Paid 88-90
18/2/88

ASSOCIATION FOR AUTOMATED REASONING NEWSLETTER

No. 9 January 1988

From the AAR President, Larry Wos...

If you wish to gain fame and fortune (well, perhaps not fortune) by some (legal) means other than writing papers, the Journal of Automated Reasoning is looking for people to write book reviews. Send your name, address, and email address (and a list of your area of expertise) to Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844, email stevens@anl-mcs.arpa.

Conferences

Southeastern International Conference on Combinatorics, Graph Theory, and Computing

The 19th Southeastern International Conference on Combinatorics, Graph Theory, and Computing will be held at Louisiana State University in Baton Rouge, LA, February 15-19, 1988. A special session on February 17 will focus on issues of implementation, verification, and portability. For further information, write to Professor Clifton E. Ealy, Jr., Department of Mathematics and Computer Science, Northern Michigan University, Marquette, Michigan 49855.

EMCSR 88

The Ninth European Meeting on Cybernetics and Systems Research will take place in Vienna, April 5-8, 1988. The topics include "Robotics and Flexible Manufacturing," "Artificial Intelligence," and "Expert Systems and Approximate Reasoning." A half-day tutorial in artificial intelligence is also planned. For further information write to EMCSR 88 - Secretariat, Osterreichische Studiengesellschaft fur Kybernetik, A-1010 Wien 1, Schottengasse 3, Austria.

CONTROL 88

The Computing and Control Division of the Institution of Electrical Engineers, London, is organizing a conference on control, computing, signal processing, and instrumentation systems. The conference will be held April 13-15, 1988, at the University of Oxford, UK, and will include discussion of the following topics:

- artificial intelligence techniques
- man-machine interfaces
- fault diagnosis
- robotics

Preceding the conference will be a two-day international workshop on Robot Control: Theory and Applications.

For further information write to Conference Services, IEE, Savoy Place, London WC2R 0BL, United Kingdom.

CADE-9

The 9th International Conference on Automated Deduction will be held at Argonne National Laboratory on May 23-26, 1988, in celebration of the 25th anniversary of the discovery of the resolution principle at Argonne in the summer of 1963. For further information, contact

Ewing Lusk and Ross Overbeek, chairmen
CADE-9
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439-4844

New Journal

A new journal, *Medical Expert Systems,* will begin publication in July 1988. Devoted to medical knowledge engineering, the journal will feature original contributions, tutorials, discussions, and reviews on

- AI and medical decision-making
- Formalizing medical knowledge and skill
- Expert system tools in medicine
- Building and evaluating expert systems in medical practice
- Physician-oriented interfaces and environments
- AI and medical databases
- Intelligent instruments and text books
- Expert systems in medical education
- Ethical, social, and economic problems of medical expert systems
- AI news, software, and hardware news

The journal will be published six times a year and will cost $34 for individuals, $68 for institutions. For further information write to Burgverlag, P.O. Box 1247, 4542 Tecklenburg, West Germany.

New Books

Advances in Artificial Intelligence

Advances in Artificial Intelligence is the official publication of the 2nd International Conference on Artificial Intelligence held in France in December 1986. The book comprises 22 papers covering such topics as knowledge acquisition, natural language, reasoning, speech recognition, and man-machine communication. The 328-page book is published by Kogan Page Ltd. of London.

Automated Reasoning: 33 Basic Research Problems

Automated Reasoning: 33 Basic Research Problems by Larry Wos, published by Prentice-Hall, offers problems for research at the doctoral level or greater. The author states that solving any of these problems will result in a substantial increase in the power of automated reasoning programs. Of particular interest, Chapter 2 presents a detailed discussion of the obstacles to effective reasoning, for a computer program or a person. Equally valuable, Chapter 6 presents a rather extensive collection of sets of clauses to permit one to attack questions from a variety of fields in mathematics and logic, specific problems for testing one's progress in attempting to solve one of the 33 posed problems, and solutions to the test problems. This inexpensive book would serve well as a primary text for research topics in automated reasoning and, especially because of Chapter 2, would serve well as a secondary text in a course in artificial intelligence.

The Encyclopedia of Artificial Intelligence

John Wiley & Sons has published a new two-volume reference entitled *The Encyclopedia of Artificial Intelligence*. Over 200 experts have covered virtually every aspect of the field of AI, including blackboard systems, computer chess methods, binary resolution, and causal and default reasoning. Of particular note are the more than 5000 literature references and the extensive cross-referencing by subject and key word.

Automated Theorem Proving

In the second edition of his book *Automated Theorem Proving,* W. Bibel presents a comprehensive treatment of the state of the art in inferential technology based on classical logic. The book combines a rigorous formal style of presentation with extensive illustrative discussions and a number of exercises of increasing difficulty. Thus the book is suited as a text as well as a standard reference book.

An Obvious Solution for a Non-Obvious Problem

(Christoph Walther)

In AAR Newsletter no. 6, Pelletier and Rudnicki discussed problems that are difficult for automated theorem provers and people, not because of size but because of their logical and conceptual complexity. They gave the following example of such a problem [1]:

(1) {¬P(xy), ¬P(yz), ((xz)}

(2) {¬Q(xy), ¬Q(yz), Q(xz)}

(3) {¬Q(xy), Q(yz)}

(4) {P(xy), Q(xy)}

(5) {¬P(ab)}

(6) {¬Q(cd)}

The clauses 5 and 6 are from the negated conclusion of the statement, where a, b, c, and d are Skolem constants. THINKER, the natural deduction system described in [2], required 1808 lines in its proof of this problem and used 115.6 seconds on CPU time on an Amdahl 5860 [1]. Pelletier and Rudnicki discussed the concept of "non-obvious problems" of which the given one is an instance. Obviously it was THINKER's behavior that caused them to classify the given problem in particular and "non-obvious problems" in general as a challenge for automated theorem provers.

We think, however, that there is another quite simple (and also quite obvious) reason why an automated theorem prover based on some kind of breadth-first search (even if improved by some strategies and heuristics) has difficulties with this problem or even fails in finding a solution for it.

The symmetry axiom (3) for the predicate Q causes a naive proof procedure to expand the search space with a great amount of useless and redundant clauses. On computation of generation N+1, each clause of generation N containing a literal with predicate letter Q is recomputed with arguments interchanged. Hence, on computation of generation N+2, each clause of generation N that contains a literal with predicate letter Q has to be recomputed. As a consequence, the search space grows very fast.

But there is also a well-established and quite obvious remedy for this kind of problem. With subsumption, the symmetric closure of Q is still computed, but the recomputation of certain clauses of generation N during the computation of generation N+2 is prohibited; that is, these clauses are removed by subsumption immediately after their generation.

We gave the problem to the Markgraf Refutation Procedure (MKRP) a connection graph-based proof procedure described in [3]. The system initially was given 35 K cons-cells to store the connection graph and was stopped manually when it asked for more memory. Using 134 seconds of CPU time, the system had computed 103 resolvents and generated a search space of 1676 resolution links so far on this unsuccessful run. For the second run, subsumption was used. After generation of a search space of 672 resolution links and 62.2 seconds of CPU time, the empty clause was found with 69 resolution steps.

This result still can be improved by a very simple idea. The symmetry axiom for Q (i.e., clause 3) is removed from the set of clauses and is incorporated into the unification procedure instead. Now, for instance, {Q (xz), Q(cd)} has two most general unifiers (under symmetry): $\{x \leftarrow c , z \leftarrow d\}$ and $\{x \leftarrow d, z \leftarrow c \}$. As a consequence, we have to compute two resolvents from the clauses 2 and 6, namely, $R_1 = \{ \neg Q (cy) , \neg Q (yd) \}$ and $R_2 = \{ \neg Q (dy), \neg Q(yc)\}$. But R_2 is subsumed by R_1 because the empty substitution is a most general unifier (under symmetry) both for $\{ \neg Q(cy), \neg Q(yc) \}$ and for $\{\neg Q(dy), \neg Q(yd)\}$, and therefore R_2 can be removed from the clause set. Note that without built-in symmetry, R_1, which is computed in the first generation, does not subsume R_2, which is computed in the second generation, and both clauses remain in the clause set. With subsumption and built-in symmetry, the MKRP system used 31.1 seconds of CPU time to generate a search space of 380 resolution links before the empty clause was found after 43 resolution steps.

For all runs we used a 3 1/2-year-old version of the MKRP system running on a SIEMENS 7760 machine. All runs were with the set of support strategy, with clauses 5 and 6 as the set of support.

References

[1] Pelletier, F. J., and Rudnicki, P., "Non-Obviousness," AAR Newsletter no. 6, September 1986.

[2] Pelletier, F. J., "Completely Non-Clausal, Completely Heuristically Driven, Automatic Theorem Proving," Technical Report TR82-7, University of Alberta, 1982.

[3] Eisinger, N., and Ohlbach, H. J., "The Markgraf Karl Refutation Procedure (MKRP)," Proceedings 8th Conference on Automated Deduction, *Lecture Notes in Computer Science, Vol. 230,* 1986.

PEANO and the Steamroller Problem

(John Pollock)

J. Pollock has developed an automated theorem prover in first order arithmetic. The prover, called PEANO, begins with Peano's postulates and proves moderately sophisticated theorems. A relatively novel feature of the system is that it is completely automatic, proceeding entirely without human direction. The system does not try to prove hard theorems "from scratch." Rather, it stores its theorems in memory and uses previous theorems in the course of proving new ones. If it does not have what it needs to prove a new theorem, it makes conjectures and tests them against simple models; if they survive the test, it then tries to prove them as lemmas. This approach may lead to the generation of new conjectures and the attempt to prove new lemmas, and so on.

As a rough indication of its power, PEANO does the Schubert steamroller problem (in nonclausal form) in about 1.5 minutes running on a Compaq 386 (this is running uncompiled and unoptimized).

PEANO is an offshoot of work whose ultimate objective is the design of a general-purpose automated reasoning program that does "defeasible reasoning." Currently, Pollock has developed a preliminary version of such a program, called OSCAR.

Solution to Shubert's Steamroller Problem

(Ronald W. Satz)

R. Satz has adapted Prof. Umrigar's F-Prolog system to microcomputers, written an user interface, and successfully tested his Prolog theorem prover program on Shubert's Steamroller Problem. The problem took 88 minutes to solve on a Leading Edge Model M (8088-2) and 13 minutes on a Tandy 4000 (80836). Below is a brief explanation of how Satz's theorem-prover works, followed by a statement of the Steamroller Problem, the input clauses used, and the results.

EXPERT THINKER is a theorem prover designed for MSDOS microcomputers (with 640K of internal memory and a floppy or hard drive). Written and compiled in Arity Prolog and Turbo Prolog, it is an adaptation of the DEC-10 Prolog program called F-Prolog by Professors Umrigar and Pitchumani (Proc. 1985 Symposium on Logic Programming, Boston, Mass., July 1985, pp. 40-47). F-Prolog includes unification with the occurs check, true negation, the capability to use non-Horn clauses, the meson reduction and extension operations, and staged depth-first search. EXPERT THINKER adds an easy-to-use operator interface to this system.

Problem Statement in English: Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal likes to eat either all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants. Therefore there is an animal that likes to eat a grain-eating animal.

Input Clauses:

```
a(X) <= w(X).
a(X) <= f(X).
a(X) <= b(X).
a(X) <= c(X).
a(X) <= s(X).
w(w1).
f(f1).
b(b1).
c(c1).
s(s1).
g(g1).
p(X) <= g(X).
(e(X,Y) or e(X,Z)) <= a(X) & p(Y) & a(Z) & p(V) & m(Z,X) & e(Z,V).
m(X,Y) <= c(X) & b(Y).
m(X,Y) <= s(X) & b(Y).
m(X,Y) <= b(X) & f(Y).
m(X,Y) <= f(X) & w(Y).
e(X,Y) <= w(X) & f(Y).
e(X,Y) <= w(X) & g(Y).
e(X,Y) <= b(X) & s(Y).
e(X,Y) <= b(X) & c(Y).
p(h(X)) <= c(X).
e(X,h(X)) <= c(X).
```

e(X,h(X)) <= c(X).
p(i(X)) <= s(X).
e(X,i,X)) <= s(X).

Solutions: a(f1) & a(b1) & g(g1) & e(f1,b1) & e(b1,g1)

An Indexing Mechanism for Finding More General Formulas

(Bill McCune)

Say we have a large set of formulas, and we wish to quickly access the members that are more general than or identical to a given formula. This note presents a mechanism for accomplishing that task.

The applications for which we are currently using the indexing mechanism are forward subsumption and demodulation. In forward subsumption, one must determine if a given literal is subsumed by any literal in the database. (If nonunit clauses are involved, then one must also attempt various mappings of the literals, but that is beside the point of this note.) In demodulation, one must find the first rewrite rule (or the set of rewrite rules) that can be used to rewrite a given expression. One important aspect of the new indexing procedure is that the matching substitution is constructed as the indexing occurs. In contrast, other indexing mechanisms return formulas that are potential matches; then a routine must be called to verify the match and construct the substitution.

The subsumption index for the literals (x and y are variables)

1. P(e(x,x))
2. P(e(x,e(x,e(x,x))))
3. P(e(x,e(x,e(y,y))))
4. P(e(x,e(y,e(y,x))))
5. P(e(e(x,y),e(y,x)))
6. P(e(e(e(x,y),x),y))

is the tree (leaves are labeled with the literal number)

```
                    P
                    |
                    e
                 /    \
                x       e
           /    |       |   \
       x        e       x     e
     1     /    |       |       \
         x      y       y         x
       /        |       |         |
     e          e       e         y
   /   \        |       |         |
x        y      y       y         x
|        |      |       |         |
x        y      x       x         y
2        3      4       5         6
```

The index tree has the following properties: (1) each path from the root to a leaf represents a literal; and (2) no node has more than one child with the same label. (This ensures a maximum amount of sharing.)

The indexing routine is given a literal and uses a backtracking algorithm that searches the tree from left to right. The backtrack points (choice points) are the variable nodes. When an unbound variable node is visited, the variable is bound to the appropriate term in the given literal. When a bound variable node is visited, the appropriate term in the given literal is checked for identity with the binding. Failure occurs when no child (or no more children) match with the appropriate term in the given literal. Failure causes backtracking to the most recent variable node.

Note that the sharing of the initial segments of the literals in the tree allows some of the actions of binding to be shared. For example, the literal P(e(b,e(b,e(b,b)))) matches literals 2 and 3. The first occurrence of x on the paths to 2 and 3 is the same; the given literal can be matched with both 2 and 3 binding x only once.

We use a similar scheme for finding rewrite rules to apply when demodulating a given expression. For the set of rewrite rules

Li -> Ri,

the Li's (left-hand sides of the rewrite rules) are indexed in the same way as the literals are indexed for forward subsumption. The Ri's are stored in the leaves of the index tree, so that when an Li has been matched with a given term, the current substitution can be applied to the Ri to build a replacement for the given term. If the rewrite rule is conditional and the instance fails the conditions, backtracking occurs.

A variation must be used if one wishes to use an ordered set of rewrite rules or an ordered set of subsuming literals, because maximum sharing imposes some order on the index tree. In the preceding subsumption example, if literal 1 is to be tested between literals 5 and 6, the index tree can be built with that order—but because the amount of sharing is reduced, more space is needed to store the tree, and more time may be required to search the tree.

Performance of the indexing mechanism seems to be very good, especially for forward subsumption, where we have seen speedups as high as 20 over our previous techniques. In one example involving deeply nested equivalential calculus literals, 48 seconds (on a SUN 3/260) were required to test 19,000 literals against a database that averaged 6,000 literals (about 6000 were subsumed). A database of 13,000 literals took 8.5 seconds to construct and required about 1 megabyte of memory. The technique also performs well for large sets of small literals, including sets of ground literals.

For the demodulation application, the speedups are 3-4 over our previous techniques for most problems. It appears that the overhead for the indexing mechanism is small, because for very small sets of rewrite rules, performance is virtually the same as the performance of a similar procedure that does not use indexing.

We have extended the technique to handle full unification, but the results have been disappointing—a lot of complexity is introduced into the procedures, the index requires an enormous amount of memory (more data is stored in each node), and the time performance is not much improved. We also intend to investigate the application of Prolog-like technology to compile the indexing/matching trees. A technical report is being prepared.