A CASE FOR RULE-DRIVEN SEMANTIC PROCESSING

Martha Palmer
Department of Computer and Information Science
University of Pennsylania

0.0 INTRODUCTION

The primary task of semantic processing is to provide an appropriate mapping between the syntactic constituents of a parsed sentence and the arguments of the semantic predicates implied by the verb. This is known as the Alignment Problem. [Levin]

Section One of this paper gives an overview of a generally accepted approach to semantic processing that goes through several levels of representation to achieve this mapping. Although somewhat inflexible and cumbersome, the different levels succeed in preserving the context sensitive information provided by verb semantics. Section Two presents the author's rule-driven approach which is more uniform and flexible yet still accommodates context sensitive constraints. This approach is based on general underlying principles for syntactic methods of introducing semantic arguments and has interesting implications for linguistic theories about case. These implications are dicussed in Section Three. A system that implements this approach has been designed for and tested on pulley problem statements gathered from several physics text books. [Palmer]

1.0 MULTI-STAGE SEMANTIC ANALYSIS

A popular approach [Woods], [Simmons], [Novak] for assigning semantic roles to syntactic constituents can be described with three levels of representation - a schema level, a canonical level, and a predicate level. These levels are used to bridge the gap between the surface syntactic representation and the "deep" conceptual representation necessary for communicating with the internal database. While the following description of these levels may not correspond to any one implementation in particular, it will give the flavor of the overall approach.

1.1 Schema Level The first level corresponds to the possible surface order configurations a verb can appear in. In a domain of equilibrium problems the sentence

"A rope supports one end of a scaffold." could match a schema like "<physobj> SUPPORTS <locpart> of <physobj>". The word ordering here implies that the first <physobj> is the SUBJ and the <locpart> is the OBJ. Other likely schemas for sentences involving the SUPPORT verbs are "<physobj> SUPPORTS <physobj> AT <locpart>," "<physobj> SUPPORTS <force>," "<physobj> IS SUPPORTED," and "<locpart> IS SUPPORTED." [Novak] Once a particular sentence has matched a schema, it is useful to rephrase the information in a more "canonical" form, so that a single of inference rules can apply to a group of schemas.

1.2 Canonical Level This intermediate level of representation usually consists of the verb itself, (or perhaps a more primitive semantic predicate chosen to represent the verb) and a list of possible roles, e.g. arguments to the predicate. These roles correspond loosely to a union of the various semantic types indicated in the schemas. The schemas above could all easily map into:

SUPPORTS(<physobj>1,<physobj>2,
<locpart>,<force>).

The "canonical" verb representation found at this level bears certain similarities to a standard verb case frame, [Simmons, Bruce] in the roles played by the arguments to that predicate. There has been some controversy over whether or not any benefits are gained by labeling these arguments "cases" and attempting to apply linguistic generalities about case. [Fillmore] The possible benefits do not seem to have been realized, with a resulting shift away from explicit ties to case in recent work. [Charniak], [Wilks]

1.3 Predicate Level However, the implied relationships between the arguments still have to be spelled out, and this is the function of our third and final level of representation. This level necessarily makes use of predicates that can be found in the data base, and for the purposes of the program is effectively a "deep" semantic representation. A verb such as SUPPORT would require several predicates in an equilibrium domain. For example, the 'scaffold' sentence above could result in the following list corresponding to the general predicates listed immediately below.

'Scaffold' Example

SUPPORT(rope,scaffold)
UP(F1,rope)
DOWN(F2,scaffold)
CONTACT(rope,scaffold)
LOCPT(rtend1,rope)
LOCPT(rtend2,scaffold)
SAMEPLACE(rtend1,rtend2)

General Predicates

SUPPORT(<physobj>1,<physobj>2)
UP(<force>1,<physobj>1)
DOWN(<force>2,<physobj>2)
CONTACT(<physobj>1,<physobj>2)
LOCPT(<locpart>1,<physobj>1)
LOCPT(<locpart>2,<physobj>2)
SAMEPLACE(<locpart>1,<locpart>2)

Producing the above list requires common sense deductions [Bundy] about the existence of objects filling arguments that do not correspond directly to the canonical arguments, i.e. the two <locpt>s, and any arguments that were missing from the explicit sentence. For instance, in our scaffold example, no <force> was mentioned, and must be inferred. The usefulness of the canonical form is illustrated here, as it prevents tedious duplication of inference rules for slightly varying schemas.

The relevant information from the sentence has now been expressed in a form compatible with some internal database. The goal of this semantic analysis has been to provide a mapping between the original syntactic constituents and the predicate arguments in the final representation. For our scaffold example the following mapping has been achieved. The filling in of gaps in the final representation, although motivated by the needs of the database, also serves to test and expand the mapping of the syntactic constituents.

```
SUBJ <- rope                     <physobj>1
OBJ  <- end                      <physobj>2
OFPP<- scaffold                  <locpart>2
```

An obvious question at this point is whether or not the mappings from syntactic constituents to predicate arguments can be achieved directly, since the above multi-stage approach has at least three major disadvantages:

1) It is tedious for the programmer to produce the original schemas, and the resulting amount of special purpose code is cumbersome. It is difficult for the programmer to guarantee that all schemas have been accounted for.

2) This type of system is not very robust. A schema that has been left out simply cannot be matched no matter how much it has in common with stored schemas.

3) Because of the inflexibility of the system it is frequently desirable to add new information. Adding just one schema, much less an entire verb, can be time consuming. How much of a hindrance this will be is dependent on the extent to which the semantic information has been embedded in the code. The LUNAR project's use of a meaning representation language greatly increased the efficiency of adding new information.

The following section presents a system that uses syntactic cues at the semantic predicate level to find mappings directly. This method has interesting implications for theories about cases.

2.0 <u>RULE-DRIVEN SEMANTIC ANALYSIS</u>

This section presents a system for semantic processing that maps syntactic constituents directly onto the arguments of the semantic predicates suggested by the verb. In order to make these assignments, the possible syntactic mappings must be associated with each argument place in the original semantic predicates. For instance, the only possible syntactic constituent that can be assigned to the <physobj>1 place of a SUPPORT predicate is the SUBJ, and a <physobj>2 can only be filled by an OBJ. But a <locpart> might be an OBJ or the object of an AT preposition, as in "The scaffold is supported at one end." (The scaffold in this example is the syntactic subject of a passive sentence, so it is also considered the logical object. For our purposes we will look on it as an OBJ). It might seem at first glance that we would want to allow our <physobj>2 to be the object of an OF preposition, as in "The rope supports one end of the scaffold." But that is only true if the OFPP follows something like a <locpart> which can be an OBJ in a sentence about SUPPORT. (Of course, just any OFPP will not supply a <physobj>2. In "The rope supports the end of greatest weight.", the object of the OFPP is not a <physobj> so could not satisfy <physobj>2. The <physobj>2 in this case must be provided by the previous context.)

It is this very dependency on the existence of other specific types of syntactic constituents that was captured by the schemas mentioned above. It is necessary for an alternative system to also handle context sensitive constraints.

2.1 <u>Decision</u> <u>Trees</u> The three levels of representation mentioned in Section One can be viewed as the bottom, middle and top of a tree.

```
                  SUPPORT(p1,p2)
                  CONTACT(p1,p2)
                  LOCPT(lpt1,p1)
                  LOCPT(lpt2,p2)
                         |
                         |
                         |
           SUPPORT(p1,p2,lpt,force)
                       / | \
                      /  |  \
                         |
  SUBJ                  OBJ          OFPP
<physobj> SUPPORTS <locpart> OF <physobj>
```

"The rope supports one end of the scaffold."

The inference rules that link the three levels deal mainly with any necessary renaming of the role an argument plays. The SUBJ of the schema level is renamed <physobj>1 or p1 at the canonical level, and is still p1 at the predicate level.

One way of viewing the schemas is as leaf nodes produced by a decision tree that starts at the predicate level. The levels of the tree correspond to the different syntactic constituents that can map onto the arguments of the original set of predicates. Since more than one argument can be renamed as a particular syntactic constituent, there can be more than one branch at each level. If a semantic argument might not be mentioned explicitly in the syntactic configuration, this also has to be expressed as a rule, ex. p1 -> NULL. (Ex. "The scaffold is supported.") When all of the branches have been taken, each terminal node represents the set of decisions corresponding to a particular schema. (See Appendix A.) Note that the canonical level never has to be expressed explicitly. By working top down instead of bottom up unnecessary duplication of inference rules is automatically avoided.

The information in the original three levels can be stored equivalently as the top node of the decision tree along with the renaming rules for the semantic arguments (rewrite rules). This would reverse the order of analysis from the bottom-up mode suggested in section one to a top-down mode. This uses a more compact representation, but would be computationally less efficient. Growing the entire decision tree every time a sentence needed to be matched would be quite cumbersome. However, if only the path to the correct terminal node needed to be generated, this approach would be computationally competitive. By ordering the decisions according to syntactic precedence, and by using the data from the sentence in question to prune the tree WHILE it is being generated, the correct decisions can usuallly be made, with the only path explored being the path to the correct schema.

2.2 Context Sensitive Constraints Context sensitivity can be preserved by only allowing the p2->OFPP rule to apply after a mapping for lpt1 has been found, evidence that an lpt1->OBJ rule could have already applied. To test whether such a mapping has been made given a LOCPT predicate, it is only necessary to see if the lpt1 argument has been renamed by a syntactic constituent. The renaming process can be thought of as an instantiation of typed variables, - the semantic arguments - by syntactic constituents. [Palmer, Gallier, and Weiner] Then the following preconditions must be satisfied before applying the p2->OFPP rule: (/\ stands for AND)

```
p2->OFPP/ LOCPT(lpt1,p2)
     /\ not(variable(lpt1))
```

These preconditions will still need to be satisfied when a LOCPT predicate is part of another verb representation. Anytime a <locpart> is mentioned it can be followed by an OFPP introducing the <physobj> of which it is a location part. This relationship between a <locpart> and a <physobj> is just as valid when the verb is 'hang' or 'connect.' Ex. "The pulley is connected to the right end of the string." " The particle is hung from the right end of the string." These particular constraints are general to the domain rather than being restricted to 'support'. This illustates the efficiency of associating constraints with semantic predicates rather than verbs, allowing for more advantage to be taken of generalities.

There is an obvious resemblance here to the notation used for Local Constraints grammars [Joshi and Levy]:

```
p2->OFPP/ DOM(LOCPT) /\
     LMS(lpt1) /\ not(var(lpt1))

DOM - DOMinate,
LMS - Left Most Sister
```

It can be demonstrated that the context sensitive constraints presented here are a simple special case of their Local Constraints, since the dominating node is limited to being the immediate predicate head. Whether or not such a restricted local context will prove sufficient for more complex domains remains to be proven.

2.3 Overview As illustrated above, our mappings from syntactic constituents to semantic arguments can be found directly, thus gaining flexibility and uniformity without losing context sensitivity. Once the verb has been recognized, the semantic predicates representing the verb can drive the selection of renaming rules directly, avoiding the necessity of an intermediate level of representation. The contextual dependencies originally captured by the schemas are preserved in preconditions that are associated with the application of the renaming rules. Since the renaming rules and the preconditions refer only to semantic predicates and arguments to the predicates, there is a sense in which they are independent of individual verbs. By applying only those rules that are relevant to the sentence in question, the correct mappings can be found quickly and efficiently. The resulting system is highly flexible, since the same predicates are used in the representation of all the verbs, and many of the preconditions are general to the domain. This facillitates the addition of similar verbs since most of the necessary semantic predicates with the appropriate renaming rules will already be present.

3.0 THE ROLE OF CASE INFORMATION

Although the canonical level has often been viewed as the case frame level, doing away with the canonical level does not necessarily imply that cases are no longer relevant to semantic processing. On the contrary, the importance here of syntactic cues for introducing semantic arguments places even more emphasis on the traditional notion of case. The suggestion is that the appropriate level for case information is in fact the predicate level, and that most traditional cases should be seen as arguments to clearly defined semantic predicates.

These predicates are not merely the simple set of flat predicates indicated in the previous sections. There is an implicit structuring to that set of predicates indicated by the implications holding between them. A SUPPORT relationship implies the existence of UP and DOWN forces and a CONTACT relationship. A CONTACT relationship implies the existence of LOCPT's and a SAMEPLACE relationship between them. The set of predicates describing 'support' can be produced by expanding the implications of the SUPPORT(p1,p2) predicate into UP(f1,p1) and DOWN(f2,p2) and CONTACT(p1,p2). CONTACT(p1,p2) is in turn expanded into LOCPT(lpt1,p1) and LOCPT(lpt2,p2) and SAMEPLACE(lp1,lpt2). These definitions, or <u>expansions</u>, are represented as the following rewrite rules:

```
support<->SUPPORT(p1,p2)

SUPPORT(p1,p2)<->
     UP(f1,p1)/\DOWN(f2,p2)
     /\CONTACT(p1,p2)

CONTACT(p1,p2)<->
     LOCPT(lpt1,p1)/\LOCPT(lpt2,p2)
     /\SAMEPLACE(p1,p2)
```

When 'support' has been recognized as the verb, these rules can be applied, to build up the set of semantic predicates needed to represent support. If there were expansions for UP and DOWN they could be applied as well. As the rules are being applied the mappings of syntactic constituents to predicate arguments can be made at the same time, as each argument is introduced. The case information is not merely the set of semantic predicates or just the SUPPORT(p1,p2) predicate alone. Rather, the case information is represented by the set of predicates, the dependencies indicated by the expansions for the predicates, and the renaming rules that are needed to find the appropriate mappings. The renaming rules correspond to the traditional syntactic cues for introducing particular cases. They are further restricted by being associated with the predicate context of an argument rather than the argument in isolation.

When this structured case information is used to drive semantic processing, it is not a passive frame that waits for its slots to be filled, but rather an active structure that goes in search of fillers for its arguments. If these instantiations are not indicated explicitly by syntax, they must be inferred from a world model. The following example illustrates how the active case structure can also supply cases not mentioned explicitly in the sentence.

3.1 <u>Example</u> Given a pair of sentences like

"Two men are lifting a dresser. A rope supports the end of greatest weight."

we will assume that the first sentence has already been processed. Having recognized that the verb of the second sentence is 'support', the appropriate expansion can be applied to produce:

```
SUPPORT(rope,p2)
```

This would in turn be expanded to:

```
UP(f1,rope)
DOWN(f2,p2)
CONTACT(rope,p2)
```

In expanding the CONTACT relationship, an lpt1 for 'rope' and a p2 for 'end' need to be found. (See Section Two) Since the sentence does not supply an ATPP that might introduce an lpt1 for the 'rope' and since there are no more expansions that can be applied, a plausible inference must be made. The lpt1 is likely to be an endpoint that is not already in contact with something else.This implicit object corresponding to the free end of the rope can be name 'ropend2.' The p2 is more difficult. The OFPP does not introduce a <physobj>, although it does specify the 'end' more precisely. The 'end' must first be recognized as belonging to the dresser, and then as being its heaviest end, 'dresserend2.' This is really an anaphora problem that cannot be decided by the verb, and could in fact have already been handled. Given 'dresserend2', it only remains for the 'dresser' to be inferred as the p2 of the LOCPT relationship, using the same principles that allow an OFPP to introduce a p2. The final set of predicates would be

```
            SUPPORT(rope,dresser)
                     /|\
                    / | \
                   /  |  \
     UP(f1,rope)  |  DOWN(f2,dresser)
                      |
               CONTACT(rope,dresser)
                     /|\
                    / | \
                   /  |  \
LOCPT(ropend2,rope)LOCPT(dresserend2,dresser)
                      |
                      |
          SAMEPLACE(ropend2,dresserend2)
```

Both the ropend2 and 'dresser' were supplied by plausible reasoning using the context and a world model. There are always many inferences that can be drawn when processing a single sentence. The detailed nature of the case structure presented above gives one method of regulating this inferencing.

3.2 Associations with linguistics A recent trend in linguistics to consider cases as arguments to thematic relations offers a surprising amount of support for this position. Without denying the extremely useful ties between syntactic constituents and semantic cases, Jackendoff questions the ability of case to capture complex semantic relationships. [Jackendoff] His main objection is that standard case theory does not allow a noun phrase to be assigned more than one case. In examples like "Esau traded his birthright (to Jacob) for a mess of pottage," Jackendoff sees two related actions: "The first is the change of hands of the birthright from Esau to Jacob. The direct object is Theme, the subject is Source, and the to-object is Goal. Also there is what I will call the secondary action, the changing of hands of the mess of pottage in the other direction. In this action, the for-phrase is Secondary Theme, the subject is Secondary Goal, and the to-phrase is Secondary Source." [p.35] This, of course, could not be captured by a Fillmore-like case frame. Jackendoff concludes that, "A theory of case grammar in which each noun phrase has exactly one semantic function in deep structure cannot provide deep structures which satisfy the strong Katz-Postal Hypothesis, that is, which provide all semantic information about the sentence." Jackendoff is not completely discarding case information, but rather suggesting a new level of semantic representation that tries to incorporate some of the advantages of case. Making constructive use of Gruber's system of thematic relationships [Gruber], Jackendoff postulates "The thematic relations can now be defined in terms of [these] semantic subfunctions. Agent is the argument of CAUSE that is an individual; Theme is the argument of CHANGE that is an individual; Source and Goal are the initial and final state arguments of CHANGE. Location will be defined in terms of a further semantic function BE that takes an individual (the Theme) and a state (the Location).[p.39]

Indeed, Jackendoff is one example of a trend noted by Janet Fodor She points out that "it may be more revealing to regard the noun phrases which are associated in a variety of case relations with the LEXICAL verb as the arguments of the primitive SEMANTIC predicates into which it is analyzed. These semantic predicates typically have very few arguments, perhaps three at the most, but there are a lot of them and hence there will be a lot of distinguishable 'case categories.'(Those which Fillmore has identified appear to be those associated with semantic components that are particularly frequent or prominent, such as CAUSE, USE, BECOME, AT.)" [p.93] Fodor summarizes with, "As a contribution to semantics, therefore, it seems best to regard Fillmore's analyses as merely stepping stones on the way to a more complete specification of the meanings of verbs." The one loose end in this neat summation of case is its relation to syntax. Fodor continues, "Whether there are any SYNTACTIC properties of case categories that Fillmore's theory predicts but which are missed by the semantic approach is another question...."

It is the thesis of this paper that these syntactic properties of case categories are the very cues that are used to drive the filling of semantic arguments by syntactic constituents. This system also allows the same syntactic constituent to fill more than one argument, e.g. case category. The following section presents further evidence that this system could have direct implications for linguistic theories about case. Although it may at first seem that the analysis of the INSTRUMENT case contradicts certain assumptions that have been made, it actually serves to preserve a useful disctinction between marked and unmarked INSTRUMENTS.

3.3 The INSTRUMENT Case

The cases necessary for 'support' were all accomodated as arguments to semantic primitives. This does not imply, however, that cases can never play a more important role in the semantic representation. It is possible for a case to have its own expansion which contains information about how semantic predicates should be structured. There is quite convincing evidence in the pulley domain for the influential effect of one particular case.

In this domain INSTRUMENTS are essentially 'intermediaries' in 'hang' and 'connect' relationships. An <inter>mediary is a flexible line segment that effects a LOCATION or CONTACT relationship respectively between two physical objects. Example sentences are "A particle is hung by a string from a pulley," and "A particle is connected to another particle by a string." The following rewrite rules are the expansions for the 'hang' and 'connect' verbs, where the EFFECT predicate will have its own expansion corresponding to the definition of an intermediary.

hang <-> EFFECT(inter,LOCATION(pl,loc))

connect <-> EFFECT(inter,CONTACT(pl,p2))

Application of these rules repectively results in the following representation for the example sentences:

EFFECT(string,LOCATION(particle1,pulley1))

EFFECT(string,CONTACT(particle1,particle2))

The expansion of EFFECT itself is:

```
EFFECT(inter, REL(arg1,arg2)) <->
                    REL(arg1,inter),
                    REL(inter,arg2))
```

where REL stands for any semantic predicate. The application of this expansion to the above representations results in:

```
LOCATION(particle1,string)
LOCATION(string,pulley1)
```

and

```
CONTACT(particle1,string)
CONTACT(string,particle2)
```

These predicates can then be expanded, with LOCATION bringing in SUPPORT and CONTACT, and CONTACT bringing in LOCPT.

3.4 Possible Implications There seems to be a direct connection between the previous expansion of intermediary and the analysis of the INSTRUMENT case done by Beth Levin at MIT.[Levin] She pointed out a distinct difference in the use of the same INSTRUMENT in the following two sentences:

"John cut his foot with a rock."

"John cut his foot on a rock."

In the first sentence there is an implication that John was in some way 'controlling' the cutting of his foot, and using the rock to do so. In the second sentence there is no such implication, and John probably cut his foot accidentally. The use of the 'with' preposition marks the rock as an INSTRUMENT. that is being manipulated by John, whereas 'on' introduces an unmarked INSTRUMENT with no implied relationshion to John. It would seem that something like the expansion for EFFECT could help to capture part of what is being implied by the 'control' relationship. Bringing in the transitivity relationship makes explicit a connection between John and the rock as well as between the foot and the rock. In the second sentence only the connection between the foot and the rock is implied. The connection implied here is certainly more complicated than a simple CONTACT relationship, and would neccessitate a more detailed understanding of 'cut.' But the suggestion of 'control' is at least indicated by the embedding of the CUT predicate within EFFECT and CAUSE.

```
CAUSE(John,EFFECT(rock,(CUT(foot-of-John)))
```

The tie between the AGENT and the INSTRUMENT is another implication of 'control' that should be explored.

That the distinction between marked and unmarked INSTRUMENTS can be captured by the EFFECT relationship is illustrated by the processing of the following two sentences:

"The particle is hung from a pulley by a string."

"The particle is hung on a string."

In the first sentence an 'inter' (a marked INSTRUMENT) is supplied by the BYPP, and the following representation is produced:

```
EFFECT(string,LOCATION(particle,pulley))
```

In the second sentence no 'inter' is found, and in the absence of an 'inter' the EFFECT relationship cannot be expanded. The LOCATION(particle,string) predicate is left to stand alone and is in turn expanded. (The ONPP can indicate a 'loc.')

The intriguing possibility of verb independent definitions for cases requires much more exploration.[Charniak] The suggestion here is that a deeper level of representation, the predicate level, is appropriate for investigating case implications, and that important cases like AGENTS and INSTRUMENTS have implications for meta-level structuring of those predicates.

3.5 Summary In summary, there is a surprising amount of information at the semantic predicate level that allows syntactic constituents to be mapped directly onto semantic arguments. This results in a semantic processer that has the advantage of being easy to build and more flexible than existing processers. It also brings to light substantial evidence that cases should not be discarded but should be reexamined with respect to the roles they play as arguments to semantic predicates. The INTERMEDIARY case is seen to play a particularly important role having to do not with any particular semantic predicate, but with the choice of semantic predicates in general.

References

[1] Bruce, B., Case system for natural language, "Artificial Intelligence," Vol. 6, No. 4, Winter, pp. 327-360.

[2] Bundy, et-al, Solving Mechanics Problems Using Meta-Level Inference, Expert Systems in the Micro-Electronic Age, Michie, D.(ed), Edinburgh University Press, Edinburgh, U.K., 1979.

[3] Charniak, E., A brief on case, Working Paper No.22, (Castagnola: Institute for Semantics and Cognitive Studies), 1975.

[4] Fillmore, C., The case for case, Universals in Linguistic Theory, Bach and Harms (eds.) New York; Holt, Rinehart and Winston, pp. 1-88.

[5] Fodor, Janet D., Semantics: Theories of Meaning in Generative Grammar, Language and Thought Series, Thomas Y. Crowell Co., Inc., 1977, p. 93

[6] Gruber, J.S., Lexical Structures in Syntax and Semantics, North-Holland Pub. Co., 1976.

[7] Jackendoff, R.S., Semantic Interpreter in Generative Grammar, MIT Press, Cambridge, MA, 1972, p. 39.

[8] Levin, B. "Instrumental With and the Control Relation in English," MIT Master's Thesis, 1979.

[9] Novak, G.S., Computer Understanding of Physics Problems Stated in Natural Language, American Journal of Computational Linguistics, Microfiche 53, 1976.

[10] Palmer, M., Where to Connect? Solving Problems in Semantics, DAI Working Paper No. 22, University of Edinburgh, July 1977.

[11] Palmer, M., "Driving Semantics for a Limited Domain," Ph.D. Thesis, forthcoming, University of Edinburgh.

[12] Palmer, M., Gallier, J., and Weiner, J., Implementations as Program Specifications: A Semantic Processer in Prolog, (submitted IJCAI, Vancouver, August 1981).

[13] Simmons, R.F., Semantic Networks: Their Computation and Use for Understanding English Sentences, Computer Models of Thought and Language, Schank and Colby (eds.) San Francisco: W.H. Freeman and Co., 1973.

[14] Wilks, Y., Processing Case, "American Journal of Computational Linguistics," 1976.

[15] Woods, W.A., Semantics and Quantification in Natural Language Question Answering, BBN Report 3687, Cambridge, Mass, November 1977.

APPENDIX A

```
                         SUPPORT(p1,p2) /\ CONTACT(p1,p2) /\
                         LOCPT(lpt1,p1) /\ LOCPT(lpt2,p2)
                                  /                 \
        p1 -> SUBJ         /                          \     p1 -> NULL
                          /                            \
SUPPORT(SUBJ,p2) /\ CONTACT(SUBJ,p2)      SUPPORT(p1,p2) /\ CONTACT(p1,p2)
/\ LOCPT(lpt1,SUBJ) /\ LOCPT(lpt2,p2)     /\ LOCPT(lpt1,p1) /\ LOCPT(lpt2,p2)
              /          \                               /          \
p2 -> OBJ/                \  lpt2 -> OBJ                /            \
         /                 \                         etc.           etc.
SUPPORT(SUBJ,OBJ)         SUPPORT(SUBJ,p2) /\
/\ CONTACT(SUBJ,OBJ)      CONTACT(SUBJ,p2) /\
/\ LOCPT(lpt1,SUBJ)       LOCPT(lpt1,SUBJ) /\
/\ LOCPT(lpt2,OBJ)        LOCPT(OBJ,p2)
        |                            \
lpt2 -> ATPP                          \  p2 -> OFPP
          |                            \
SUPPORT(SUBJ,OBJ)                    SUPPORT(SUBJ,OFPP) /\
/\ CONTACT(SUBJ,OBJ)                 CONTACT(SUBJ,OFPP) /\
/\ LOCPT(lpt1,SUBJ)                  LOCPT(lpt1,SUBJ) /\
/\ LOCPT(ATPP,OBJ)                   LOCPT(OBJ,OFPP)
          |                                      \
          |                                       \
 SUBJ     |           OBJ           ATPP           \
<physobj> SUPPORTS <physobj> AT <locpart>           \
                                                     \
                                      SUBJ            \         OBJ           OFPP
                                     <physobj>   SUPPORTS   <locpart>   OF  <physobj>
```