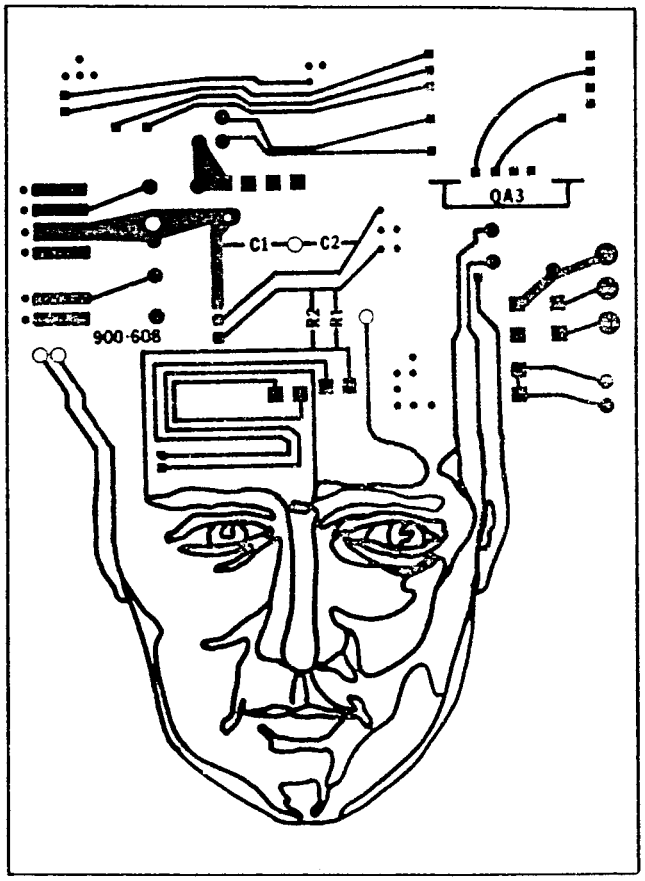
1985 Conference On Intelligent Systems and Machines



April 23-24, 1985

Center for Robotics and
Advanced Automation
Oakland University
Rochester, Michigan

PARSING AS A FORM OF INFERENCE IN A MULTIPROCESSING ENVIRONMENT

TA 1

Jeannette G. Neal & Stuart C. Shapiro

Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260

Abstract

A natural language processing system in which parsing is accomplished by a general purpose inference mechanism is described. Inference is performed by a multiprocessing system using a bi-directional inference strategy which produces left-corner parsing. Although strings are analyzed left-to-right, the system can handle both left-recursive and right-recursive grammars. An example is presented to demonstrate and illustrate these features.

Parsing System

This paper discusses a parsing system in which parsing is a form of inference in a multiprocessing environment. This system has been developed as part of our research into the development of an expert AI system which can treat language as its discourse domain and can use a language as its own metalanguage [9,13]. It has been implemented using the SNePS semantic network processing system [11,14].

Our approach is to view language processing as a form of reasoning or thought of a cognitive agent. Our System's language processing knowledge is stored in declarative form as rules and assertions in its general knowledge base so as to be available to be queried, reasoned about, and used, just as any other domain knowledge is available in an artificial intelligence or expert system.

The two important SNePS subsystems which primarily determine the behavior of our parser are: (a) MULTI [7], which provides a simulated multiprocessing form of control, and (b) SNIP [12], the SNePS Inference Package, which uses MULTI for its control structure.

MULTI maintains a queue of processes and continually executes processes from the queue until the queue becomes empty. The MULTI system provides primitives for creating processes, scheduling processes, and manipulating process registers. During inference, for each rule that is triggered, a process is created representing the active form of the rule with responsibilities that include keeping track of variable bindings and the number of antecedents that have been satisfied for the given rule.

SNIP, the SNePS Inference Package, employs bi-directional inference, a form of inference resulting from interaction between forward and backward inference which loosely correponds to bi-directional search through a space of inference rules. This technique focuses attention towards the active processes and prunes the search through the space of inference rules by ignoring rules which have not been activated. This cuts down the fan out of pure forward or backward chaining. New rules are activated only if no active rules are applicable.

Our parsing System is not a robust parser for a predetermined language such as English. Instead, our System understands a predefined Kernel Language (KL) with which a "teacher-user" can define or bootstrap to a target language of his/her choice. The syntactic definition of a language is expressed to our System as a set of rules and assertions, initially in the System's Kernel Language. This KL includes a set of predefined terms, and three simple forms of linguistic rewrite rules. The rules and assertions defining the syntax of a target language are stored in the System's semantic network knowledge base.

We have developed semantic network knowledge representations for initial and bounded surface strings and the predefined relations: (1) a lexeme being a member of a certain lexical category, (2) a bounded string being a member of a certain category, (3) a certain phrase structure being a constituent of another structure, (4) a syntactic structure expressing a certain concept, (5) the subset relation, (6) the rule connectives of SNePS [11].

Parsing Strategy

All processing in our System, particularly the parsing process, is a form of logic-based inference. Our System uses a combined bottom-up top-down parsing strategy which is primarily due to the use of bi-directional inference by the SNePS Inference Package under the control of MULTI. Furthermore, all analyses of a string are derived in parallel as a result of the simulated multiprocessing form of control provided by the MULTI subsystem of SNePS. Our parsing System is similar to the Prolog-based language processing systems of Pereira & Warren [10], Warren & Pereira [15], Dahl [3,4], McCord [6], Dahl & McCord [5] in that parsing is a form of inference. A Prolog-based system, however, uses a top-down backtracking strategy to derive a goal analysis of a surface string while our System uses a combined bottom-up top-down strategy with no backtracking.

Parsing begins in a bottom-up manner as our System reads one token at a time in a left-to-right direction from the input utterance to be analyzed. When the System reads a token of the input string, all applicable rules are started in parallel in the form of processes created by the MULTI subsystem. For each rule that is triggered, a process is created to keep track of variable bindings and the number of antecedents that have been satisfied. A process is suspended if not all its antecedents are satisfied and is resumed if more antecedents are satisfied as the reading of the string continues. As parsing proceeds, the parse tree(s) for an input utterance is (are) represented in the System's network knowledge base. Our System builds and retains network structures corresponding to alternative analyses of a given input string. Retention of the alternatives avoids the reanalysis of previously processed surface strings that occurs in a backtracking system. The strategy of our System is similar to "left-corner bottom-up parsing" [1].

Left-Recursion and Right-Recursion

The definition of the syntax of certain types of strings naturally requires left-recursive rules. For example, rules to define the syntax of a noun phrase that includes the use of a noun phrase possessive as in "big John's father" would naturally use left-recursion. Left-recursion is a problem for some top-down left-to-right parsers such as the Augmented Transition Network [16,17,18]. Left-recursive definitions are also a problem for Prolog-based parsers since Prolog attempts to satisfy the goals in a conjunction (whether in a rule body or in a question) in a left to right order and it searches its data base of rules and facts in a top-down depth-first manner [2].

Our System accepts and applies syntactic rules incorporating either left-recursion, right-recursion, or both. McKay and Shapiro [8] discuss the fundamental behavior of the SNePS Inference Package with regard to recursive rules and the active connection graph representation of the rule processes.

The following small grammar incorporates both left- and right-recursion. It specifies that an NP1 consists of a noun, proper-noun or a sequence of one or more adjectives followed by a noun or proper-noun. An NP1 cannot contain a possessive noun-phrase modifier, however. An NP2, on the other hand, consists of a possessive noun-phrase modifying an NP1. The grammar specifies that the possessive relation is left-associative in a string such as "JOHN 'S SISTER 'S HOUSE". POSS is our identifier for a possessive noun-phrase and we use POS-SYM as a non-terminal identifying the class of possessive symbols (e.g. "'S").

(1) NP ---> NP1
(2) NP ---> NP2
(3) NP1 ---> ADJ NP1
(4) NP1 ---> NOUN
(5) NP1 ---> PROPER-NOUN
(6) POSS ---> NP1 POS-SYM
(7) POSS ---> NP2 POS-SYM
(8) NP2 ---> POSS NP1

Rule (3) is right-recursive and rules (7) and (8) combined involve left-recursion. Our System accepts the above definition and can apply it during the parsing process, as we show in the next section. The order in which these rules are input to the System is actually immaterial, since, during parsing, applicable rules are used in parallel.

Discussion in the Context of an Example

Let us examine the System's application of the above set of rules, which were shown exactly as they would be input to the System. This form of context-free rewrite rule is part of the Kernel Language of the System. As indicated previously, all knowledge, including both rules and assertions, are represented in declarative form in the semantic network knowledge base. The above rules are represented in the form of conditional rules in SNePS [11]. The network representations of rules (1) through (8) above can be paraphrased as

follows. (NOTE: In all the paraphrased rules of this paper, V_1 and V_2 are universally quantified variables and the numbers in parentheses are rule numbers, not line numbers. Thus, for example, nested rule (3b) begins with "if V_2 follows V_1" and continues to the period at the end of the sentence.)

(1a) If V_1 is an NP1 then V_1 is an NP.
(2a) If V_1 is an NP2 then V_1 is an NP.
(3a) If V_1 is an ADJ then
(3b) if V_2 follows V_1 and V_2 is an NP1
then the string consisting of V_1 followed by V_2 is an NP1.
(4a) If V_1 is a NOUN then V_1 is an NP1.
(5a) If V_1 is a PROPER-NOUN then V_1 is an NP1.
(6a) If V_1 is an NP1 then
(6b) if V_2 follows V_1 and V_2 is a POS-SYM
then the string consisting of V_1 followed by V_2 is a POSS.
(7a) If V_1 is an NP2 then
(7b) if V_2 follows V_1 and V_2 is a POS-SYM
then the string consisting of V_1 followed by V_2 is a POSS.
(8a) If V_1 is a POSS then
(8b) if V_2 follows V_1 and V_2 is an NP1
then the string consisting of V_1 followed by V_2 is an NP2

To examine the System's use of the above rules, we assume that the System has also been instructed that "BIG", "RICH", "OLD", and "BEAUTIFUL" are each in the category ADJ, "SISTER" and "HOUSE" are each in the category NOUN, "JOHN" is a PROPER-NOUN, and "'S" is a POS-SYM (possessive symbol). Our System does not do morphological analysis so we assume a pre-processor that recognizes possessives and detaches the suffix. Now the string "BIG JOHN 'S RICH OLD SISTER 'S BEAUTIFUL HOUSE" is input to the System.

When the System begins to read an input string, parsing starts in a bottom-up manner. The System reads the first word "BIG" and this triggers rule (3a). Since the antecedent of rule (3a) is satisfied, the System checks the antecedent of rule (3b), questioning whether a string immediately following the word "BIG" is an NP1. When a SNePS rule is triggered, a process is created forming the active version of the rule to collect instantiations of the antecedents of the rule so that appropriate instantiations of the consequents of the rule can be deduced. If not all the antecedents are satisfied, the process is suspended and acts as a demon, waiting for instances of the unsatisfied antecedents. This is the case for the nested rule (3b). Since no string follows the word "BIG" yet, the process (P1) for rule (3b) is suspended.

Rule processes, with their communication links, form the equivalent of an hypothesized parse tree with associated expectations. The hypothesized parse tree corresponding to the process (P1) of rule (3b) is illustrated in Figure 1. In the following figures, the question-marks indicate expectations which have not yet been satisfied.

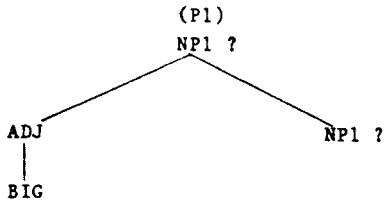


Figure 1. Hypothesized Parse Tree Corresponding to Active Version of Rule (3B)

When the second word of the input string is read, rule (5a) is triggered and the System concludes that the string "JOHN" is an NP1. In general, if no relevant processes are awaiting new information acquired by the System, all applicable rules are started in parallel. Thus if no process existed for rule (3b), awaiting an NP1, rules (1a), (3b), (6a), and (8b) would be activated in parallel. However, the inference system ignores unactivated rules as long as there are applicable rule processes awaiting data, essentially parsing in a top-down manner in this situation. The suspended process (P1) for rule (3b) is waiting for an NP1 following the string "BIG" and therefore no new rules are triggered. Since the antecedent of rule (3b) is satisfied, the System concludes that the string "BIG JOHN" is an NP1. No rule processes are awaiting this new conclusion, so rules (1a), (3b), (6a) and (8b) are triggered.

Since the antecedent of rule (1a) is satisfied, the System concludes that the string "BIG JOHN" is an NP.

The antecedent of rule (3b) is satisfied, but rule (3b) is not known by the System to be true. Therefore, a process (P2) is established to form the active version of rule (3a). It immediately questions whether the antecedent of rule (3a) is satisfied and, finding this not to be the case, is suspended to await an appropriate instance of this antecedent. Such an instance will never occur, of course, since no ADJ will ever precede the NP1 "BIG JOHN". Therefore process P2 will remain forever suspended.

Since the antecedent of rule (6a) is satisfied, another process (P4) is created as the active form of rule (6b). This process immediately seeks a POS-SYM following the NP1 "BIG JOHN" and is suspended since no such POS-SYM currently exists.

Since the antecedent of rule (8b) is satisfied, a process (P5) is created as the active form of rule (8a) to establish whether a POSS precedes the string "BIG JOHN". Just as process P2 will remain suspended, P5 will also.

The combined parse tree and hypothesized parse trees thus far established are shown in Figure 2.

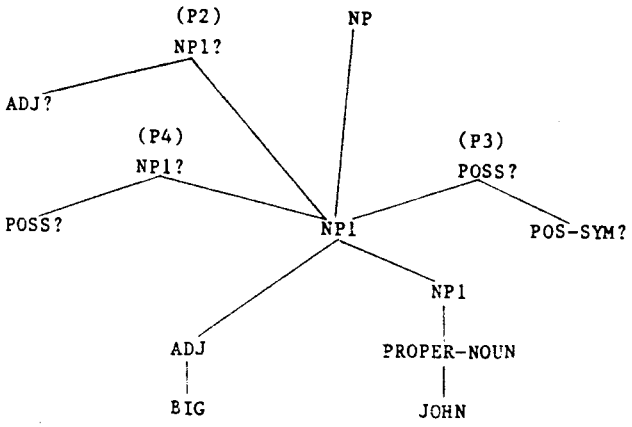


Figure 2. Combined Parse Tree and Hypothesized Parse Trees

When the third token "'S" is read from the input string, since the System knows that it is a POS-SYM, the existing process (P3) for rule (6b) consumes this information and concludes that the string "BIG JOHN 'S" is a POSS. This data triggers rule (8a) and, as a result, a process (P6) is created for rule (8b). P6 checks for an NP1 following this POSS string and, finding none, is suspended to await such an NP1.

The next word "RICH" of the input string is known to be an ADJ which triggers rule (3a). This causes a process (P8) to be created for rule (3b) to await an NP1 following the ADJ "RICH". Similarly, when the next word "OLD" is read by the System, since it is known to be an ADJ, it triggers rule (3a) and another process (P10) is created as an active form of rule (3b). Process P10 awaits an NP1 following the ADJ "OLD" whereas process P8 for rule (3b) awaits an NP1 following the ADJ "RICH". The current parse trees and suspended processes are illustrated in Figure 3.

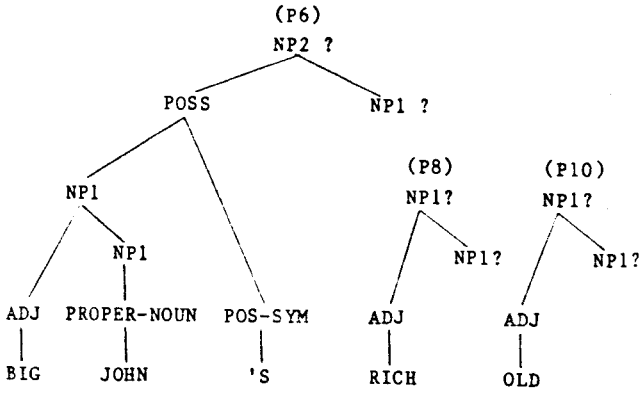


Figure 3. Combined Parse Tree and Hypothesized Parse Trees

The next token read by the System from the input stream is the NOUN "SISTER". This NOUN satisfies the antecedent of rule (4a) so the System concludes that the string "SISTER" is an NP1. This conclusion triggers no inactive rules since a process (P10 of Figure 3) already exists to await this information. Process P10 is an active form of rule (3b). Since the antecedent of rule (3b) is now satisfied, the System concludes that the string "OLD SISTER" is an NP1. Process P8 was waiting for an NP1 following the word "RICH". Therefore the conclusion that "OLD SISTER" is an NP1 does not cause the creation of any new rule processes. Instead, the System concludes, via process P8, that since the antecedent of rule (3b) is satisfied by the string "OLD SISTER", the string "RICH OLD SISTER" is an NP1.

This new data is consumed by the demon process P6 (shown in Figure 3) awaiting an NP1 following the string "BIG JOHN 'S". Therefore the string "BIG JOHN 'S RICH OLD SISTER" is recognized as an NP2. No demon processes await this information, so rules (2a) and (7a) are triggered in parallel. By application of rule (2a), the System concludes that the string "BIG JOHN 'S RICH OLD SISTER" is an NP. Also, since the antecedent of rule (7a) is satisfied, a process (P12) is created corresponding to rule (7b) to question whether a POS-SYM follows the NP2 "BIG JOHN 'S RICH OLD SISTER". Finding none, it is suspended. The current parse tree is illustrated in Figure 4.

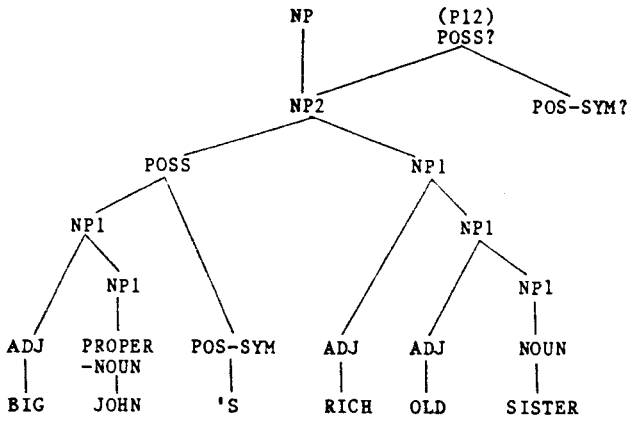


Figure 4. Combined Parse Tree and Hypothesized Parse Trees

When the next token "'S" is read from the input stream, this POS-SYM is expected by process P12 and the System effectively parses in a top-down manner again. Therefore the System concludes that the string "BIG JOHN 'S RICH OLD SISTER 'S" is a POSS string. Rule (8a) is again triggered and a process (P14) is created as the active version of rule (8b) which is suspended to await an NP1 following the newly recognized POSS string. The next word "BEAUTIFUL" is read by the System and triggers rule (3a). Since the antecedent of rule (3a) is satisfied, the System creates a process (P16) for rule (3b) to await an NP1 following the ADJ "BEAUTIFUL". The current parse trees are illustrated in Figure 5.

When the final token "HOUSE" is read by the System, rule (4) is triggered and the System concludes that the string "HOUSE" is an NP1. This

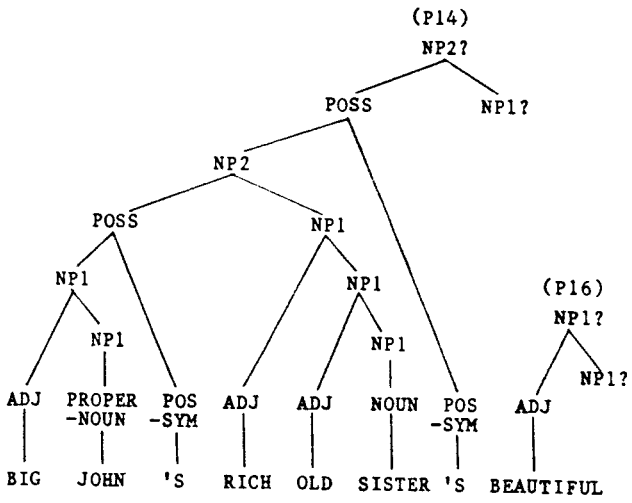


Figure 5. Combined Parse Tree and Hypothesized Parse Trees

information is expected by process P16 and the System, parsing in a top-down manner, reaches the conclusion that the string "BEAUTIFUL HOUSE" is an NP1. This information is expected by process P14 and, continuing in a top-down manner, the System concludes that the entire input string is an NP2. Since there are no demon processes awaiting this new information, it triggers rules (2a) and (7a). A process is created for rule (7b), but it will remain suspended since the last token of the input string has been read. By application of rule (2a), however, the System concludes that the input string is an NP. The final parse tree is illustrated in Figure 6.

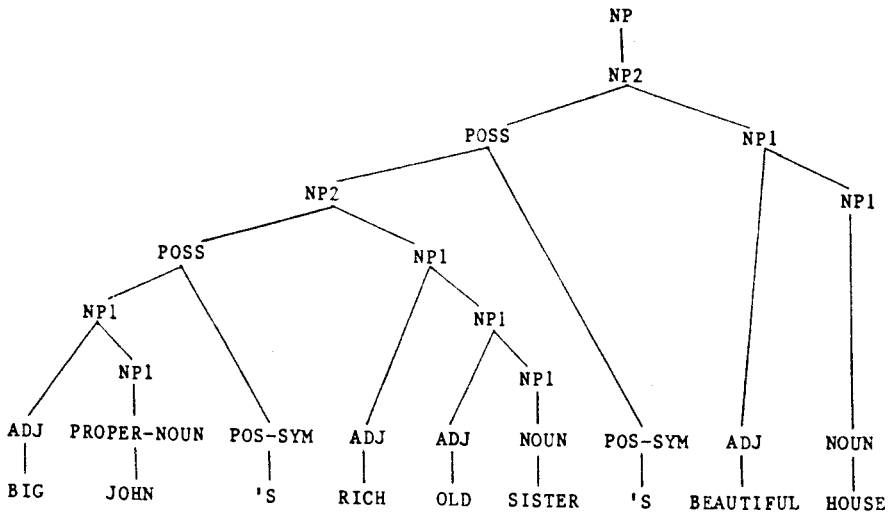


Figure 6. Final Parse Tree for the Example Input String

The completed successive applications of rule (3) to conclude that the strings "OLD SISTER" and "RICH OLD SISTER" were NPls involved right-recursion only. This is in contrast to the application of rule (7) in combination with rule (8) to recognize the nested NP2 strings in the string "BIG JOHN 'S RICH OLD SISTER 'S BEAUTIFUL HOUSE". Recognition of the nested NP2 strings required left-recursion.

Summary

We have implemented a rule-based parser using a general purpose inference machine such that parsing is a form of inference in a multiprocessing environment. We have used the SNePS semantic network processing system with its MULTI subsystem providing a simulated multiprocessing form of control for the bi-directional inference performed by the SNIP subsystem.

We represent syntactic knowledge in the System's semantic network knowledge base. This knowledge base is the repository of all the System's knowledge and is searched during inference processing for applicable rules and assertions.

The parsing strategy of our System combines bottom-up and top-down strategy in a manner similar to left-corner bottom-up parsing. All analyses of a string are derived in parallel as a result of the simulated multiprocessing form of control.

The definition of the syntax of certain strings naturally requires left-recursive rules. An example of such a string is a noun-phrase which includes a noun-phrase possessive as a modifier. Left-recursion is a problem for some popular top-down left-to-right parsers such as most implementations of the Augmented Transition Network. Our System accepts and applies syntactic rules incorporating left-recursion and/or right-recursion as demonstrated in the example of this paper.

References

1. Burge, W.H. (1975), Recursive Programming Techniques, Addison-Wesley Publishing Co.
2. Clocksin, W.F. & Mellish, C.S. (1981), Programming in Prolog, Springer-Verlag, New York.
3. Dahl, V. (1979), "Quantification in a Three-Valued Logic for Natural Language Question-Answering Systems", Proc. IJCAI-79, pp. 182-187.
4. Dahl, V. (1981), "Translating Spanish into Logic through Logic", AJCL, Vol. 7, No. 3, pp. 149-164.
5. Dahl, V. & McCord, M.C. (1983), "Treating Coordination in Logic Grammars", AJCL, Vol. 9, No. 2, pp. 69-91.
6. McCord, M.C. (1982), "Using Slots and Modifiers in Logic Grammars for Natural Language", Artificial Intelligence, Vol. 18, No. 3, pp. 327-367.
7. McKay, D.P. & Shapiro, S.C. (1980), "MULTI - A LISP Based Multiprocessing System", Conference Record of the 1980 LISP Conference, Stanford Univ., pp. 29-37.
8. McKay, D.P. & Shapiro, S.C. (1981), "Using Active Connection Graphs for Reasoning with Recursive Rules", Proc. of IJCAI-81, pp.368-374.
9. Neal, J.G. & Shapiro, S.C. (1984), "Knowledge Based Parsing", Technical Report No. 213, Dept. of Comp. Science, SUNY at Buffalo.
10. Pereira, F.C.N. & Warren, D.H.D. (1980), "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, pp. 231-278.
11. Shapiro, S.C. (1979), "The SNePS Semantic Network Processing System". In N. Findler, ed. Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, New York, pp. 179a-203.
12. Shapiro, S.C., Martins, J., & McKay, D. (1982), "Bi-Directional Inference", Proc. of the Cognitive Science Society, pp. 90-93.
13. Shapiro, S.C. & Neal, J.G. (1982), "A Knowledge Engineering Approach to Natural Language Understanding", Proc. of the 20th Annual Meeting of the Association for Computational Linguistics, pp. 136-144.
14. Shapiro, S.C. & the SNePS Implementation Group (1981), SNePS User's Manual, Dept. of C.S., SUNY at Buffalo, NY.
15. Warren, D.H.D. & Pereira, F.C.N. (1982), "An Efficient Easily Adaptable System for Interpreting Natural Language Queries", AJCL, Vol. 8, No. 3-4, pp. 110-119.
16. Winograd, T. (1983), Language as a Cognitive Process, Vol.1, Addison-Wesley Publishing Co.
17. Woods, W.A., Kaplan, R. & Nash-Webber, B.(1972), "The Lunar Sciences Natural Language Information System: Final Report", BBN Rep. No. 2378, Bolt Beranek & Newman, Inc.
18. Woods, W.A. (1977), "Lunar Rocks in Natural English: Explorations in Natural Language Question Answering", in Linguistic Structures Processing, A. Zampolli (ed.), North-Holland Pub. Co.,pp.521-570.