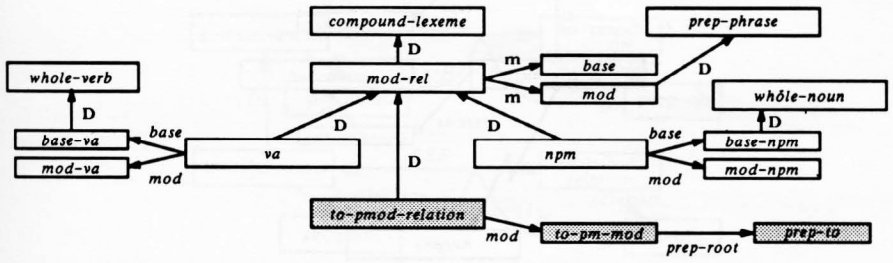


Figure 3. The modifying-relation compound-lexeme hierarchy.

prepositional phrase are found in the sentences: "The train went to New York" and "the train to New York leaves at five." In cases such as these, linguistic relations help in encoding particularly general phrases. In other cases, an interaction between linguistic structure and conceptual information helps to constrain a phrase somewhat more. The next segment briefly addresses this combination of knowledge sources.

4. Linguistic/Conceptual Relations

Strict linguistic constraints often fail to capture a generalization underlying some expressions, as in the case of "giving a kiss" or "giving permission," "getting permission," and "having permission." These expressions seem to belong to the group of *abstract possession concepts whose possessed is "permission,"* which combines a class of concepts with a lexical category.

These compound lexemes have the unique characteristic of allowing linguistic relations to have explicit conceptual constraints. In the phrase "give a hug" there is an abstract relationship between the concept of *giving* and the simple lexeme *noun-hug* that implies the concept of *hugging*. Figure 4 shows the representation of this linguistic/conceptual relation, called *lc-give-hug*.

The hierarchy of linguistic relations helps the language analyzer to refine its initial interpretation into a more complete meaning representation. This refinement, known as *concretion*—making concepts more concrete—is covered in Section IV–B.3.

This presentation of TRUMP's lexicon has illustrated the range of linguistic structures that help to identify lexical and other linguistic relations in their various surface forms. The hierarchy of linguistic structures provides a flexible means of representing generalizations that contribute to linguistic expression. To take advantage of this flexibility in semantic interpretation, TRUMP must also be able to relate linguistic structures to their conceptual interpretations. This relationship is the next topic.

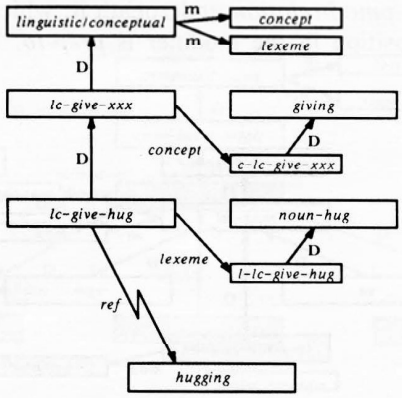


Figure 4. The linguistic/conceptual relation lcr-give-hug.

C. Associating Language and Meaning

TRUMP's knowledge representation, Ace, emphasizes knowledge *about* language in a form applicable to both analysis and generation. This knowledge includes both linguistic and conceptual information, as well as associations between language and meaning. Two such associations, REF and VIEW, link linguistic structures to their conceptual structures and associate concepts with other concepts that they help to express.

1. The REF Association

Linguistic relations provide a hierarchy of linguistic roles that are associated with conceptual interpretations. REF provides both analyzer and generator with an explicit link between linguistic and conceptual roles. This explicit relationship also promotes the interaction between abstract and more specialized linguistic knowledge. For example, the relationship between the indirect object and the conceptual *recipient* is a high-level REF association that may combine with particular knowledge about verbs of a more concrete nature.

Two important REF relationships in Ace are illustrated in Figure 5. The association of the *to-pmod-relation* at the level of *g-trans-event* for "generalized transfer events," a higher level than the *verb-indir-relation,* represents the knowledge that the object of the preposition "to" describes a more general role, *destination,* than the *recipient* described by the indirect object of a verb. Generalized transfer events include physical transfers as well, while in transfer events the recipient must abstractly receive the transferred object.

REF addresses the problem of determining conceptual relationships by providing a general means of associating linguistic and conceptual roles. Unlike systems such as Hirst's,[27] such relationships are often derivable from the knowledge structures in the network rather than being required explicitly for each word sense template. For example, the relationship between the indirect object of the verb "sell" and the *customer* role is derived from the ROLE-PLAY between *indirect-object* and *recipient* and the fact that the *customer* plays a *recipient* role in the transfer of merchandise (cf. Ref. 28). With the preposition "to," as in "sold the book to Fred," TRUMP fills the role of *cus-*

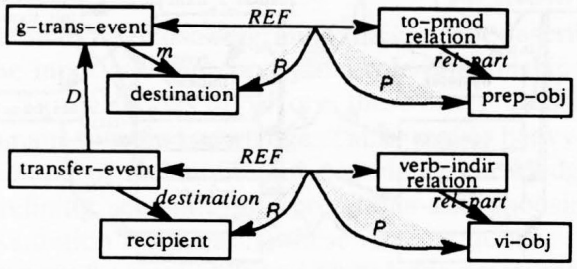


Figure 5. REF links linguistic and conceptual structures.

tomer by virtue of its ROLE-PLAY relationship with the *destination* role. REF thus helps to encode both generic information, such as these role relationships, and specialized knowledge, such as the meanings of particular phrases.

2. The VIEW Association

The VIEW association is similar to REF, except that it represents metaphorical relationships rather than associations between linguistic and conceptual roles. For example, there is a broad class of English "transfer" expressions, such as "give a kiss," "take a punch," and "get a back rub." A conceptual relationship captures this generalization, that is, *being acted upon is often viewed as a transfer, in which the patient of the activity is viewed as the recipient.*

The example of a command that "takes arguments" is similar, in that commands do not really "give" or "take" or "get" anything, but the language is extremely consistent in describing them as if they did. Figure 6 shows a VIEW relationship used to handle expressions like "getting arguments," "taking arguments," and "giving back a result." This representation has been slightly abbreviated due to its complexity; in reality the common manner of describing the complex *command-execution* event belongs to a broader class of metaphors along with "The frosting takes two cups of sugar." The VIEW as illustrated, however, encompasses quite a range of expressions. The power of this VIEW comes from the ROLE-PLAYs that capture the consistency of the metaphorical expressions, such as that relating *command* to *cooperator* and thereby to the *source* role of an initial transfer in an exchange (*causal–2–trans*). VIEW thus provides a tool for handling numerous role relationships that otherwise would be enumerated as special cases in the system.

VIEW and REF combine to determine the range of expressions for particular concepts. For example, Figure 7 shows a section of the linguistic/conceptual knowledge base of TRUMP, including several different types of *transfer-events,* the linguistic relations associated with these events, and VIEW and REF associations representing relationships that help to express different types

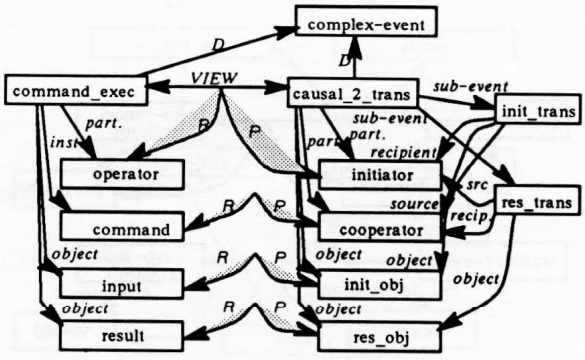


Figure 6. Capturing metaphorical generalizations.

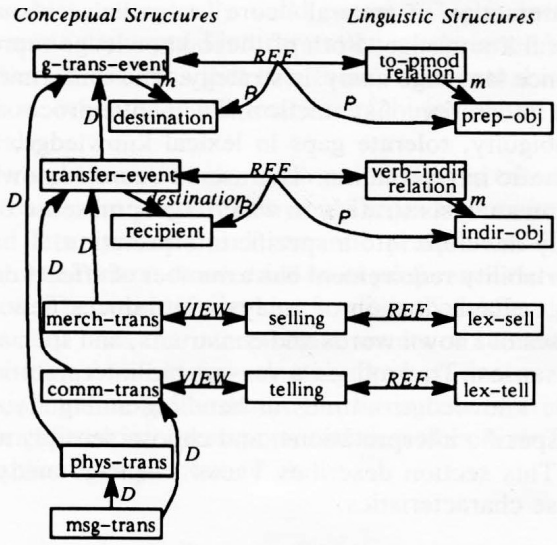


Figure 7. A knowledge base snapshot.

of transfers. The concept of a *merchandise-transfer,* for example, is linked by a VIEW to either *buying* or *selling*. The concept of a *msg-transfer* (as in "send the message to Jones" is a physical transfer, transfer of possession, and communication transfer—one might substitute "tell" or "give" for send.

This sort of linguistic/conceptual hierarchy helps to illustrate how TRUMP can take advantage of knowledge about linguistic expression, such as the relationship between transfers and indirect objects, in analyzing more specific words and constructs.

This section has covered the basic framework of TRUMP's core knowledge base, a uniform hierarchy that explicitly links linguistic and conceptual structures at multiple levels. The next section describes how the language analyzer uses this knowledge, both in controlling the consideration of possible interpretations and in refining its work into a specific meaning representation for an input.

IV. LANGUAGE ANALYSIS IN TRUMP

A language analyzer must determine the concept described by each word or phrase of the input and the conceptual role relationships that tie together these concepts. TRUMP's job is to perform these interpretation functions with a minimum of domain-specific knowledge. TRUMP makes heavy use of the generic knowledge base and of techniques for combining knowledge sources in constructing and refining semantic interpretations and choosing an appropriate meaning representation from among these interpretations.

Two of the main features of TRUMP knowledge representation as described in Section III are the uniformity of encoding of linguistic and conceptual knowl-

edge, and the interaction of general "core" knowledge about language with specialized phrasal knowledge. Both of these knowledge representation characteristics influence language analysis strategy. The uniformity of representation supports the integration of syntactic and semantic processing, which helps to deal with ambiguity, tolerate gaps in lexical knowledge, and produce an appropriate semantic interpretation. The use of "core" knowledge about language motivates an analysis strategy in which a general sense of the meaning of an input gradually develops into a specific interpretation.

The transportability requirement has a number of effects on the way TRUMP analyzes language. Each domain of analysis introduces new words and constructs, new senses of known words and constructs, and specialized interpretations of known senses. To produce a reasonable interpretation, TRUMP must combine multiple knowledge sources in handling ambiguity, work gradually from general to specific interpretations, and choose sensibly among candidate interpretations. This section describes TRUMP language analysis mechanism, emphasizing these characteristics.

A. Integrating Syntax and Semantics

Any natural language parser that handles real text must address critical issues such as controlling ambiguity and selecting an appropriate parse. The control of a language analyzer, in addition to being a major efficiency concern, is especially important for a transportable system because a small change in domain knowledge can change the interpretation of an input. This is not a problem with a domain-specific system, where each word or phrase might have a single prescribed interpretation.

This discussion concentrates on the dynamic knowledge structures that keep track of possible interpretations of an input and relate these conceptual interpretations to linguistic alternatives. The selection of methods for maintaining these interpretations drastically impacts efficiency and also contributes to the choice of the "best" interpretation, described in Section IV–C.

TRUMP's strategy for language analysis is close to that of Lytinen's integrated parser.[29] Some parsing models[30–33] use strictly syntactic methods; others[26,34–36] depend on semantic information to aid in parsing. Real interfaces and text processing systems, however, make little use of the "correct" parse; an accurate parse is a potential aid in conceptual interpretation but not a prerequisite. For this reason, some systems emphasizing conceptual analysis[37,38] actually produce no parse trees, or perform only a rudimentary syntactic analysis. By contrast, the integrated strategy acknowledges the role of syntax in language interpretation, but emphasizes the contribution of syntax to preferences in semantic interpretation rather than the converse.

While TRUMP adopts the integrated approach, it addresses a number of complications in combining syntax and semantics. For example, Lytinen's system uses a "best conceptual fit first" strategy, preferring a parse that produces the desired conceptual interpretation. This method fails when the best parse actually combines syntactic and semantic preferences. Also, it can poten-

tially try all possible conceptually valid parses. This is inefficient, because occasionally the same conceptual interpretation is supported by more than one parse. Furthermore, it assumes that the "best" conceptual interpretation is a local consideration, while later information might invalidate an interpretation that fits best locally.

Like Lytinen's program, TRUMP allows semantics to take the lead in determining how sentences are parsed; however TRUMP also maintains multiple interpretations and lets syntactic and semantic preferences interact. Furthermore, it allows global context to override local preferences in choosing the best final interpretation. The result is a parsing strategy that generally applies syntax before semantics, but uses the intermediate results of semantic interpretation as a guide in parsing as well as for selecting the best final result.

The discussion that follows shows the basic framework for maintaining linguistic and conceptual interpretations and describes how TRUMP deals with some of the complications mentioned above.

1. Dynamic Structures

TRUMP is a multiple-path parser, in contrast to some deterministic parsers,[33,39] which account for many linguistic preferences without requiring complex data structures to maintain the state of the parser. Like other multiple-path systems, TRUMP uses a *chart,* a graph structure that represents multiple parses as sequences of candidate sets. Each distinct path through the chart comprises a possible parse. TRUMP's chart extends to include both syntactic and semantic possibilities, explicitly representing the interdependencies among these candidates. This syntactic–semantic chart structure applies to sentence analysis as well as serving as a starting point for a lexical hypothesis in lexical acquisition.[40] A "walk" through this extended chart represents a candidate semantic interpretation. Because relationships between linguistic and conceptual structures are explicit, the program can easily prune paths from the structure as it obtains new information.

Figure 8 shows an example of a dynamic chart structure, which links possible linguistic interpretations to conceptual structures that depend on a particular interpretation. The nodes labeled `L1`, `L2`, etc. are linguistic assumptions that produce concepts such as `C1`, `C2`, and so forth.

The extended chart structure permits some special ways of dealing with ambiguity, including eliminating inconsequential ambiguity, taking advantage of ambiguity, and delaying ambiguity handling. The examples that follow illustrate some of these methods.

2. Conceptual Structure Sharing

Resolving syntactic ambiguities is not always desirable in a semantic interpreter. These ambiguities may become inconsequential, or the application may require a greater degree of ambiguity. Compound nominals can be some of the most syntactially ambiguous constructs, but actually seem to require only lim-

PORTLAND STATE UNIVERSITY LIBRARY

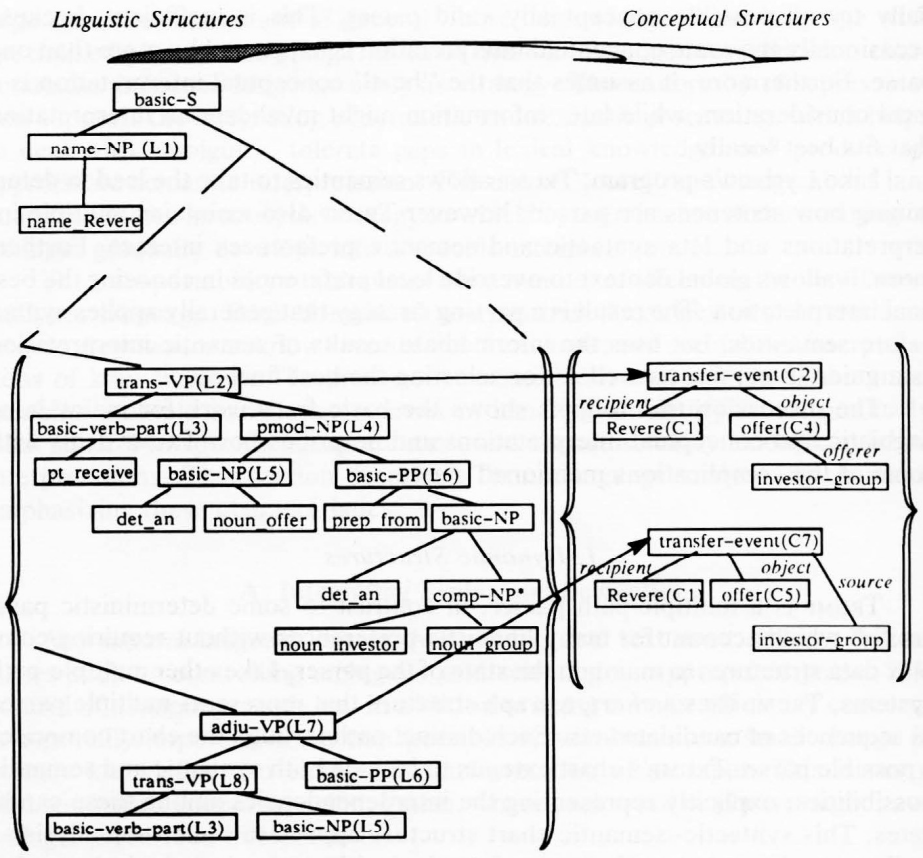


Figure 8. A dynamic syntactic–semantic chart.

ited syntactic processing. For this reason, some systems delay or avoid handling ambiguity, practicing "ambiguity procrastination."[41] An alternative method in machine translation systems reflects certain ambiguities in an internal meaning representation, so that the final translation can preserve the ambiguities. TRUMP applies each of these techniques selectively using *structure sharing*—replacing multiple segments of the linguistic charge with a single structure, thus "pretending" that the ambiguity does not exist.

Figure 9 illustrates the effect of structure sharing on the dynamic chart. In the figure, two concepts (C2 and C7) have the same conceptual interpretation, allowing TRUMP to collapse the verb phrases L2 and L7 to a single node.

Conceptual structure sharing occurs frequently in many domains in which TRUMP operates; for example, in the area of financial news, "1987 budget deficit" and "binding contractual obligations" cause structure sharing because the parse does not affect the meaning interpretation in either case (e.g., a deficit in the 1987 budget is the same as a budget deficit in 1987). One of the first sen-

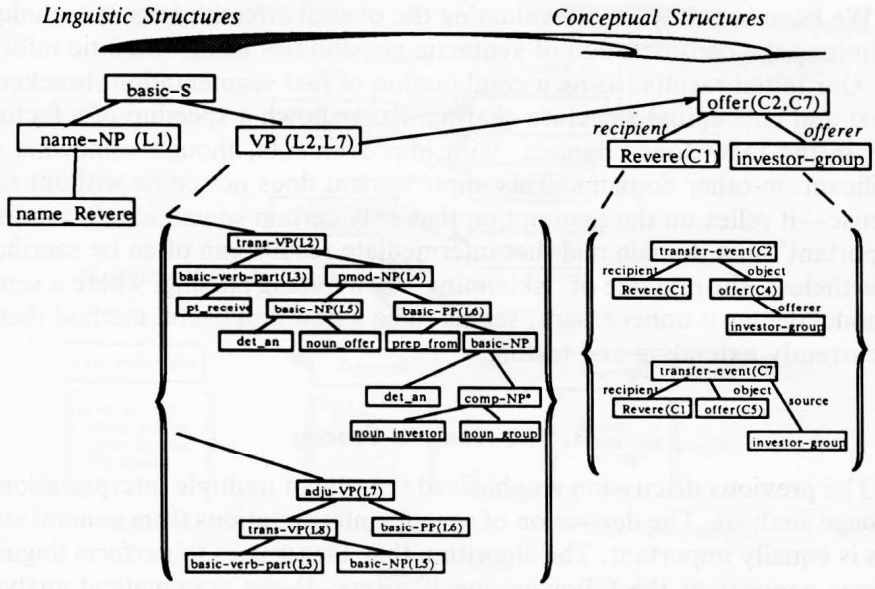


Figure 9. Joining structures with similar conceptual interpretations.

tences TRUMP, in the SCISOR application, processed from the Dow Jones service, was the following:

REVERE SAID IT HAD RECEIVED AN OFFER FROM AN INVESTOR GROUP TO BE ACQUIRED FOR $16 A SHARE, OR ABOUT $127 MILLION.

The above is not an unusually complex sentence, but it has dozens of syntactically reasonable parses. The phrase RECEIVED AN OFFER FROM AN INVESTOR GROUP, however, has two parses that are conceptually equivalent ([RECEIVED AN OFFER] [FROM AN INVESTOR GROUP] and [RECEIVED] [AN OFFER FROM AN INVESTOR GROUP]). The phrase FOR $16 A SHARE can attach either to TO BE ACQUIRED or to AN OFFER without consequence; TRUMP will even derive the same interpretation if it attaches to RECEIVED. Conceptual structure sharing in this example thus reduces the number of parses considered *by a factor of six* by identifying linguistic structures with shared meaning.

An alternative approach to ambiguity, which would be just as efficient, would be to select one parse instead of determining that the choice was inconsequential. This would work in the above cases, but would fail to identify ambiguities where *preservation* is desirable. For example, in the DEUCE database interface,[11] a user asks "What's the mail room number," which could mean the phone number of the mail room, the room number of the mail room, or the room number of the mail. Since the correct response depends on the information in the database, it is important for the analyzer to preserve the ambiguity until it finds an answer.