Beyond Semantic Ambiguity[1]

Galina Datskovsky Moerdler and Kathleen R. McKeown
Columbia University
Department of Computer Science
New York, N.Y. 10027.

Abstract:

An advice giving system, such as an expert system gathers information from a user in order to provide advice. In this type of dialogue a single user statement or question may map into several facts of an underlying system, while several non consecutive statements may derive only one such fact. To support this type of interaction, a truly flexible natural language interface must be able to handle an extended notion of semantic ambiguity; it must avoid failure on producing partial semantic interpretations and be able to gather additional information for the interpretations from subsequent input. In this paper we describe a semantic mechanism that is able to handle this type of semantic ambiguity, while retaining other desirable properties of a general semantic interpreter.

1 Introduction

Dialogue between advice givers and advice seekers is likely to involve a certain amount of give and take. Advice seekers may provide background information about their situation and may ask questions indicating the kind of advice they need. Advice givers, in turn, may ask questions to clarify the advice needed or gather further information needed to determine the advice to provide. In responding to these questions, the advice seeker may opt to provide additional unrequested information deemed relevant. Providing this sort of flexibility as part of a natural language interface to an underlying expert system raises a number of special challenges.

In previous papers, we have presented a representation that can be used for semantic interpretation given the unstructured nature of expert system rule bases [Datskovsky Moerdler 88] and have shown how to use that representation to derive both background information (i.e., facts) and desired advice (i.e., expert system goals) from a *single* user question [Datskovsky Moerdler et.al. 87]. Given the back and forth nature of advice seeking dialogue, however, information pertaining to a single system fact may be provided by *several user statements* and these statements may not always occur consecutively in conversation. To provide true flexibility, then, a natural language interface to an expert system must be able to handle an extended notion of semantic ambiguity; it must avoid failure on producing a partial semantic interpretation and be able to gather additional information for the interpretation from subsequent input. In fact, this ability is crucial to the success of the NI interface. In this paper, we present a mechanism for handling extended semantic ambiguity, describe its interaction with our semantic interpreter, and show its implementation as part of a natural language interface we have developed for a tax advising expert system.

2 Description of the Semantics

Our semantic interpreter consists of a group of hierarchies that are formed from verb categories. Currently, we have 14 categories, derived from work in linguistics and from analysis of Roget's Thesaurus. The hierarchies form a connected forest that resides on top of an underlying expert system. The top node of a hierarchy corresponds to a general verb category, while many of the lower nodes are derived from properties of verbs with more specific meanings. For example, consider the *Transfer of Possession* hierarchy shown in Figure 1. The top node is derived from general verbs such as *give* or *receive*. A verb such as *pay*, however, being more specific in meaning, would point to the *monetary* node. If a verb has multiple meanings, it can belong to more than one category. For example, consider the verb *to support*. It can imply financial support, as in *I support my father by paying his rent and food bills*, or it can mean ideological support, as in *I support my father because I think he is right*. Thus, it can be a member of at least two categories, *Transfer of Possession* and *Relationship*.

The meaning of a verb in context is disambiguated based on the features of other elements of the sentence. Therefore, each node of our hierarchies has a set of selectional restrictions on the agent, patient, object and modifier roles attached to it. These restrictions are derived from the features of nouns, adjectives and other parts of speech. Each child inherits the restrictions of its parent. Often, not all case role restrictions can be specified at the higher nodes of the hierarchies, and are put off until the lower levels.

When parsing a sentence, syntactic processing is initiated first using an ATN parser [Woods 73; Woods 70]. As soon as the main verb is found, an appropriate semantic hierarchy is selected based on the definition of that verb in the dictionary[2]. The hierarchy is then partially traversed, with

[1]This research was partially supported by Office of Naval Research grant N00014-82-K-0256.

[2]The definitions are ordered in such a way that the meaning most common to the domain is tried first.

selectional restrictions guiding the parsing algorithm down the hierarchy. A child of a given node is selected if certain of the restrictions are appropriately filled. Syntax is called when a missing restriction is unfilled, and not all of the sentence is syntactically processed. Because semantic objects can be implied by nouns, certain noun phrases can also point to low level nodes in the semantic hierarchies.

We are testing our semantic formalism as a front end to a small expert system, Taxpert [Ensor et. al. 85], that deals with Tax code issues. In particular, it helps a user determine whether he can or can not claim someone as a dependent. In this context, leaf nodes of the hierarchies point to propositions used by Taxpert's rule base. Two such propositions are shown in Figure 1[3]. These are facts that Taxpert may need to determine whether the user can claim a dependent and represent the dependent's gross income and amount of support received. Variables in the propositions, indicated by question-marks (?), are filled from user input.

The task for the natural language interface is to map natural language statements and questions into one or more of the system propositions by traversing the semantic hierarchies. Problems arise, however, when a single utterance does not derive a proposition, but only allows for partial traversal of a hierarchy. Such utterances are defined as *semantically ambiguous.*

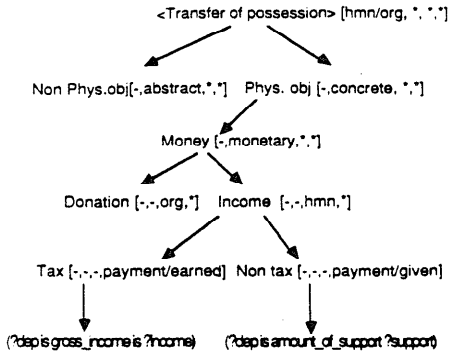

Figure 1: A Partial Hierarchy for the Transfer of Possession category[4].

2.1 Dealing with an Extended Notion of Semantic Ambiguity.

To process semantically ambiguous input, the system must know when one sentence completes a previous one and when it is simply another incomplete sentence. It must decide which sentence determines the hierarchy used and which sentence provides additional information to guide the system down that hierarchy. It must also decide how long to wait before asking the user for additional information not provided in the paragraph, or whether this information can be guessed based on context.

Any sentence that does not derive a proposition is placed on a stack and stored there until a sentence that may potentially complete or advance it further down the hierarchy is encountered. We differentiate between two types of sentences that can potentially complete other sentences already on the stack: those that are processed in the same hierarchy, and those that are processed in a different one.

If a new sentence in the same hierarchy is a candidate for completing a previous one, the two sentences are checked to find which semantic roles, such as agent, patient etc, are in agreement. The new semantic roles from the second sentence are used to attempt to complete traversal of the hierarchy and derive a proposition. If a proposition is not reached the new sentence is also stacked.

The second case, where two sentences complete each other, but are in different hierarchies is more difficult to handle. First, a sentence that is a candidate for completing another already on the stack must have an anaphoric reference to the stacked sentence. If no such reference exists, the second sentence is not even considered as a possible candidate. Another requirement is that the main verb of the sentence be a stative verb, as these can be used to describe an additional set of features for a semantic role of a previous sentence. Other verbs generally indicate that the sentence is providing a new piece of information and therefore should derive a separate proposition. Finally, we check whether one of the sentences that are considered as possible complements is not on a path to a proposition in which case it is likely to provide additional description that can be used in another sentence. If a sentence meets these requirements, the algorithm tries to use the information to complete traversal of the previous hierarchy. If a proposition is derived, the previously stacked sentence is popped off. However, if a proposition is not derived, then as in the first case, both sentences are stacked.

2.1.1 Example

Suppose the user initially inputs a paragraph of background information as shown in figure 2. Appendix I shows how the system processes this input. The first sentence of the paragraph contains three separate pieces of information. The verb *to support* has multiple meanings, but since *Transfer of Possession* is its most common meaning in this domain, the *Transfer of Possession* hierarchy, shown in figure 1, is tried first. Since the meaning is very specific and indicates the transfer of non taxable income, *support* points directly to the node *Non_Tax* in the hierarchy. However, because it is essential to know the amount of support in order to derive a complete proposition, processing stops there and this part of the sentence is stacked as incomplete (steps 1-4 of trace). However, two complete proposition are derived from the first sentence, one from the noun phrase *my daughter* and the other from the relative clause. The noun *daughter* points to the *Daughter* node of *Possession* hierarchy, partially shown in

[3]Because we used a preexisting expert system, we did not in any way modify the form of its propositions.

[4]In the figure, * stands for wild card, and - means that the feature is inherited from the parent node.

figure 5. Before filling the variables, the system checks to make sure that the possessive *my* agrees with the head pronoun, in this case *I*. The variables are then filled and the proposition (dependent is daughter_of user) is derived as shown in statement 3 of the trace. The main verb of the relative clause is the verb *To Be*, which points to the *Status* hierarchy shown in figure 3. It is traversed based on the features of the other elements of the sentence and the proposition (daughter is_age 18) is derived (steps 5-10 of trace).

I support my daughter, who is 18.
I give her 8000 dollars, but she also earns a salary.
Can I claim her if her salary is 1000 dollars?

Figure 2: A typical paragraph of input.

The first part of the second sentence, *I give her 8000 dollars* is processed in the *Transfer of Possession* hierarchy. No proposition is derived, because it is not specified whether the income is taxable or not. The algorithm then tries to complete the statement already on the stack. Since it is a potential candidate, the system tries to combine the information in the two sentence. It checks for the agreement between the agents and patients of the two sentences, which is indeed verified, tries to complete the previous sentence, and derives the proposition (daughter is amount_of_support 8000), (steps 11-18 of trace). The second part of the sentence is also processed in the *Transfer of Possession* hierarchy; however, since it does not derive a proposition, it is likewise put on the stack (steps 19-24 of trace).

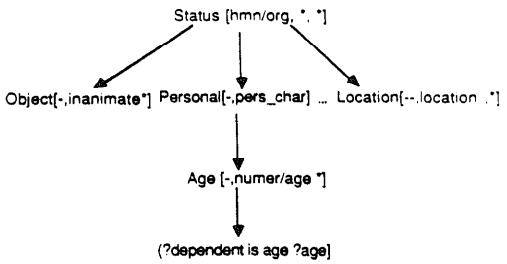

Figure 3: A Partial Hierarchy for the Status category.

Finally, the system considers the question *Can I claim her if her salary is 1000 dollars.* The main verb of a yes/no question generally indicates the goal. The verb *claim* is defined in the system's dictionary as *Classification, Dependency*[5], indicating that the verb belongs to the general category of *Classification* and a more specific subnode of that category, *Dependency*. Based on the definition of the verb the algorithm enters the *Classification* hierarchy shown in figure 4 at the *Dependency* node, and the proposition (user can_claim daughter) is derived as the goal, indicating that the user wants to know whether he can or can not claim a dependent, (steps 25-27 of trace). The relative clause of the second sentence, *Her salary is 1000 dollars*, does not derive a complete proposition. However, because its main verb is stative and it has a definite reference *her salary*, to the np *a salary* in the sentence currently on the stack, it is used to complete that sentence, and proposition (daughter is income 1000) is derived, (steps 28-36). Thus, the initial paragraph entered by the user provides not only the goal, i.e. the question the user wants answered, but also four pieces of additional information.

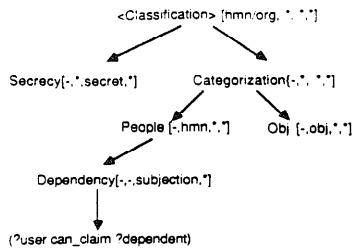

Figure 4: Partial Tree Formed for the *Clasification* category.

If any sentence were still to remain on the stack after the entire paragraph was processed, the system would ask the user to complete the missing information; however since there are no more input sentences and no information is left on the stack, the appropriate facts and goals will be passed to the working memory and the inference engine of the expert system respectively.

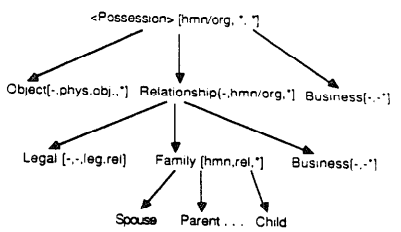

Figure 5: Partial Tree Formed for the *Possession* category.

2.2 When to Ask.

In order to decide when it is appropriate to ask for missing information, we assign each of the selectional restrictions one of three categories: obligatory, essential, and non-essential. Obligatory roles are those that are syntactically mandatory and so are always filled. Essential roles are syntactically optional, but must be filled for proper semantic processing. Finally, non-essential roles are both syntactically and semantically optional and therefore can be derived based

[5]Although there are other meanings of the verb, this is the most frequently used meaning in the tax domain, so the system tries this category first.

on previous input or domain knowledge[6]. If during tree traversal, an essential role is not filled at a given level of the tree, the system has to fill it before going on. This can be done by asking the user for further information, or by processing other sentences of a paragraph.

Obligatory, or syntactically mandatory roles are assigned in an obvious way. For example, if all the verbs that point to a given node in a hierarchy are transitive, then the restriction on the object is obligatory. Essential features include all features necessary to decide which proposition to derive. For example, the *payment/given* and *payment/earned* features in the Transfer of Possession hierarchy are essential, because they are used to decide whether the algorithm should select the node *Tax* or *Non_Tax*, which are both parents of propositions. Features which are used to fill the variable in a proposition are also essential, because without them a complete proposition can not be derived.

If a feature is often omitted without altering the meaning of a sentence, or can be guessed from either domain knowledge (i.e. there is a standard default that is generally assumed), or previous discourse, that feature is non essential. For example, it is not important to know who initiates a transaction in most sentences with Transfer of Possession verbs since most often this role can be derived from context, or a human actor is implied, so the feature *human/organization*, in the top node of the *Transfer of Possession* hierarchy is non essential.

When not specified, the non essential roles can be derived by the semantic formalism. However, this may lead the algorithm into the wrong subtree. Consider the sentence *I gave a donation to a University*. The parser has no way of knowing whether the donation is abstract (e.g. a copyright) or concrete, but follows the most probable meaning, i.e concrete. If the next input indicates that the donation was a copyright to a book, and it guessed the role incorrectly, it must back up to the point where the guess was made and start again. The same strategy applies when an incorrect parse tree was chosen altogether, i.e. when the verb is a member of more than one hierarchy. The most likely meaning is always taken first. If the choice was erroneous, the situation would be quickly detected because the essential roles will not match the features of the verb modifiers, so the algorithm will be forced to back up and try a different tree.

3 Comparison with Palmer's and Lytinen's work

Other work in semantics closely related to our own includes Palmer [Palmer 85] and Lytinen [Lytinen 84].

Palmer's inference driven semantic analysis was specifically designed for finite, well-defined, limited domains. Although we base our case role filling model on a modification of Palmer's, in our system, the user is always queried for the essential roles, while the non essential ones are guessed. Because our semantics is interactive, role filling can be more accurate and extend to a variety of domains. Palmer's semantic representation is in the form of predicates. There is only a limited hierarchy and no interaction with the user. Thus, she provides no mechanism for processing semantically incomplete input and is not able to handle ongoing dialogue.

Lytinen's work is in the area of machine translation. It specifically addresses the issue of word disambiguation through general disambiguation rules in conjunction with a hierarchically organized conceptual memory[7]. The greatest similarity between this work and our own is the hierarchical memory representation. However, Lytinen's hierarchy alone could not be used for parsing. It had to be used in conjunction with scripts and rules. Unlike our hierarchies, Lytinen's hierarchy is not based on properties of specific lexical items, such as verbs, but rather it is a memory representation of various events, and can therefore only recognize events encoded into it. Lytinen also provides no mechanism for combining semantic information from several sentence to provide interpretation.

4 Summary.

In this paper we presented a semantic representation that can handle an extended notion of semantic ambiguity. In particular, our interpreter is able to combine information from different points in the discourse to complete a partial semantic parse. Our system uses the set of verb hierarchies, a stacking mechanism and a matching algorithm that allows it to combine information from different semantically incomplete sentences. It uses a role classification model which helps the system process semantically ambiguous input and decide when to ask the user for more information.

The parser is implemented in Common Lisp on a Symbolics Lisp machine. It currently has a vocabulary of over 700 words and can process a variety of sentences and paragraphs. We are in the process of further increasing its vocabulary and capabilities. While the system can currently process yes-no questions, as well as statements, we also plan to implement WH question processing. We are also in the process of testing the generality of our approach by transporting it to a first order predicate logic planner.

5 Acknowledgments

We would like to thank the people at AT&T Bell Laboratories of Holmdel, New Jersey for their cooperation on this project.

[6]The original model is due to Palmer [Palmer 85], however it is extended in this work.

[7]This memory representation is an IS-A hierarchy of various concepts.

APPENDIX I:

(process '((I support my daughter who is 18) (I give her 8000 dollars but she also earns a salary) (can I claim her if her salary is 1000 dollars)))

verb: support

1."In:" TRANS_OF_POS

2."Considering children of " |NON_TAX|

3. "proposition from np " MY DAUGHTER (DEPENDENT IS |DAUGHTER_OF| USER)

can not complete proposition

4. "stack " (((TRANS_OF_POS + |NON_TAX|) |NON_TAX| ((I SUPPORT MY DAUGHTER)) ...

parsing 'who is 18'

5. "In:" STATUS

6. "Considering children of " STATUS

7. "Considering children of " PERSONAL

8. "Considering children of " AGE

9. " proposition " (DAUGHTER IS AGE 18)

10. "stack"

(((TRANS_OF_POS + |NON_TAX|) |NON_TAX| ((I SUPPORT MY DAUGHTER)) ...

processing second sentence:

verb: give

11. "In:" TRANS_OF_POS

12. "Considering children of " TRANS_OF_POS

13. "Considering children of " |PHYS_OBJ|

14. "Considering children of " MONEY

15. " STACK" (((TRANS_OF_POS + |NON_TAX|) |NON_TAX| ((I SUPPORT MY DAUGHTER)) ...

completing sentence already on stack with new information

16. "Considering children of " |NON_TAX|

17. "proposition" (DAUGHTER IS AMOUNT_OF_SUPPORT 8000)

18. " stack " NIL

processing 'but she also earns a salary'

19. "In:" TRANS_OF_POS

20. "Considering children of" TRANS_OF_POS

21. "Considering children of " |PHYS_OBJ|

22. "Considering children of " MONEY

23. "Considering children of " INCOME

24. " stack " (((TRANS_OF_POS -) INCOME ((I GIVE HER 8000 DOLLARS)) ...

processing final question:

25. "In:" CLASSIFY

26. "Considering children of " DEPENDENCY

27. "proposition" (USER |CAN_CLAIM| DAUGHTER)

Verb 'is'

28. "In:" STATUS

29. "Considering children of " STATUS

30. "Considering children of " OBJECT

completing sentence already on stack with new information:

31. "Considering children of" INCOME

32. "Considering children of" TAX

33. "proposition" ((DAUGHTER IS |GROSS_INCOME| 1000))

34. "stack" NIL

35. "GOAL:" (USER |CAN_CLAIM| DAUGHTER)

References

[Ballmer et. al. 81] Levelt, W.J.M. (editor). *Springer Series in Language and Communication, volume 8: Speech Act Classification.* Springer-Verlag, 1981.

[Datskovsky Moerdler 88] G. Datskovsky Moerdler. Structure from Anarchy: Meta Level Representation of Expert System Predicates for Natural Language Interfaces. In *Proceedings of the Second Conference on Applied Natural language Processing.* 1988.

[Datskovsky Moerdler et.al. 87] G. Datskovsky Moerdler, K. McKeown, J.R. Ensor. Building Natural Language Interface to Expert Systems. In *Proceedings of the IJCAI.* 1987.

[Ensor et. al. 85] Ensor, Gabbe and Blumenthal. Taxpert -- A Framework for Exploring Interactions Among Experts. 1985.in preparation.

[Hirst 83] Hirst, G. *Semantic Interpretation Against Ambiguity.* PhD thesis, Brown University, 1983.

[Levin 83] Levin, B. *On the Nature of Ergativity.* PhD thesis, MIT, 1983.

[Levin 85] Levin, B. Lexical Semantics in Review: An Introduction. In Levin, B. (editor), *Lexical Semantics in Review.* MIT, 1985.

[Lytinen 84] Lytinen S.L. *The Organization of Knowledge in a Multi-lingual Integrated Parser.* PhD thesis, Yale University, 1984.

[Lytinen 86] Lytinen,S. A More General Approach to Word Disambiguation. *Experience, Memory, and Reasoning.* LEA, 1986.

[Miller 72] Miller, G.A. English Verbs of Motion: A Case Study in Semantics and Lexical Memory. *Coding Processes in Human Memory.* V.H. Winston and Sons, 1972.

[Osgood 79] Osgood, Charles, E. *Focus on Meaning Volume I: Explorations in Semantic Space.* Mouton Publishers, 1979.

[Palmer 83] Palmer, M. Inference-Driven Semantic Analysis. In *Proceedings of the AAAI.* 1983.

[Palmer 85] Stone Palmer, M. *Driving Semantics for a Limited Domain.* PhD thesis, University of Edinburg, 1985.

[Schank 75] Schank, R . C. *Conceptual Information Processing.* North Holland, Amsterdam, 1975.

[Winograd 72] Winograd, T. Understanding Natural Language. *Cognative Psychology,* 1972.

[Woods 70] Woods, W.A. Transition Network Grammars for Natural Language Analysis. *Computational Linguistics,* 1970.

[Woods 73] Woods, W.A. An Experimental Parsing System for Transition Network Grammars. *Natural Language Processing.* , 1973.