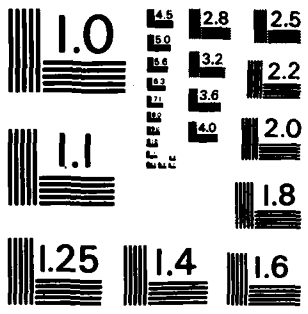
AD A137 062   LOGIC PROGRAMMING AND KNOWLEDGE BASE MAINTENANCE(U)   1/1
SYRACUSE UNIV NY SCHOOL OF COMPUTER AND INFORMATION
SCIENCE K A BOWEN NOV 83 AFOSR TR 84 0005
UNCLASSIFIED   AFOSR 82 0292   F/G 6/4   NL

END
DATE
FILMED
2 84
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0

(14)

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 84-0005	2. GOVT ACCESSION NO. AD-A137062	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) *LOGIC PROGRAMMING AND KNOWLEDGE BASE MAINTENANCE*		5. TYPE OF REPORT & PERIOD COVERED Annual, 1 SEP 82-31 AUG 83
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Kenneth A. Bowen		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0292
9. PERFORMING ORGANIZATION NAME AND ADDRESS Logic Programming Research Group, School of Computer & Information Science, 313 Link Hall, Syracuse University, Syracuse NY 13210		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A7
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research/NM Bolling AFB DC 20332		12. REPORT DATE NOV 83
		13. NUMBER OF PAGES 5
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

DTIC ELECTE JAN 20 1984 D A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The work conducted this year followed the projections set forth in the grant proposal rather closely. On the theoretical side, the investigators have continued to explore questions of the logical status of some of the standard data structures involved in various artificial intelligence applications involving knowledge bases. The greatest attention has been focused on frames. Exploration of the axiomatization and representation of semantic nets by similar methods has been carried out. The nodes of the net are treated by methods similar to frames. Most of the attention here has focused (CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Con't

ITEM #20, CONTINUED: on the work of Woods and Brachman and the KLONE formalism. The investigators have conducted a number of explorations with their existing experimental metaProlog simulator. This simulation was coded in Edinburgh Prolog and run on Syracuse University's DEC-10 computer. Progress in these areas is discussed in greater detail in this interim report.

AFOSR-TR- 84-0005

LOGIC PROGRAMMING AND KNOWLEDGE BASE MAINTENANCE

Grant AFOSR-82-0292
ANNUAL REPORT
1983

Kenneth A. Bowen

The work conducted this year followed the projections set forth in the grant proposal rather closely. On the theoretical side, we have continued to explore questions of the logical status of some of the standard data structures involved in various artificial intelligence applications involving knowledge bases. The greatest attention has been focused on frames. Here it is felt that classes of concrete frames (e.g. representatives of scenes, or aircraft or ships) can both be axiomatized and also be given efficient, but still logical, implementations in some logic programming systems, specifically those providing a measure of metalevel expressiveness and inference. The basic axiomatization of object-level 'frames with slots' as 'entities with attributes' is possible in simple object-level logic programming systems. However, some of the more powerful aspects of frame-based reasoning utilize default filling of slots, together with "is-a" and "kind-of" hierarchies. The explicit representations of these modes of reasoning requires metalevel facilities in the axiomatizing logic. We have also devoted study and experiment to the development of efficient implementation techniques for these axiomatizations in Prolog-type languages. We have developed a technique which represents the frame as a generalized record structure embodying pointers between frames (which are not directly accessible to the logic programming system). The difficulty in using such representation in logic programming systems lies in providing efficient access to the components in the face of dynamic change of components and the logical requirements of the basic system. Specifically, ordinary Prolog systems would represent updates by term complexes containing embedded copies of the original structure. This is done in order to provide support for exploration of logical alternatives (implemented via backtracking). Our approach causes updates to be immediately written directly to the components of the structure, and preserves information about the original state of the structure via an extension of the so-called "trail mechanism" used in Prolog implementations to reset values of variables during backtracking. Most of our attention to the literature for this area has been directed at the original paper of Minsky, the FRL formalism, and its refinement in Englemann's KNOBS.

Exploration of the axiomatization and representation of semantic nets by similar methods has been carried out. The nodes of the net are treated by methods similar to frames. Most of our attention here has focused on the work of Woods and Brachmann and the KLONE formalism.

We have conducted a number of explorations with our existing experimental metaProlog simulator. This simulation was coded in Edinburgh Prolog and run on Syracuse University's DEC-10 computer. [Due to the heavy loading on this machine, the explorations have not been nearly as extensive as might be desired -- most have been conducted at inconvenient off-hours.] These experiments have fallen into three classes: 1) expression of various quasi-intelligent expert systems tasks; 2) development of basic knowledge base systems; and 3) exploration of reasoning systems for maintenance of knowledge bases. We discuss each of these below.

1) Expression of expert systems tasks. We have coded and run in metaProlog a diagnostic assistant based on the Oak Ridge spills expert of [Rosie]. This experiment demonstrated the usefulness and flexibility of the basic "theory" mechanism of metaProlog in such a setting. First, it permitted a "divide and conquer" approach to the representation of the static background data (e.g. the structure of the drainage network, the contents of the tanks), providing both convenience and clarity of description in the code and increased efficiency of execution. Second, it permitted the clear expression and delineationof the components of expertise employed by the assistant, such as the rules for inferring the identity of the spill material based on its observed properties or the rules for tracing it back in the drainage network. Third, it permitted the representation of the assistant's knowledge of the spill and its possible source as a dynamically evolving theory (i.e., set of assertions). Initially it contains no knowledge of the nature of the spill material and entertains all tanks as possible source and all manholes as potentially requiring investigation. As the program proceeds, assertions regarding the nature and identity of the spill material are gradually added to this theory, and assertions regarding the various tanks as possible sources are gradually deleted.

The second expert system experiment explored fragements of a digital fault-finder based on ideas of [Eshghi]. This example not only exercised the theory mechanism in manners similar to the Oak Ridge spills assistant example, but seriously exercised the control and proof-extraction/processing capabilities of the system. The basic predicate of the system (analogous to eval in LISP systems) is

demo(T, G, C, P).

This predicate holds precisely when P is a proof of goal G from theory T and P is organized according to the control information C. The potential uses of additional control information include: increased efficiency; obtaining solutions not obtainable by default control (e.g. avoiding depth-first run-away in certain settings); obtaining proofs with special properties for further processing; obtaining (as the "proof") the search tree for a goal which fails to be provable relative to a given theory.
These latter two capabilities are central to the fault-finding approach taken by Eshghi. Let the topology of the ciruit under test be represented by the theory c, let the functional behavior of the circuit components be represented by the theory b, and let the laws of propagation of signals in cirucits be encoded in the theory f. This latter theory axiomatizes a predicate predict(I,O) where I describes the values on the input lines and O describes the values on the ouptut lines. Normal simulation of the circuit would be carried out by providing concrete input values in I, letting O be a variable, and invoking

```
demo(c+b+f, predict(I,O)).
```

The essence of the fault-finder is now as follows. Let If,Of be a faulty input-output pair. The system is invoked with

```
demo(c+b+f, predict(If,Of), cntrl, P)
```

where P is a variable, c+b+f represents the union of the theories c, b, and f, and cntrl is a specialized control expression. Since If and Of are instantiated and are an incorrect input-output pair for the circuit, the ordinary call

```
demo(c+b+f, predict(If,Of))
```

would fail in the logic programming sense. However, the particular control information cntrl expresses the request that the search tree for the attempted proof be returned in P and that it be organized in a particular format. This search tree P is then used to guide the generation of a set of theories

```
H = h1, h2, ...
```

which closely resemble c+b, but which attempt to allow the proof of predict(If, Of) to succeed. The set or sequence H is then winnowed down in a manner similar to the classical D-algorithm: distinguishing inputs are generated for pairs of elements of H, and the output of the acutual faulty circuit is used to discard elements of H. The elements of H remaining describe modifications of the original circuit which can account for the observed fault.

The possibly large size of the generated search tree and the resulting set H led to a serious exploration of methods of importing concurrency to the system so as to drive the generation of the proof tree as a producer and the generation and winnowing of H as a consumer (whether in actual or co-routine concurrency).

2) Development of basic knowledge base systems.
This mode of expressing concurrency has been extremely useful in organizing the experimental knowledge bases whose maintenance is the focus this research. Several very small knowledge bases have been constructed using the tools developed thus far, and the required needs for logical maintenance of simple and complex integrity constraints explored. This has led to the inclusion in the system of indexing and control information to support limited (one-step) focused bottom-up processing of logic programming clauses. This mechanism appears to be adequate for the expression and maintenance of object-level integrity constraints ranging from very simple requirements on the types of arguments to maintaining various logical relations across updates (e.g. that, in a personnel database, salaries of supervisors remain greater than salaries of their subordinates, or that raises not exceed a given percentage).
3) Exploration of reasoning systems for knowledge base maintenance.
We are beginning to explore methods of expressing truly meta-level knowledge about the base-level database. The fundamental focus at present is on the maintenance of consistency. We are exploring two means of dealing with this problem which are in some sense complementary and which apparently can be used jointly. The first is the use of truth-maintenance machinery in the style of [Doyle].
This amounts to maintaining sophisticated audit trails of the proofs constructed in operating the base-level system. The second consists in maintaining descriptions of "what the system knows about"-- which may be thought of as very sophisticated secondary indicies. Thus, in a whimsical WWI intelligence database, the system would maintain such meta-level information which would tell it easily that it knew about the home bases of various individuals, but that it did not know anything about the home base of the Red Baron. Thus if an update to the database consisted of "the home base of the Red Baron is Reims", the system could add it knowing that consistency was being maintained, since under these circumstances,

to the School of Computer Science by the Data General Corporation. Though eventually we will return to an implementation on the WICAT. Once this core system is up and stabilized, we will return to the construction of serious example knowledge bases using the newly implemented system, and the exploration of truth maintenance over these knowledge bases.

References

[Doyle] Doyle, Jon, Truth Maintenance Systems for Problem Solving, Report AI-TR-419, Artificial Intelligence Laboratory, MIT, 1978.

[Eshghi] Eshghi, Kave, Application of Meta Level Programming to Fault Finding in Logic Circuits, Proceedings of the First International Logic Programming Conference, Marseilles, 1982, 240-246.

[Rosie] Fain, J., Hayes-Roth, F., Sowizral, H., and Waterman, D., Programming in ROSIE: An Introduction by Means of Examples, Report N-1646-ARPA, RAND Corporation, 1982.

ATE

LMED

8