WORKSHOP ON AUTOMATIC DEDUCTION

MIT

Cambridge, Mass.

Aug. 17 - 19, 1977

COLLECTED ABSTRACTS

Compiling Deduction Rules From a Semantic Network Into a Set of Processes

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Amherst, New York 14226

Detailed Summary

For some time, we have been investigating the representation of deduction rules in semantic networks [1;2;9-13]. Recently, we have been implementing an inferencing system which, given the pattern for a piece of network to be deduced, locates relevant deduction rules [12], and "compiles" them into a set of processes which are then given to a multi-processing system for execution. The multi-processing approach was motivated partly by Kaplan's producer-consumer model of parsing [7] and partly by Wand's frame model of computation [14], which itself was based on the "little man" metaphor of Papert [8] and Hewitt's ACTOR model [3;4;5].

Deduction rules are represented in semantic network form for several reasons: they can be entered in the same way as other information, either in the same formal input language or in (some subset of) a natural language using the same parser and grammar; they can be treated as data - entered, retrieved, discussed, etc.; relevant deduction rules can be retrieved using the same network matching routines and in the same operation as retrieving explicit information; in semantic networks it is natural to represent a rule as a connective and an unordered set of arguments, delaying the decision of which argument(s) is(are) the antecedent(s) and which are(is) the consequent until the rule is to be used in a deduction. To illustrate the last point, consider the rule stating that the following propositions are equivalent:

1. Block x supports block y.
2. Block x is under block y.
3. Block y is above block x.

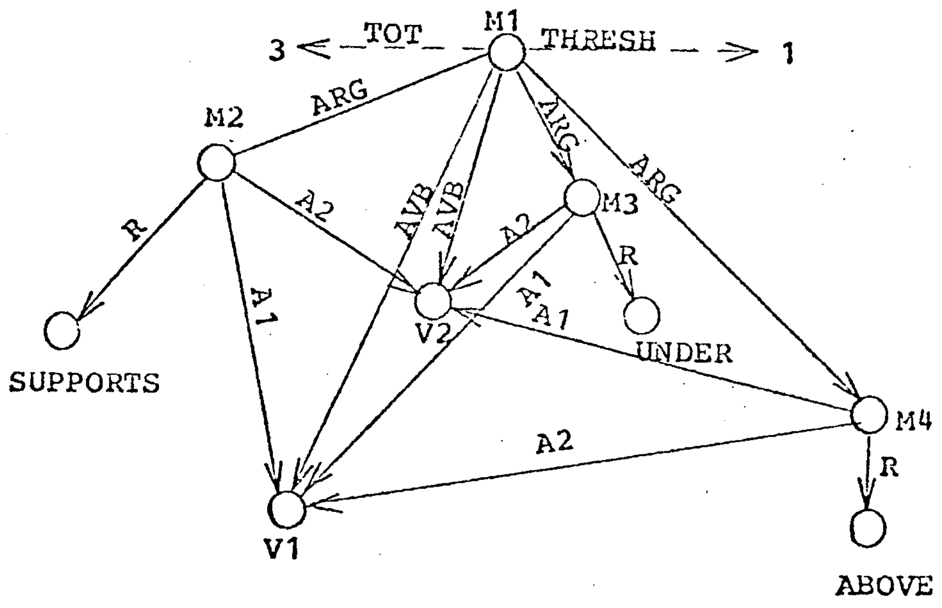
We may write this rule symbolically as:

$$\forall x,y \; {}_3\theta_1 \quad (\mathrm{Supports}(x,y),\ \mathrm{Under}(x,y),\ \mathrm{Above}(y,x))$$

in the SNEPS input language [11], this is written as:

```
(BUILD AVB($X $Y) TOT 3 THRESH 1
     ARG((BUILD A1 *X R SUPPORTS A2 *Y)
          (BUILD A1 *X R UNDER A2 *Y)
          (BUILD A1 *Y R ABOVE A2 *X)))
```

and as a semantic network, we draw it as:



If we want to deduce a node that matches M2,M3, or M4, we can use M1 as a consequent theorem [6], and the other two nodes become antecedents. If a node matching M2,M3, or M4 is asserted, we can use M1 as an antecedent theorem and the other two nodes become consequents.

When an argument of a deduction rule is matched, processes are created to carry out the indicated inference. Some processes analyze the deduction rule and create other processes that are specialized for using the deduction rule in the proper direction and with the proper sets of antecedents and consequents. For example, if node M2 were matched during a backward chaining operation, processes would be created for using rule M1 as a consequent theorem with M3 and M4 as antecedents, either of which is sufficient for deducing M2. Once created, these processes can be saved so that if the same rule is needed again to deduce the same consequent, the processes need not be recreated. Processes may be assigned priorities and resource bounds. They may be executed in parallel (or simulated parallel) subject to differences in priorities. When a process expends its resources, it is suspended until it is assigned additional resources.

Every process has a name which defines the action the process will perform and a continuation link to the process that is to be scheduled for activation after it has completed its job. Each process also has other "slots" or "registers" peculiar to the action it will perform. Processes pass information back along their continuation links by scheduling instances of the "messenger" process, ANS, which inserts its message in the MSG register of the receiving process and then schedules that process.

Two kinds of processes control the use of deduction rules used in a backward-chaining (consequent) manner. The process USE controls the top level of a deduction rule, while the process USE-1 controls embedded rules. Both processes have registers for the rule (RULE), the consequent (CQ) and a binding of the variables used in the rule (BNDG). For example, suppose the rule

$\forall x,y(x \text{ ON } y \rightarrow \forall z(y \text{ ON } z \rightarrow x \text{ ON } z))$

were to be used to deduce answers to the question, (A ON ?). A USE process would be created whose registers would be*

RULE: $\forall x,y(x \text{ ON } y \rightarrow \forall z(y \text{ ON } z \rightarrow x \text{ ON } z))$

CQ: $\forall z(y \text{ ON } z \rightarrow x \text{ ON } z)$

BNDG: ((x.A)(z.?))

Above it on a path of continuation links would be the USE-1 process with registers

RULE: $\forall z(y \text{ ON } z \rightarrow x \text{ ON } z)$

CQ: x ON z

BNDG: ((x.A)(z.?))

when the USE process receives a message informing it that (A ON B) is valid, it would create and schedule a specialized USE-1 process with registers

RULE: $\forall z(y \text{ ON } z \rightarrow x \text{ ON } z)$

CQ: x ON z

BNDG: ((x.A)(y.B)(z.?))

This process would attempt to answer the question (B ON ?).

Both USE and USE-1 processes have continuation links to processes with the name ANS-CATCH. This process has three registers: MSG, DATA, and BOSSES. The contents of BOSSES is a list of processes. Whenever ANS-CATCH is activated, it takes its messages (from MSG), and whichever ones are not already in DATA are added to DATA and sent to all the BOSSES. When a process wants to use a deduction rule to deduce some consequent,

---

*Actually the RULE and CQ registers would contain nodes, not symbolic expressions.

it first checks if a USE or USE-1 process already exists to use that rule for that consequent with a binding compatible with its own. If one is found, the process adds itself to the list of BOSSES in the ANS-CATCH above the USE or USE-1 and immediately takes all the answers in the DATA register of the ANS-CATCH. In this way, if a deduction rule is useful in several places in a deduction, duplicate work is avoided. In the case of recursive rules, like those for transitive relations, the result is a cycle of continuation links - an ANS-CATCH among whose BOSSES is a process with a path of continuation links to the ANS-CATCH itself. Answers will circulate in this cycle of processes, moving one link in the chain of transitive relations with each cycle, until no more answers can be produced. The ANS-CATCH process can be viewed as a specialized data base connected to rules which can be used antecedently whenever an assertion is added to its DATA register. Although these rules are pattern-directed, they are guaranteed to match any assertion that gets added to the ANS-CATCH.

References

1. Bechtel, R.J. Logic for semantic networks, M.S. Thesis. Technical Report No. 53, Computer Science Department, Indiana University, Bloomington, IN., July, 1976.
2. Bechtel, R.J.,and Shapiro, S.C. A logic for semantic networks. Technical Report No. 47, Computer Science Department, Indiana University, Bloomington, IN., March, 1976
3. Greif, I., and Hewitt, C. Actor semantics of PLANNER-73, Proc. 2nd ACM Symp. on Principles of Programming Languages, Palo Alto, 1975.
4. Hewitt, C.; Bishop, P.; and Steiger, R. A universal modular ACTOR formalism for artificial intelligence. Proc. IJCAI 3, Stanford, CA., Aug., 1973, 235-245.
5. Hewitt, C., et al. Actor induction and meta-evaluation. Proc. 1st ACM Symp. on Principles of Programming Languages, Boston, 1973, 153-168.
6. Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. AI-TR-258. M.I.T., A.I. Lab., 1972.
7. Kaplan, R.M. A multi-processing approach to natural language, Proc. NCC, 1973, 435-440.
8. Papert, S.A. Teaching children to be mathematicians versus teaching about mathematics. Int. J. Math. Educ. Sci. Technol. 3 (1972), 249-262.
9. Shapiro, S.C. The MIND System: a data structure for semantic information processing. R-837-PR, The Rand Corporation, Santa Monica, California, August, 1971.

10. Shapiro, S.C. A net structure for semantic information storage, deduction and retrieval. Proc. Second Int. Joint Conference on Artificial Ingelligence, The British Computer Society, London, England, September, 1971, 512-523.

11. Shapiro, S.C. An introduction to SNePS. Technical Report No. 31, Computer Science Department, Indiana University, Bloomington, IN., Revised December, 1976.

12. Shapiro, S.C. Representing and locating deduction rules in a semantic network. To be presented at the Workshop on Pattern-Directed Inference Systems, U. of Hawaii, May 23-27, 1977.

13. Shapiro, S.C. and Bechtel, R.J. Non-standard connectives and quantifiers for question-answering systems, in progress.

14. Wand, M. The frame model of computation. Technical Report No. 20, Computer Science Department, Indiana University, Bloomington, IN., December, 1974.