AAR NEWSLETTER

No. 7 October 1986

From the President...

As promised, we continue our acyclic production of the AAR newsletter by getting out two issues in two months.

This issue is particularly exciting in that it features two related articles motivated by work reported in an AAR Newsletter half a year ago. We hope that the newsletters will continue to spur on research and comments and that our readers will continue to send in reports of their investigations.

Certainly, research in automated reasoning has a major incentive. I am referring to the $100,000 prize that has come about as a result of the continued and careful work of Dr. W. W. Bledsoe. The prize awaits the first person whose automated theorem-proving program succeeds in proving a very significant theorem from mathematics, on the order of the "fixed point theorem." It must be demonstrated, of course, that the program did the work, and not the person.

But I also find encouraging an article in the August issue of *American Scientist*—an article the uninitiated may find discouraging to the field of automated reasoning. Let me explain. Peter Denning cites the recent work of Dreyfus and Dreyfus (*Mind over Machine*) in which a rather skeptical view is taken about the ability of computers to think. Comment the authors: computers are "incapable of processing information in the same way human experts do." Why do I find this encouraging? Because it confirms what I have been preaching lately. Automated reasoning is different from classical artificial intelligence—we need not imitate the way people solve problems to provide a valuable and powerful reasoning assistant.

New in Print

A two-volume bibliography for the field of artificial reasoning has been compiled by H. Rylko. The bibliography, entitled *Artificial Intelligence: Bibliographic Summaries of the Select Literature*. contains more than 200 capsule reviews of AI documents, including coverage of technical and research reports. The volumes may be obtained from the Report Store, 910 Massachusetts, Room 503, Lawrence, KS 66044-2975. The cost is $190.

Call for Papers

Following the success of the 6th International Workshop on Expert Systems and Their Applications, the Agence de l'Informatique has scheduled the 7th Workshop for May 13-15, 1987.

The meeting, one of the principal international events on applications of artificial intelligence, will provide a forum for the presentation of new implementations of expert systems and basic tools and techniques for building such systems.

Submitted papers, due December 15, should be sent to

Jean-Claude Rault
Agence de l'Informatique
Tour Fiat Cedex 16
92084 Paris-La Défense, France.

A Note on Smullyan's Birds

(from Ross Overbeek and Barney Glickfeld)

Raymond Smullyan recently published a marvelously readable introduction to combinatory logic titled *To Mock a Mockingbird.* The book develops the subject by posing a graduated set of problems. We have found the problems quite tractable for theorem provers that include paramodulation, demodulation, and subsumption. As an outgrowth of our interest in the area, we developed a program named *birdbrain* that takes input in the form of electronic mail addressed to *bb@anl-mcs.arpa* and produces the appropriate responses in the form of return mail. The program seems capable of proving the vast majority of problems in Smullyan's book (we are working through the entire problem set now and have yet to encounter a problem that the system failed to answer properly).

What we believe is unique about our system is that there is no human intervention—it reads its own mail, submits problems to a theorem prover, and sends back its responses. The system includes lemmas on over 400 combinators (which we think of as "birds," in keeping with Smullyan's formulation of the problems). At this point, the system exists only as a prototype, and occasionally produces unexpected answers to ill-formed queries. However, we invite you to submit queries to it and experiment with its capabilities. To get more information on the system, you should simply mail a message composed of the single word *help to bb@anl-mcs.arpa.* The system should send a users manual via return mail, and you can explore its characteristics from there. The manual is self-explanatory and includes instructions on how to forward complaints and suggestions to "birdbrain's keeper". Occasionally, we have found that return mail fails to get through; so, if you fail to get any response, please call Ross Overbeek at 312-972-7856.

mail bb@anlmcs.arpa@munnari.oz

A Solution to the Mockingbird Problem
(from Jean-Luc Rémy and Pierre Réty)

In the AAR Newsletter No. 5, Ross Overbeek submitted a mockingbird problem extracted from the book *To Mock a Mockingbird,* by Raymond Smullyan.

The problem consists of solving an equation in an equational theory. The equation is

$S(a,x)=x,$

where a is a constant and x is the unknown. The equational theory is given by the two following axioms:

$S(x,S(y,z))=S(f(x,y),z)$ composition condition

$S(x,x)=S(m,x)$, where m is a so-called mockingbird.

Overbeek ran the problem on the Argonne theorem prover ITP, and the program derived a short, elegant solution in just a few seconds.

We suggest here another solution using a rewriting approach. Indeed, there exists a complete way of solving equations, using narrowing, when the axioms can be organized into a rewriting system with the two properties of finite termination and confluence. For a formal description of the process, we refer to the paper by J. P. Jouannaud, C. Kirchner, and H. Kirchner entitled "Incremental construction of unification algorithms in equational theories" (Ref. 1). Also of interest is the paper by C. Kirchner et al. entitled "NARROWER: a new algorithm for unification and its application to logic programming" (Ref. 2).

In this example, the second equation prevents the finite termination, since, for $x=m$, we have an identity. Also, trying to complete the two equations into a confluent system does not seem to be successful. However, it is still possible to apply the narrowing process on the equation to check if there exists any solution. If the answer is yes, we are done. On the other side, if the answer is no, we cannot conclude, since our method is sound but not complete. It turns out that two narrowing steps are enough to solve the equation. Here is the description of these steps. (For the sake of simplicity we use different variables for the axioms and the equation to be solved. Therefore, we rename it $S(a,x1)=x1$.

First step. We unify the left-hand sides of the equation and the composition condition, using the substitution $[x\rightarrow a,\ x1\rightarrow S(u,v),\ y\rightarrow u,\ z\rightarrow v]$. We apply the substitution to get

$S(a,S(u,v))=S(u,v).$

Now we apply the composition condition

$S(f(a,u),v)=S(u,v).$

Second step. We unify the left-hand sides of the equation and the second axiom, using the substitution $[u\rightarrow u1,\ x\rightarrow f(a,u1),\ v\rightarrow f(a,u1)]$.

After applying the substitution, we get

$S(f(a,u1), f(a,u1)) = S(u1, f(a,u1))$.

Now we apply the second axiom

$S(m, f(a,u1)) = S(u1, f(a,u1))$.

This last equation has a trivial solution, namely, $u1=m$. Therefore, grouping altogether the substitutions, we get a solution of the original equation

$x1 = S(m, f(a,m))$.

We intend to look at other problems in Smullyan's book to see if narrowing, using incomplete rewriting systems, can give solutions.

Note

Much of our research in this area is related to work on the REVE system. Included in REVE is REVE2, which manipulates classical term rewriting systems to complete them into confluent ones, to solve equations and to prove assertions. Also included is REVEUR3, which is concerned with rewriting systems modulo equations (such as associativity and commutativity), and REVEUR4, which is concerned with conditional rewriting systems.

References

1. J. P. Jouannaud, C. Kirchner, and H. Kirchner, "Incremental construction of unification algorithms in equational theories," *Lecture Notes in Computer Science,* no. 154, Springer-Verlag, Berlin, 1983.

2. C. Kirchner, H. Kirchner, P. Lescanne, and P. Réty, "NARROWER: A new algorithm for unification and its application to logic programming," *Lecture Notes in Computer Science,* no. 202, Springer-Verlag, Berlin, 1985.

A Proof of a Non-Obvious Theorem
(from Bill McCune)

In AAR Newsletter No. 6, Pellitier and Rudnicki presented a non-obvious theorem. The negation of the theorem in clause form is as follows:

1. $\neg Pxy \ \neg Pyz \ Pxz$
2. $\neg Qxy \ \neg Qyz \ Qxz$
3. $\neg Qxy \ Qyx$
4. $Pxy \ Qxy$
5. $\neg Pab$
6. $\neg Qcd$

The LMA-based theorem prover *tp0* produced a refutation of this clause set in just under a minute on a Sun 3 workstation. The proof resulted in the generation of 537 clauses, of which 350 were immediately subsumed; of the remaining 187, 74 were subsumed later on. There are 17 steps in the proof, and the proof has depth 6. The inference rule was binary resolution, and the search strategy was unit preference (without any additional ordering strategy) with clauses 5 and 6 in the initial set of support. These numbers can be compared to Pellitier's results using *THINKER* on an Amdahl 5860—that proof required the generation of 1808 formulas, and it took 115.6 seconds.

Binary resolution with unit preference and set of support is effective on other non-Horn problems. For example, the solution presented by Lusk and Overbeek of the Salt and Mustard puzzle ["Non-Horn Problems", *JAR*, 1(1) (1985)] required the generation of more than 32,000 clauses. Using the same input clauses, binary resolution with unit preference and set of support produced the solution after generating 2,689 clauses.

A New Journal

A new journal, entitled *Applied Artificial Intelligence,* has been announced by editor-in-chief Robert Trappl. The journal will cover topics from applied research to actual applications:

- uses of expert systems
  natural language systems
  speech
  vision
  robotics

- evaluations of existing AI systems and tools, emphasizing
  comparative studies and
  user experiences

- theoretical research relevant to potential applications

Clausal Form Translator
(from Peter Malkin)

As part of my senior thesis, I have developed a procedure for translating first-order predicate calculus to clausal form. The procedure involves 16 steps:

1. Enter all sentences into a linked list of parse trees containing formulas.
2. Universally quantify all unbound variables.
3. Negate the conclusion.
4. Drive in negations into the predicate level.
5. Rename constants, variables, and functions for use by the program.
6. Put sentences into pure form.
7. Put sentences into conjunctive normal form.
8. Eliminate all conjunctions making the conjuncts separate sentences.
9. Again, rename all necessary variables.
10. Put expression into prenex form.
11. Skolemize the sentences.
12. Put sentences into conjunctive normal form.
13. Eliminate all conjunctions by making the conjuncts separate sentences.
14. Again, rename all necessary variables.
15. Remove universal quantifiers.
16. Eliminate disjunctions making sentences into clauses.

When these steps have been completed, the original argument is in clausal form. This form consists of sets of literals implicitly conjoined, the literals in each of the sets being implicitly disjoined.

The procedure has been implemented in a program comprising seven separate files. Having the program stored this way enables the user to change a single section of code, thus making compilation and testing easier. The program also offers other advantages. The clausal logic form it produces can be used by ITP, a powerful theorem prover developed at Argonne which allows experimentation with new strategies and techniques. Moreover, the passage from first-order predicate form to clausal form facilitates other well-known theorem-proving procedures, including the Herbrand method, ground resolution, and resolution.

For further information about this translation program, write to

Peter K. Malkin
8 Brewer Street
Cambridge, Massachusetts 02138