Concretion: Assumption-Based Understanding

Paul S. Jacobs
Artificial Intelligence Program
GE Research and Development Center
Schenectady, NY 12301 USA

Abstract

A language understanding program must produce as precise a meaning representation as possible from a linguistic input. CONCRETION is the process of developing a specific interpretation by combining various levels of conceptual information. This process represents an assumption-based method of language interpretation, and departs from the traditional approach of treating multiple interpretations as independent. Concretion allows the language analyzer to develop a sufficiently specific representation without excessive computation or brittle interpretation rules.

1 Introduction

The ambiguity and imprecision of language are the key problems in building language understanding programs. Most systems that perform semantic interpretation [Bobrow and Webber, 1980, Sondheimer *et al.*, 1984, Lytinen, 1984, Hirst, 1987] address this imprecision by offering means of selecting among alternative interpretations. The problem with these approaches is that they fail to take into account the interrelationships among the interpretations, which often support or refute one another to various degrees. A better model is one in which the candidates exist not as distinct choices but as assumptions contributing to a complete meaning representation. The language understanding process thus gradually refines a semantic representation based on the support or refutation of each element.

For example, consider the following potential inputs:

1. John *cut* the salami.
2. John *gave a kiss* to Mary.
3. The investor group *seeking control* of Warnaco...
4. The 'rm' command *takes* three arguments.
5. *Move back* to the last position.

The examples above represent potential inputs from several disparate real and "toy" domains of TRUMP (TRansportable Understanding Mechanism Package)[Jacobs, 1986b, Jacobs, 1987]. The idea of TRUMP is to use a combination of "core" knowledge about language and certain specialized knowledge to produce a complete semantic interpretation. In each of the examples, the italicized word or phrase represents a vague, ambiguous, or metaphorical verb sense. The problem for a good semantic interpreter is to derive the real or intended sense of each phrase without excessive computation or specialized knowledge. For example, the following are reasonable paraphrases of a complete semantic interpretation of the above examples:

1. John *cut* the salami
   cut ⇒ *sliced* (NOT *chopped* or *shortened*)
2. John *gave a kiss* to Mary
   gave a kiss ⇒ *kissed* (NOT *presented*)
3. The investor group *seeking* control of Warnaco...
   seeking ⇒ *trying* (NOT *searching*)
4. The 'rm' command *takes* three arguments.
   takes ⇒ *requires as input* (NOT *moves*)
5. *Move back* to the last position.
   Move back ⇒ *return* (NOT *move backwards*)

Each of these examples represents a clear, ordinary use of language. Yet a semantic interpreter must use a great deal of knowledge to distinguish the intended sense of the italicized phrase from other related and competing senses. *It is simply not practical to treat this process as one of discriminating among a large set of distinct interpretations.* The space of intended meanings is too large, and there are too many common characteristics of various senses. To deal effectively with the complexity of this process, a semantic interpreter must accomplish the following:

1. Identify prospective interpretations—The system must use linguistic information to select interpretations that are consistent with the input.
2. Use linguistic and conceptual knowledge to combine interpretations–This may result in ruling out certain candidates, or in forming new and more precise interpretations from the combination of knowledge sources.
3. Assume a specific interpretation—As in the above examples, a practical understanding of the input must be somewhat more than the maximum that can be "safely" inferred. The system must produce some knowledge structures that are likely candidates but are not certain from the linguistic description.
4. Fail gracefully on contradictions—If an assumed interpretation results in a contradiction, the system must preserve those interpretations that do not conflict. If other interpretations are dependent on a conflicting one, these too must be discarded.

The requirements above suggest a model of language understanding that progressively refines a semantic interpretation, based on linguistic and contextual information, but that incorporates into each specific interpretation knowledge upon which that interpretation builds. In other words, the ultimate goal of the system is to produce the most specific consistent interpretation, and the means of achieving that goal is to treat each interpretation as an assumption.

This assumption-based interpretation process is known as CONCRETION * [Wilensky, 1983]. The idea of concretion is to

*This term was originally proposed by Joe Faletti. The problem of concretion was initially defined in a series of seminars conducted by Robert

MICROSOFT CORP.
EXHIBIT 1030

determine as specific a meaning as is possible from an input, while enabling recovery if this interpretation proves overly specific. This process is the essential element of a framework that satisfies the criteria mentioned above.

Concretion is an important method for dealing with the problem of vagueness and imprecision as framed above. A system that performs concretion can successfully produce a complete interpretation without overcommitting to such an interpretation. The discussion that follows describes the concretion process as implemented in TRUMP and considers how this technique improves upon previous approaches.

2 Concretion

Concretion is the process of taking abstract concepts and producing from them concepts that are more precise, or *concrete*. The motivation for this mechanism is strong in story understanding [Norvig, 1983, Wilensky, 1983], because understanding a story seems to involve a continuous refinement of the major concepts into more specific categories. Concretion does not really involve inference, since often the specific meaning is quite explicit in the text. The process of concretion is evident in understanding simple words and phrases in limited linguistic contexts as well, as illustrated in the examples presented earlier.

Concretion is important because it is the mechanism that allows general knowledge about language to apply at very specific levels of semantic interpretation. This is essential for natural language interfaces and well as text processing systems, because it allows a core of linguistic and conceptual knowledge to be used for a variety of domains, and makes the addition of domain-specific linguistic knowledge easier. For example, knowledge about verbs such as *give* and *take* and their relation to *transfer-events* applies in discussing operating systems or corporate takeovers as well as in more general applications. It is hard to see how portability can be achieved without the capability to entertain a range of interrelated meaning representations.

A typical natural language input can test several aspects of the concretion process. In example 3, *the investor group seeking control of Warnaco*, A first-pass semantic analysis derives a *seeking* action—The investor group is the searcher, and control of Warnaco is being sought. Domain-independent conceptual knowledge suggests that looking for a state means trying to realize the state. Domain-specific knowledge produces the assumption that the phrase describes a corporate takeover attempt. An interpretation of this specificity is necessary to drive inferences and cooperative responses.

The concretion process is illustrated in figure 1. Each stage described above, and each knowledge source, must be distinct. The surface level semantic analysis is essential because it derives conceptual relations that would apply also to "looking for", "searching for", and even "pursuing"; this analysis thus avoids the redundant representation of each construct. This intermediate analysis also makes it possible to use abstract conceptual roles (such as *actor* and *recipient*) to determine specific underlying roles (such as the *target* and *suitor* of a corporate takeover) (cf. [Jacobs, 1987]). The second aspect of concretion, applying conceptual knowledge to produce

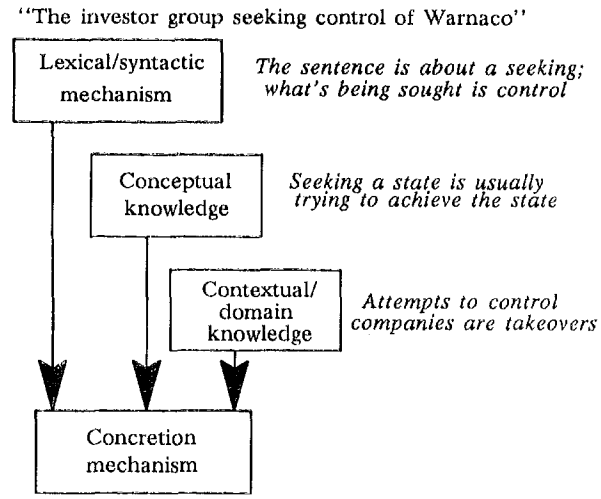

Figure 1: Concretion integrates knowledge sources

more specific interpretations, is necessary to refine vague terms and identify metaphorical or other non-literal constructs. The third component, using domain-specific knowledge, separates this general conceptual knowledge from assumptions that depend on an implied context, the domain of corporate takeovers in this example.

2.1 Types of Concretion

Concretion is the specialization of an abstract concept to a more precise interpretation in a given context. As the examples of the previous sections illustrate, concretion can involve a combination of linguistic and conceptual knowledge, and can result in either a direct specialization or a metaphorical extension. In all cases, concretion requires four ingredients:

- An instantiated concept to be specialized
- A linguistic or conceptual "trigger"
- A target concept type
- A conceptual relation between source and target

For example, in *John cut the salami*, the concept to be concreted is *cutting*, the trigger is the combination of *cutting* with *edible-obloid* or some such, the target concept is *slicing* (indicated by the trigger), and the relation is subcategorization, or DOMINATEs.

Concretion is often triggered by a linguistic structure, such as a particular combination of lexical items or the use of a certain phrase structure. Figure 2 shows the concretion of the concept *cutting* to the concept *severing* in the phrase *cut the tip off*. In this case, the knowledge base contains the information that *severing* is DOMINATEd by *cutting*. The DOMINATE relation, labeled *D*, indicates that *severing* is a subcategory of *cutting* [Wilensky, 1984]. Diagonal links labeled with italicized words show relationships between concepts and conceptual roles. The specialization of *cutting* in this instance is triggered by the linguistic relation *v_part-cut_off*, representing the use of the verb *cut* with the particle *off*.

A specific interpretation is often reached through a series of concretions, as in the *seeking control* example. In the case of *cut*, the assumed interpretation can be further specialized if the conceptual object is of a particular type, such as *body_part*, as

Wilensky at the University of California at Berkeley. In addition to Wilensky, Faletti, and the author, participants in these seminars included Yigal Arens, Margaret Butler, David Chin, Chuck Fillmore, Paul Kay, Marc Luria, Jim Martin, Jim Mayfield, Peter Norvig, Lisa Rau, and Nigel Ward.

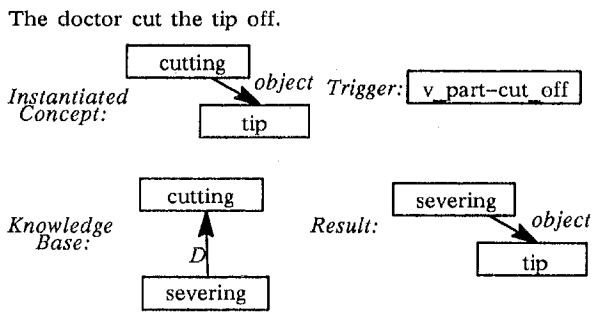

Figure 2: Concretion with Linguistic Trigger

shown in figure 3. In this example, the concept of *amputating* is reached through a combination of linguistic and conceptual clues.

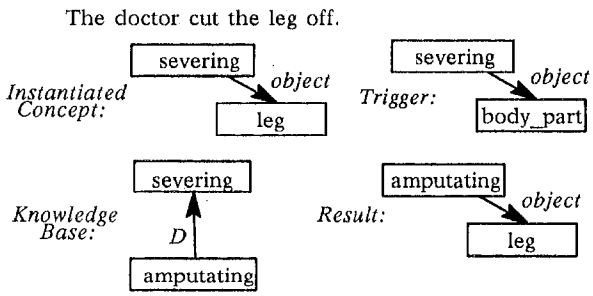

Figure 3: Concretion with Conceptual Trigger

An intended meaning is not necessarily a subcategory of an intermediate interpretation, as shown by the *seeking control* example. Associations between concepts that are analogous or metaphorically related are represented as VIEWs[Jacobs and Rau, 1985, Jacobs, 1986a], which also associate related roles. Figure 4 illustrates the application of a VIEW in the concretion process. In this example, the use of the noun *kiss* to describe the conceptual object of a *giving* serves as a trigger. A general VIEW of *action* as *transfer-event* relates *giving* (which is a *transfer-event* to *kissing* (which is an *action*), and also the *recipient* of the *giving* to the conceptual object of the action. The *m* label (for MANIFEST) on some roles indicates that the roles are not necessarily specializations of any more abstract relations. When used in this concretion, the VIEW derives a *kissing* concept with Mary as the *kissee*. There are two conceptual objects in a kiss—the person being kissed and the surface being kissed—and the concretion mechanism must use constraints on these roles to determine the correct role.

The above examples show several ways in which concretion results in the creation of a specific interpretation using a combination of linguistic and conceptual relationships. The examples also demonstrate that different types of concretion can be combined, as in *cut the leg off*. In any of these cases, a slight variation in the input can negate the resulting interpretation. When this happens, the concretion mechanism retreats to intermediate structures, thus preserving as much of the semantic result as possible. In *the doctor cut the leg off accidentally* or *the doctor cut the leg off the table*, the system will preserve the *severing* interpretation. Each concretion, therefore, is an

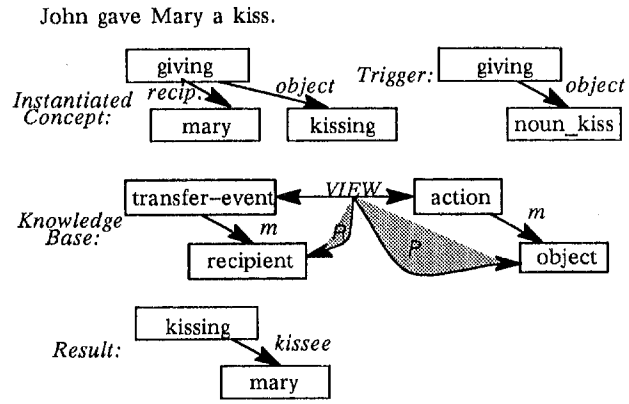

Figure 4: Concretion using VIEWs

assumption that explicitly depends on its trigger and on other consequential relationships.

2.2 Related Research

Most language analyzers do not really perform concretion. Unification-based systems [Pereira and Warren, 1980, Pereira and Shieber, 1984, Gawron *et al.*, 1982] tend to refine semantic representations by adding semantic FEATURES, represented as variables with assignments. Some of the systems that use a KL-ONE knowledge representation [Bobrow and Webber, 1980, Sondheimer *et al.*, 1984] perform a similar function, but use specific interpretation rules to place concepts in more specific categories, rather than to attempt an algorithm for combining lexical and conceptual knowledge. Hirst's[Hirst, 1987] "Polaroid Words" are described in a manner similar to concretion: the "words" gradually develop into a complete representation. However, each word sense is still independent; a "polaroid word" cannot, for example, conflate two word senses. Lytinen's MOPTRANS[Lytinen, 1984] includes a specialization mechanism that selects the most specific applicable frame, but like the Polaroid Words, it does not take into account the common assumed meaning. None of these approaches allows interpretations to be mapped or refined into more precise interpretations.

Other related research addresses the problem of concretion from a different perspective. Norvig's work [Norvig, 1986] concentrates on conceptual concretion as inference, independent of the linguistic issues. Zernik and Brown [Zernik and Brown, 1988] model language as a reason maintenance process, unlike TRUMP, which incorporates principles of reason maintenance within a more traditional linguistic framework.

3 The Concretion Algorithm

The discussion that follows describes the details of the concretion mechanism of the TRUMP semantic interpreter.

3.1 When to Concrete

Concretion is performed according to the following rules:

- *When a linguistic description produces a new conceptual structure, concrete it.*
  New pieces of conceptual knowledge are continually derived as the linguistic input is processed. When a new concept is produced, the new knowledge can interact with

existing knowledge to produce a more specific interpretation. For example, in "John cut the salami," a *cutting* with John as *cutter* is later concreted to a *slicing*, when the concept of *salami* is produced.

- *When a grammatical structure is completed, try to concrete the concepts to which it refers.*
  When a verb phrase is completed, for example, there may be a list of concepts to which the verb phrase potentially refers. Since the verb phrase can be enveloped by grammatical structures that necessarily refer to the same concepts, the conceptual knowledge produced from the verb phrase must be combined with any concepts produced from these enveloping structures. In "With a knife, John cut the salami", the meaning of the main clause is concreted using the prepositional phrase.

Summary: *Perform concretion whenever new conceptual information might result in a more specific semantic interpretation.*

3.2 How to Concrete

The concretion process is performed by taking two concepts and combining their conceptual content to produce a more specific concept. Generally, this results in filling out specific roles of the derived concept with more general role fillers. Concretion can also result in deriving a non-literal interpretation, as in the "give a kiss" and "take arguments" examples.

The concretion process often will fail, for example if the same role is filled by different concepts. In "Mary was given a letter to Bill", this blocks the possibility that Bill is the *recipient* and thus also resolves the attachment of the prepositional phrase (the correct interpretation is *A letter to Bill was given to Mary*). In other cases, concretion maps conceptual roles into new roles or subsumes roles altogether. In "give a kiss", the role of "kiss" as conceptual object disappears entirely in the concrete concept *kissing*. However, any modifiers of "kiss" become roles of the new concept, so "John gave Mary a quick kiss" is interpreted as "John kissed Mary quickly". In this case the relationship between the literal *giving* concept and the concrete *kissing* is called a VIEW.

The concretion mechanism keeps track of the linguistic structures upon which each concept is dependent. Thus if the parsing process later discards a linguistic structure that has violated a constraint, a bookkeeping mechanism can discard any concepts that hinge upon that structure. For example, the phrase "the command sent the message" has two parses, one in which the command, like 'rm', is sending a message, and the other in which the command is *receiving* the message. (One parse is a complete sentence, the second is a complex noun phrase.) In the garden path sentence, "the command sent the message halted", the semantic interpretation of the command playing the role of *sender* must be discarded for syntactic reasons. This resembles dependency-directed backtracking [Doyle, 1979, DeKleer, 1986], but is accomplished simply by constantly throwing away concepts that are no longer valid interpretations.

The input to the concretion algorithm in TRUMP is a concept C_{old} to be concreted, and a new piece of information C_{new}, also expressed as a concept. Concept C_{new} may come from a newly interpreted piece of linguistic information, a lexical or conceptual specialization rule, or a conceptual inference. The process operates as follows:

- *If C_{new} is a subcategory of C_{old}, form a concept of the same type as b, and merge C_{old}'s roles with C_{new}'s roles.*
  This results in a concreted concept of the more specific type, with a's roles converted to roles of the new type.
- *If C_{old} is a VIEW of C_{new}, proceed as if C_{new} were a subcategory of C_{old}, except use ROLE-PLAYs from the VIEW.*
  This is the metaphor application process: The result of concretion is a new concept of the type of b, but the roles must be filled according to the same VIEW that produced b.
- *If a role at any stage is filled by two different tokens, fail.*
  Concretion does not allow conflict in the filling of ROLE-PLAYs. If a VIEW results in the application of a ROLE-PLAY that potentially fills more than one role, conflict may be avoided. For example, in "Ali gave a punch to the jaw to Frazier", the two "to" phrases are allowed, but are automatically excluded from describing the same conceptual role.
- *If concretion results in the further specification of a VIEW, re-apply the VIEW.*
  This is another bookkeeping process. If a concept has already been concreted by a VIEW and then is further specified by concretion, the same VIEW is automatically applied. Thus, if a *transfer-event* is interpreted during concretion to produce an *execute-operation* and the *object* role of the *transfer-event* is later filled, the *input* role of the *execute-operation* is also filled.
- *If concretion results in violating a constraint, undo all concretions dependent on the concreted concept.*
  Since concepts that have already been concreted can continue to have their roles filled by concretion, it is possible that a violated constraint may eliminate many specific interpretations. These interpretations are treated as assumptions dependent on other concretions; thus keeping track of the dependencies allows assumed interpretations to be easily terminated based on new information.

Summary: *Produce the most specific conceptual interpretation of the input, with the appropriate roles filled, taking care to avoid conflicting interpretations.*

Many complex data structures and implementation issues are involved in the details of the above process. However, this presentation should suffice to describe how concepts are refined during semantic interpretation by applying conceptual and metaphorical knowledge.

4 Summary and Conclusion

Even within a specialized domain, a semantic interpreter must produce a meaning representation that is more specific than is necessarily implied by the linguistic input. This means that the understanding program must choose among a broad range of potential word senses, making it impractical to treat each as completely distinct. A better method is to make assumptions about the meaning of the input and proceed with processing from these assumptions. A major problem for the system is to select assumptions that are suggested by the input and to correct these assumptions elegantly in the event of conflicting information.

CONCRETION is the assumption-based part of semantic interpretation. The concretion mechanism described here selects the most specific concept triggered by the input, fills out the roles of this concept based on a variety of linguistic and conceptual information, and keeps track of the assumptions

upon which the concept is dependent. If a contradiction is found, either in filling out the roles or in later stages of understanding, the system is able to use these dependencies to correct the interpretation without excessive processing. This assumption-based model allows processing of specialized interpretations while permitting easy correction of assumptions that prove invalid.

References

[Bobrow and Webber, 1980] R. Bobrow and B. Webber. Knowledge representation for syntactic/semantic processing. In *Proceedings of the National Conference on Artificial Intelligence*, Palo Alto, California, 1980.

[DeKleer, 1986] J. DeKleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28(1), 1986.

[Doyle, 1979] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3), 1979.

[Gawron *et al.*, 1982] J. M. Gawron, J. King, J. Lamping, E. Loebner, A. Paulson, G. Pullum, I. Sag, and T. Wasow. The GPSG linguistics system. In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, Toronto, Ontario, 1982.

[Hirst, 1987] G. Hirst. *Semantic Interpretation and the Resolution of Ambiguity.* Cambridge University Press, Cambridge, England, 1987.

[Jacobs, 1986a] Paul S. Jacobs. Knowledge structures for natural language generation. In *Proceedings of the Eleventh International Conference on Computational Linguistics*, Bonn, Germany, 1986.

[Jacobs, 1986b] Paul S. Jacobs. Language analysis in not-so-limited domains. In *Proceedings of the Fall Joint Computer Conference*, Dallas, Texas, 1986.

[Jacobs, 1987] Paul S. Jacobs. A knowledge framework for natural language analysis. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987.

[Jacobs and Rau, 1985] Paul S. Jacobs and Lisa F. Rau. Ace: associating language with meaning. In Tim O'Shea, editor, *Advances in Artificial Intelligence*, pages 295–304, North Holland, Amsterdam, 1985.

[Lytinen, 1984] Steven Lytinen. *The Organization of Knowledge in a Multi-lingual, Integrated Parser.* PhD thesis, Yale University, 1984.

[Norvig, 1983] P. Norvig. Six problems for story understanders. In *Proceedings of the National Conference on Artificial Intelligence*, Washington, D. C., 1983.

[Norvig, 1986] P. Norvig. *A Unified Theory of Inference for Text Understanding.* PhD thesis, University of California, Berkeley, Berkeley, CA, 1986. Computer Science Division Report UCB/CSD 87/339.

[Pereira and Shieber, 1984] F. Pereira and S. M. Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the Tenth International Conference on Computational Linguistics*, Palo Alto, California, 1984.

[Pereira and Warren, 1980] F. Pereira and D. H. D. Warren. Definite Clause Grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13, 1980.

[Sondheimer *et al.*, 1984] N. Sondheimer, R. Weischedel, and R. Bobrow. Semantic interpretation using KL-ONE. In *Proceedings of the Tenth International Conference on Computational Linguistics*, Palo Alto, 1984.

[Wilensky, 1983] R. Wilensky. Memory and inference. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, 1983.

[Wilensky, 1984] R. Wilensky. KODIAK – a knowledge representation language. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, 1984.

[Zernik and Brown, 1988] U. Zernik and A. Brown. Default reasoning in natural language processing. In *Proceedings of the Twelfth International Conference on Computational Linguistics*, Budapest, Hungary, 1988.