Building Natural Language Interfaces for Rule-based Expert Systems

Galina Datskovsky Moerdler, Kathleen R. McKeown and J. Robert Ensor

Columbia University, New York, N.Y.
Columbia University, New York, N.Y.
AT&T Bell Laboratories, Holmdel, N J

ABSTRACT:

In this paper we discuss a semantics for translating natural language statements into facts of an underlying expert system, replacing the more conventional menu interface for gathering data from the user. We describe two issues that must be considered when building such an interface for an expert system. These issues are semantic processing of the user statements and the design of an interpreter for the expert system that efficiently utilizes the facts entered by the user. The semantic approach is based on verb categorization and hierarchical structuring of each category. The parsing algorithm based on selectional restriction is directly encoded into each verb class hierarchy. Next, we describe Director, an interpreter for rule-based expert systems that efficiently utilizes these facts for inferencing. Director uses a combination of forward and backward chaining that gives full consideration to each fact entered by the user and enables the system to process input in an efficient and focused manner.

1 Introduction

An expert system often interactively gathers data from a user in order to solve a problem. In many expert systems this information gathering is done via a menu interface. In this paper we describe a natural language interface for expert systems that replaces the more conventional menu interface for collecting information from the user. One of the tasks of a natural language interface is to translate user statements into facts of the underlying expert system, which requires a sophisticated semantics. Furthermore, such an interface places additional requirements on the inference engine of the expert system itself, namely to efficiently utilize the facts entered by the natural language module. This paper presents a semantic approach necessary for the translation as well as Director, an interpreter for rule-based expert systems** that meets the demands of the natural language interface. (For an overview of the system see figure 1.)

The semantic approach presented relies primarily on verb categorization and hierarchical structuring within each verb category. This approach differs from previous work in semantics by capturing linguistic generalizations of verb categories while providing modularity and some domain independence. In addition, it offers a semantic mechanism for an unstructured underlying system unlike previous approaches that made use of an assumed structure to define semantics (e.g. data base systems). During parsing, an appropriate hierarchy is selected according to the definition of the verb in the system's dictionary. A selectional restriction based algorithm is used to traverse the hierarchy. The restrictions on the arguments of the verb are based on the sentence noun features. Furthermore, in certain cases the noun features can be used directly in order to derive certain facts.

With a natural language interface, a user may volunteer information at any point; either at the beginning of a session or as an answer to any question posed by the expert system. The expert system must be able to use this information to arrive at a solution quickly, avoiding inferencing and associated questions that are irrelevant to the user specified goal and information. Director has been designed specifically to support the construction of expert system with natural language interfaces. It allows for facts entered through natural language to be taken into account by using an efficient combination of forward and backward chaining. Forward chaining is used to ensure that full consideration is eiven to each fact asserted by the natural language module on behalf of the user. Backward chaining is controlled by heuristics that ensure a focused interaction. These heuristics are based on descriptions of how and when data has been entered and guide the selection of rules to be evaluated. Director also provides ready access to portions of its internal structure. This feature allows the system to answer certain types of queries with a minimal number of searches of the rule base. The interpreter is designed to come to a solution quickly, with the minimal number of questions posed to the user and is therefore useful not only for systems with natural language interfaces, but also for systems whose problem domains involve data from hostile environments, such as nuclear reactors.

2 Background

In order for a human expert to be able to answer a person's question he often has to carry out extensive dialogs with that person to gather information about his needs. Extensive interaction and clarification is also needed for expert systems. One way expert systems communicate with their users is via a menu interface [Shortliffe 76] [Clancey 79]. Unfortunately, these interfaces are often tedious or awkward [Datskovsky 84] and may even limit the capabilities of associated expert systems [Pollack et. al. 82].

A natural language interface can relieve some of the

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

A typical rule-based expert system is constructed as a knowledge base and an associated interpreter (inference engine) [Hayes Roth 85]. The knowledge base is a collection of facts and production rules. A fact is a (name value) pair indicating the value of the object name; and a production rule is a statement of the form *IF premise THEN action.* Such a rule stores the knowledge that if the premise (left-hand-side) is true then the action (right-hand-side) should be performed The interpreter controls the execution of the expert system by selecting and evaluating rules. As these rules are evaluated, they alter the values of the facts and generate input and output problems associated with the menu interface by allowing the user to receive advice in the most informative and least time consuming way. That is, overall session length should be shorter using natural language since the system will have to pose fewer questions. This is possible because natural language allows the user to volunteer more than just the requested information whenever the expert system presents a question to the user.

The natural language module is responsible for interpreting incoming statements as facts. Although natural language interfaces to data base systems have been successfully constructed [Kaplan 79] [Woods et. al. 72] [Grosz ct al. 85], the task is more difficult in the expert systems domain. A semantic interpreter for a data base system usually relies on the regular structure of the data base as encoded in the schema describing it No such regularity or description is available in the expert systems' case. The lack of existing system structure makes the construction of domain independent semantics for expert systems difficult and means that some sort of structure that can be used as the basis for semantics must be built on top of the underlying rule base.

The semantics and control strategy we present in this paper is aimed at exactly this problem: interpretations of natural language responses to system questions, deriving and making use of any additional facts volunteered by the user. Since question asking is normally determined by the sequence of rule firings in expert systems, a control strategy was developed and fully implemented in Director that determines which rules to fire so as to minimize the number of questions and to ensure that they are asked in a focused and coherent order.

We are testing our ideas in the domain of tax and financial advising and using a small expert system called Taxpert [Ensor et al. 85], which deals with personal income tax matters as our experimental environment Taxpert consists of a number of agents that cooperate to solve an assortment of tax problems. Director, embodying our control strategy, serves as the control mechanism for the Dependency agent that helps the user determine whether someone qualifies as his dependent

A hypothetical example from the tax domain illustrating these issues is shown in Figure 2. Here the user is trying to determine whether an individual, Fred, can claim another individual, John. The system must determine whether 5 requirements are met in order to answer. These include the type of support given, relationship between the individuals, citizenship, income, and the type of return filed. The system asks whether Fred is the only supporter of John. The user not only supplies a *yes* as the answer to this question, but also volunteers extra information, *John is Fred's father.*

Since the user can enter any amount of information at arbitrary times with a natural language interface, the underlying expert system must be able to make use of

System: Is Fred the only supporter of John? [Y/N]
User: Yes, Fred is the only supporter of his father John.
Facts derived: (?user is only supporter of ?dependent)
(?dependent is parent_of ?user)

Figure 2: User - System Interaction

volunteered information to avoid asking unnecessary questions. Without any volunteered information, Taxpert normally must ask questions about each of the five requirements for dependency. In the example of Figure 2, note that Taxpert need not ask any questions about the relationship between the individuals since this information was volunteered in response to an earlier question.

In this paper, we focus on the interpretations of natural language responses to system questions. Ultimately, we plan to replace the menu system with a full natural language interface. In such a system, the user could indicate what goal he would like the system to solve by asking questions (e.g., "Can Fred claim John as a dependent?") as well as providing facts through natural language statements. While Director does currently support the setting of expert system goals through user input, the semantic translation of user queries to goals is still under consideration.

3 Semantic Interpreter

Our semantic mechanism relies on categorization of verbs. We have looked at over 90 verbs from the tax code and classified them into 12 categories. Analysis of categories of verbs have been done by researchers before [Osgood 79] [Ballmer et al. 81]; however, the analysis is generally done for one category only, such as verbs of motion [Miller 72]. In real world domains, like tax advising, many such categories are necessary.

Each verb category is organized hierarchically where each node of the hierarchies is derived from the meanings of one or more verbs. The leaves of the hierarchies contain either expert system facts or pointers to other hierarchies. Thus, the hierarchies form a connected forest A selectional restriction algorithm is encoded into each hierarchy. The restrictions on the arguments of the verbs rely on the noun features which are based on Roget's thesaurus. During parsing, the restrictions on the agent, patient, object and modifier of the verb help guide the parse down the hierarchy and derive the appropriate facts. Although the verb hierarchies are the primary source of facts, some facts are derived directly from the noun features.

The deep syntactic structure of the sentence, which is derived by an ATN parse, also influences semantic interpretations. In particular, it influences semantic disambiguation, the number of facts derived from a given sentence, instantiation of variables and relationships between facts. Interaction between syntax and semantics is not the main focus of this paper.

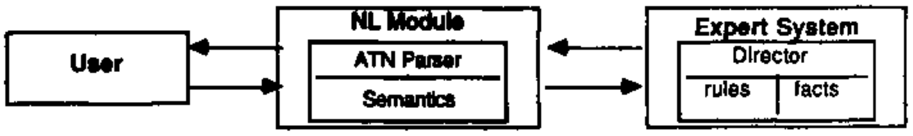
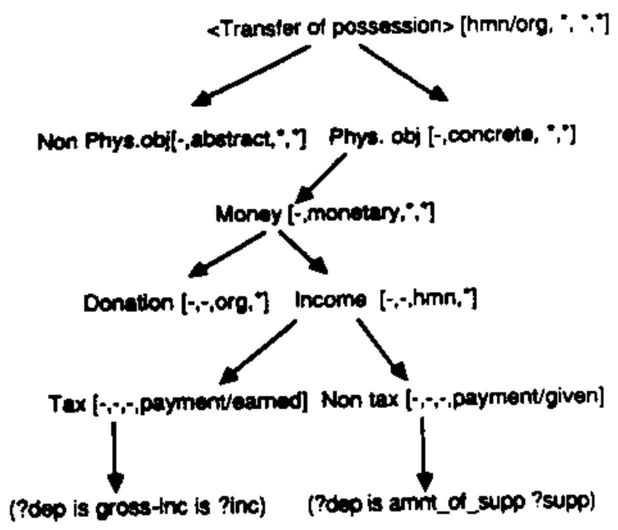


Figure 1: Natural Language - Expert System Interaction

***The Dependency agent contains over 80 rules and is implemented in Zetalisp on the the Symbolics Lisp Machine.

3.1 Parsing Example



<Transfer of possession> [hmn/org, *, *,*]
Non Phys.obj[-,abstract,*,*]
Phys. obj [-,concrete, *,*]
Money [-,monetary,*,*]
Donation [-,-,org,*]
Income [-,-,hmn,*]
Tax [-,-,-,payment/earned]
Non tax [-,-,-,payment/given]
(?dep is gross-inc is ?inc)
(?dep is amnt_of_supp ?supp)

Figure 3: Partial Tree formed for the *Transfer of possession* category****

As an example, consider one of the largest categories in our domain, *Transfer of possession.* It contains many verbs, such as *give, get, receive, provide etc..* Figure 3 shows a partial hierarchy formed for this category. A dictionary entry for a verb contains the category or categories to which the verb belongs; a plus or a minus, which indicates whether the subject of a sentence is the semantic agent or patient; and sometimes a lower level node from the parse tree. For example, the verb *to get* has a dictionary entry of *Transfer of possession <->* indicating that the underlying subject is generally the patient in a sentence with this verb. In the sentence *John gets $500,* John is the recipient or the patient The verbs *to pay* and *to earn* have more specific meanings and therefore have lower level tree nodes as entries in the dictionary nodes. The verb *to pay* is defined as *Transfer of possession <+>, monetary,* because the verb generally indicates the transfer of monetary amounts, and the verb to *earn* as *Transfer of possession* <+>, *Taxable,* because it generally indicates the existence of a taxable income.

More specific information can come from cither the verb, the arguments of the verb in the sentence, or both. The arguments in square brackets indicate the restrictions comming from the agent, patient, object and modifier of the verb, which yield more specific meanings. Consider a typical input sentence *John gets 500 dollars of support from Fred* The verb *gets* is defined in the dictionary as *Transfer of possession <->.* Thus, during the parse the *Transfer of possession* hierarchy is chosen based on the definition of the verb in the dictionary. Next, the parser has a choice of proceeding down to either *Physical Object* or *HonJ>hysical Object.* It selects *Physical Object* because *500 dollars,* which is the object of the sentence, fits the *concrete* restriction. At the next level, *concrete* is further restricted to *monetary.* Now, the choice is between *Donation* and *Income.* Here *Income* is selected based on the feature *human* of the. a^ent (John), because in our domain a monetary amount given to a *human* generally implies the money was earned, while a monetary amount given to an *organization* implies *donation.* At the next level, the choice is between *Taxable* and *Non Taxable.* Here the additional information comes from the modifier instead of the case roles as before and *Non Taxable* is selected because *support* has the *payment/given* feature in the dictionary. Finally, the fact (?depcndent is amountofsupport ?support) is added to the data base (or working memory) of the expert system.

••*In the figure, * stands for wild card, and - meam that the feature is inherited from the parent node.

Not all facts are derived from verb hierarchies. Some facts are implied directly by noun features. The tax code contains a large number of tests that deal with family relationships. Thus, features such as *relative, child, parent* etc.. are assigned to some of the nouns. For example, the feature *child* is assigned to words such as *son, daughter, step-son, step-daughter, foster-son, etc.,* and in turn has a more general feature of *relative.* The feature *child* directly implies the fact *(?dependent is child of ?user).*

3.2 Current Directions

Our semantic approach specifies how to derive expert system facts from user statements, but there are several other functions that it should have. It should be able to derive expert system goals from user queries and handle partial matches (i.e. deal with data that matches only a part of a fact), as well as deal with semantically incomplete input and anaphoric reference. The exact algorithm for instantiating the variables in the facts still has to be formalized. These issues are currently being investigated and the semantics is being fully implemented and integrated as part of Taxpert.

4 Director

The natural language interface described presents facts to the underlying expert system in a more or less unconstrained fashion. Director is an interpreter for rule-based expert systems that was specifically designed to be able to handle such input. The two major requirements for the interpreter are to efficiently utilize information volunteered by the user, while maintaining a focused and coherent interaction.

A rule-based system executes via the evaluation of its rules. These evaluations are controlled by the system's interpreter, which chooses which rules to evaluate according to some strategy. System queries to the user are generated as the rules attempt to determine the values of various data. Therefore, in these systems the goal of minimizing the number of questions and providing a focused interaction can be realized through suitable control of rule firings, i.e., through an appropriate interpreter. Many common interpreters for rule-based systems are based on sequential statement evaluation (e.g., [Bobrow et. al. 83]), forward chaining (e.g., [Forgy 81]), or backward chaining (e.g., [Van Melle 81]). Sequential control, often used as the basis for specialized user-programmed control structures, is of little direct assistance in building rule-based systems. Systems that are restricted to forward chaining inference violate the coherence requirement because they focus on deriving inferences from a set of facts, rather than investigating hypotheses. Systems restricted to backward chaining often do not allow a user to volunteer information, ignoring inferences from new information. More than a simple combination of forward and backward chaining is necessary, thus Director is based on a heuristically controlled combination of the two strategies, and so is able to efficiently utilize the facts entered by the natural language module.

4.1 Implementation

Since each rule is selected according to the selection procedure contained within the interpreter, this procedure influences the structure of the rules and the control information that must be explicitly encoded into the system. Indeed there is probably no major expert system in which the rules are independent of their interpreter [Duda 84]. In Director, each rule is invoked as a function, whose body is an if-then form in which the premise and the action are restricted Lisp s-expressions.

Each rule premise is restricted to data base (working memory) queries, i.e., the examination of the values of facts. The value of a fact may be added to the data base in only two ways: either through the action of a rule or through user input. Any fact that is not added by the action of a rule has an associated query procedure so that the user can supply its values. This query procedure is invoked if the premise of a rule tries to examine the fact's value, and the value is not present in the data base. Director automatically maintains the mappings between the rules and the query procedures for their associated facts.

The action of a rule is restricted to a single data base assignment The value to be asserted may be a constant, the value of a datum, or the result of a function evaluation. However, any input/output performed by such a function is beyond the control of Director. No query procedure is automatically associated with the fact mentioned in the action of a rule.

4.1.1 Interpreter

Director uses both forward and backward chaining. When a fact is given to the system, all possible inferences from the data in the current data base of facts are made using forward chaining. This means that full consideration is given to newly entered facts. Thus, forward chaining promotes a focus of attention according to the facts offered to the system by its user. When a user query is received, Director establishes a goal, a hypothesis, to confirm or reject. If the coal is not satisfied by simply examining the data base, backward chaining occurs. Backward chaining is guided by heuristics that try to maintain focus of attention according to both the user query and the facts recently mentioned (see Section 4.2.1). During backward chaining additional data may be entered, and forward chaining is performed to determine all inferences of this new information. This control structure allows Director to shift focus and goals in response to the user's change of focus and goals. See figure 4 for the algorithm used by Director.

4.2 Queries and Focus.

Director must select rules for evaluation in a way that tries to minimize the number of queries posed to the user. This is done through the use of heuristics, as well as by carefully recording information about user inputs. The heuristics determine which rule is most appropriate for evaluation based on the number of known facts in that rule as well as on focus considerations.

Given a fact or facts entered by the user;
1. Forward chain making all possible inferences without asking any questions.
2. If Goal is not found - Backward Chain.
3. Forward chain on all additional data.

Figure 4: Algorithm Used by the Inference Engine

4.2.1 Heuristics for Backward Chaining.

I. If (?dependent is parent_of ?user)) Then (Relationshiptest is met)

2. If (Relationshiptest is met) (?dependent gets multiple-support) (?User alone gives over 10%)) Then (Supportjest is met)

3. If (Relationship-test is met) (?User is the only supporter of ?depcndent) (?User gives over 50%))) Then (Support-test is met)

4. If (Relationship-test is met) (Supportjest is met))) Then (?dependent is claimable)

Figure 5: A set of rules from the Dependency Agent

Consider the set of rules in figure 5, which come from the Dependency agent of Taxpert. Suppose the user enters the following statements: *John gets $500 of support from Fred. Fred is the only one supporting John who, is his father. Can Fred claim John?.* These sentences add facts *(?dependent is parent of ?user)$_y$ (?User is the only supporter of ?dependent)* and *(?dependent is amount of support ?support)* to the data base and states that the goal is to know whether *(?dependent is claimable).* Director first forward chains to make all the possible inferences given the contents of the data base. In this case rule 1 is evaluated, adding *(Relationship-test is met)* to the data base. Now the system backward chains starting at rule 4. It then determines that in order to prove *(? dependent is claimable),* it must first prove *(Supportjest is met),* so the system backward chains again with the new goal. We want Director to pick the next rule in such a way as to guarantee the most focused conversation. To promote this behavior, Director tries to select the rule with both the goal in its right-hand-side and the greatest number of facts most recently added by the user in its left-hand-side. This implies that Director must differentiate those facts derived by rules and those entered by the user. Furthermore, Director must assign a time-stamp to each fact added by the user. In this example Director would try rule 3 first, because it contains *(?User is the only supporter of ?dependent)* which was entered by the user. Now the system has to ask only one question, to determine whether *(?User gives 50%)* is true before answering the user's question. However, if *(?User is only supporter of ?dependent)* were unknown, the system would choose arbitrarily between the rules 2 and 3. If rule 2 were chosen first, the user might have had to answer two additional queries, namely whether *(?User alone gives over 10%)* is true and whether *(?dependent gets multiple-support)* is true, before going on to consider rule 3 and answering the questions associated with the facts in that rule as well.

The heuristics are guided by the facts entered recently and thus do not always give optimal behavior. However, if the user mentions relevant facts as he issues queries (as would be expected if the user understood the problem domain), the behavior should be quite natural, giving a focused conversation, and minimizing the number of system questions.

4.2.2 User Control of Backward Chaining.

Sometimes the user has semantic knowledge of the queries and can, therefore, better direct the selection of rules. Forward chaining can be controlled simply by the facts that are added to the data base. Backward chaining can be controlled by the facts and the queries issued to Director. An optional mechanism is provided in Director to allow the user to help direct the rule selection process. Normally, expert systems use only information in the right-hand-sides of their rules to initiate backward chaining. Director can also use information in the left-hand-side of rules when selecting rules to use as a starting point of the backward chaining process. If the user supplies this left-hand-side information when making a request, it will be used in the initial rule selection. For example, consider the following set of rules:

1. If (?dependent is a child) (?dependent is a student)) Then (Gross_income_test is met)
2. If (?dependent is a child) (?dependent is ?age < 19)) Then (Grossincometest is met)

Suppose a user issues the following query: *Do students automatically meet the income test?*

The fact *(?dependent is a student)* and the goal *(Gross income test is met)* are derived from the question above and added to the data base. Using the maps, Director identifies that the general goal is implied by rules 1 and 2. The system now selects a rule as the starting point of the backward chaining process, choosing rule 1 according to user control. If this information was not available, or if the system did not take it into account, rule 2 may have been selected first and additional questions may have been generated.

4.23 Maps

During rule selection the interpreter must know which facts are contained in the left- and right-hand sides of the rules. This information can be obtained by searching the rule set Naive searches, however, could be expensive computationally and could make the response time of the system unreasonable. To make this searching efficient, Director maintains two maps. These maps are the rules-add-fact map (RF), and the facts-used-by-rule map (FR). The RF map provides pointers from each fact to the rules that can add it to the data base. The FR map provides pointers between each rule and the facts contained in its left-hand-side, thus specifying which facts have to be true in order for that rule to fire. The maps are built up during a preprocessing stage, which has to be performed only once for a given set of rules.

First, let us look at the RF map. Sunpose that fact C is in the right-hand-side of rule r:***(If (A ∧ B)*** . *n C).* The RF map entry for this fact would be ***(r<-->C),*** indicating that rule r adds fact C to the data base. The information in this map is used during the rule selection portion of the backward chaining phase. For example, if Director is trying to solve goal C, then the RF map provides efficient access to r. This map also allows Director to suppress the firing of certain rules: After a value is assigned to a fact, the system checks the RF map and tries to mark those rules that would assert the same value of this fact (Rules are not evaluated during the marking process, hence the only rules marked are those that reference this fact by a constant name and assert the same value as a constant.) The marked rules are not evaluated, thus avoiding rule evaluation and the superfluous queries to the user that these evaluations might cause.

Similarly, the facts-used-by-rule map would contain an entry for rule r, ***(A,B<-->r),*** indicating that rule r depends on facts A and B. The map would also contain entries for A and B showing that rule r requires their values in order to be evaluated. When some fact A is added to the data base, all those rules that use A in a forward chaining inference are readily found. In the present example, if A and B are in the data base, rule r is found in the FR map to be usable for forward chaining. This map is also used in the rule selection process of backward chaining. Having determined that rule r will be used to infer a needed fact C, the system readily determines that facts A and B need to be known.

4.2.4 Self Description.

So far, we have described what we call Director's inferencing function. The system has another function, called Display. There are many instances when a user wants the system to provide information without providing infcrencing. For example, a user may want to see everything the system knows about a certain fact A and issue the following request: "Tell me about A." Our system can handle a query of this sort by using the maps. All the rules that contain A in the left-hand-sides are found with the help of the FR map. Similarly, all the rules containing A in the right-hand-side are found with the help of the RF map. All rules containing A are returned as a response to the above query to the semantic module that translates user questions into requests to Director. Providing this information is done quickly because Director does not perform inferences, but rather only references the maps.

5 Conclusions

In this paper we described two important issues that must be addressed when constructing natural language interfaces to expert systems. First we described the semantic mechanism that is powerful enough to translate user statements into facts of the underlying expert system. This semantic approach is based on verb categorization. Each category is structured hierarchically, and the parsing algorithm is directly encoded into each hierarchy. Some issues in the construction of the complete semantic module are still being investigated. These are partial matching, i.e. what to do with inputs that only match part of a fact, instantiation of variables in the facts, as well as derivation of goals from user queries. The semantics presented is not only useful in the expert systems domain, but also in any domain where the underlying system is not well structured.

The natural language module adds facts to the data base of the underlying expert system in an unconstrained manner, thus placing extra requirements on the underlying expert system. We discussed Director, an inference engine that uses a combination of forward chaining and backward chaining so as to efficiently utilize facts entered by the natural language interface. It makes available descriptions of its rule base and allows for a limited form of user control over its backward chaining mechanism. This facility allows the user to ask questions about the information contained in the rules but not normally supplied by expert systems. These attributes of Director allow a knowledgeable user to arrive at a solution to his query in the most efficient and least time consuming way, while maintaining a focused dialogue. Director is also useful in domains where the decisions have to be made quickly, or where user queries are expensive, such as expert systems designed for use by busy professionals, such as accountants and doctors, as well as systems that work in hazardous environments, such as nuclear reactors.

The two approaches presented here mark an important step toward the construction of a full natural language interface for rule-based expert systems.

References

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

[Bobrow et. al. 83]
Bobrow, D.G., Stefik, M. *The LOOPS Manual* Xerox Corp., Palo Alto, Ca., 1983.

[Clancey 83] Clancey, W. The Epistimology of a Rule-Based Expert System - a Framework for Explanation. *Artificial Intelligence 20* , 1983.

[Clancey 79] Clancey W. Dialogue Management for Rule-Based Tutorials. In *Proceedings ofIjCAI.* Japan, 1979.

[Datskovsky 84] Datskovsky, G. *Menu Interfaces to Expert Systems: Evaluation and Overview.* Technical Report, Columbia University, New York, 1984.

[Davis 78] Davis, R. Knowledge Acquisition in Rule-Based Systmes-Knowledge About Representation as a Basis for System Construction and Maintanance. *Pattern Directed Inference Systems.* Academic Press, 1978.

[Duda 84] Duda. Presentation at the IEEE Workshop on Principles of Knowledge-Based Systems.
1984

[Duda et. al. 79] Duda, R., Gasching, J., Hart, P. Model Design in the Prospector Consultant System for Mineral Exploration. In Michie, D. (editor), *Expert Systems in the micro-electronic age.* Edinburgh University Press, 1979.

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

[Forgy 81] Forgy, C. *OPS5 User's Manual* Carnegie-Mellon University, Pittsburgh, Pa., 1981.

[Grosz et. al. 85] Grosz, B.,Martin, P., Appelt, D., Pereira, F. *Team: An Experiment in the Design of Transportable Natural Language Interfaces.* Technical Report, SRI International, 1985.

[Hayes Roth 85] Hayes Roth, F. Rule Based Systems. *Communications of the ACM, Vol. 28, No.9* , 1985.

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

[Kaplan 79] Kaplan, S J. *Cooperative Responses From a Portable Natural Language Data Base Query System.* PhD thesis, University of Pennsylvania, 1979.

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

[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.

[Pollack 83] Pollack, M.E. *Generating Expert Answers Through Goal Inference.* Technical Report, SRI International, 1983.

[Pollack et. al. 82] Pollack, M., Hirschberg J. and Webber, B. User Participation in the Reasoning Process of Expert Systems. *AAAI*, 1982.

[Shortliffe 76] Shortliffe, E.H. *Mycin: A rule-based computer program for advising physicians regarding anitimicrobial therapy selection.* PhD thesis, Stanford University, 1976.

[Van Melle 81] Van Melle, W. *The Emycin Manual.* Technical Report Stan-cs-81-885, Stanford University, Stanford, Ca., 1981.

[Webber 71] Nash-Webber, B., Verbs of Composition. 1971.Harvard University, 1971.

[Woods 73] Woods, W.A. An Experimental parsing System for Transition Network Grammars. In Rustin (editor), *Natural Language Processing.* Algorithmic Press, 1973.

[Woods et. al. 72] Woods W., Kaplan R., Nash-Webber B. *The Lunar Sciences Natural Lnaguage Information System: Final Report.* Technical Report 2378, BBN, Cambridge, Mass, 1972.