ASSOCIATION FOR AUTOMATED REASONING NEWSLETTER

No. 13 September 1989

From the AAR President, Larry Wos ...

I have two announcements about the *Journal of Automated Reasoning* that are of definite interest to our AAR members.

First, a reminder that AAR members are entitled to a discount if they subscribe to the *Journal.* The usual fee for a year's subscription is $124.50 for institutions, $49.00 for individuals who are *not* members of AAR, and $38.00 for individuals who *are* members of AAR.

Second, I am pleased to announce that the *Journal of Automated Reasoning* is planning two special issues. J Moore is editing a Special Issue on Verification, to appear as Vol. 5, no. 4, in late 1989. The issue will include articles on ''An Approach to Systems Verification,'' by Bevier, Hunt, Moore, and Young; ''Microprocessor Design Verification,'' by Hunt; ''A Mechanically Verified Language Implementation,'' by Moore; ''A Mechanically Verified Code Generator,'' by Young; and ''Kit and the Short Stack,'' by Bevier. Michael McRobbie is editing a Special Issue on Nonclassical Logic, to be published in early 1990.

The *Journal* might be particularly attractive to AAR members because of two columns that appear in each issue: Basic Research Problems, which presents problems for research in or about automated reasoning; and the Problem Corner, which presents test problems and results of experiments involving automated reasoning.

From the AAR Secretary, William McCune ...

To conserve resources, we wish to send fewer dues reminder notices. Next to your name on the mailing label is the month in which your AAR membership expires. Dues are $5.00 for one year, $9.00 for two years, and $12.00 for three years. Please make dues payable to the Association for Automated Reasoning, and send them to

William W. McCune, AAR
MCS-221
Argonne National Laboratory
Argonne, IL 60439-4844

A Challenge Problem and a Recent Workshop

L. Wos, Argonne National Laboratory

A workshop on automated reasoning was held at Argonne National Laboratory on August 1-4, 1989. The object was to explore research topics and promote research in the field. I can delightedly report that vigorous exchange occurred focusing on Robbins algebra, combinatory logic, program synthesis, and other topics. Such a good time was had by all that we plan next year to have a similar workshop at Argonne. We are accruing notes that summarize some of the activities and results, notes that may be made available by electronic mail, if all goes as expected.

Here is a challenge problem posed—and solved—during the workshop. The problem focuses on a set of equalities for the combinators B, M, and S, respectively satisfying the following three equations, where expressions are left associated unless otherwise indicated.

$Bxyz = x(yz)$; $Mx = xx$; $Sxyz = xz(yz)$;

If H is a fixed point combinator when $Hx = x(Hx)$ for all combinators x, the challenge problem asks whether an expression exists in which B, M, and S are the only terms (outside of parentheses and commas) such that the expression is a fixed point combinator.

We present the answer at the end of this newsletter.

Conferences

AAAI-90

AAAI-90 is the 8th national conference sponsored by the American Association for Artificial Intelligence. The purpose of the conference is to promote research of the highest caliber in Artificial Intelligence and to promote scientific interchange among AI researchers and practitioners. Contributions are welcome in 14 different content areas:

AI and education
Automated reasoning
Cognitive modeling
Impacts of AI technology
Knowledge acquisition,
  expert systems design
Knowledge representation
Commonsense reasoning
Machine architectures, computer languages
Machine learning
Natural learning
Perception and signal understanding
Philosophical foundations
Robotics
User interfaces

Papers for review should be single-spaced and no more than 11 pages (excluding bibliography). Each paper should have a separate title page giving the title, names and addresses of all authors, a 200-word abstract, and clear designation of the content area. Authors should submit six copies of their manuscripts by February 20, 1990, to AAAI-90, 445 Burgess Drive, Menlo Park, CA 94025-3496.

Complete Linear Derivation Systems for General Clauses

Geoff Sutcliffe, The University of Western Australia

The first refutation completeness result for a linear derivation system for general clauses was presented by Luckham (1968). Since then, the refutation completeness of several linear derivation systems for general clauses has been established. The increased use of Horn clauses, starting with Kuehner's SNL theorem prover (1972), has led to the development of similar completeness results for input derivation systems for Horn clauses. This note provides a general method for obtaining these latter completeness results from their linear counterparts.

DEFINITION: Given a set S of clauses and a *top clause* C_0 in S, a linear derivation of C_n from S is a derivation where

(i) for $i = 0,1,...,n-1$, C_{i+1} is inferred from the *center parent* clause C_i and a set of *side parent* clauses; and

(ii) each side parent clause either is in S or is a C_j for some $j < i$.

DEFINITION: A linear (input) refutation of S is a linear (input) derivation of the empty clause from S.

The refutation completeness of several linear derivation systems for general clauses has been established. These derivation systems use various combinations of inference rules, e.g., binary resolution (Luckham 1968), paramodulation, and factoring (for E-unsatisfiable sets including the reflexive and functionally reflexive axioms) (Chang 1971). These two derivation systems are also compatible with the set of support strategy (Wos 1965).

A complete linear derivation system for general clauses may usually be transformed into a complete input derivation system for Horn clauses. For this transformation to be possible, the derivation system must conform to the following restrictions:

(i) It should be compatible with the set of support strategy.

(ii) If one parent clause of an inference is negative, the other parent clauses must be non-negative, and the inferred clause must then be negative.

Let S be an unsatisfiable set of Horn clauses, and let T be the set of negative clauses in S. A minimally unsatisfiable set of Horn clauses contains one negative clause (Henschen 1974), so T is non-empty and $S-T$ is satisfiable. A complete linear derivation system that conforms to these restrictions will produce a refutation of S, with the top clause chosen from T. By induction, every center parent clause in this derivation is negative, and every side parent clause is non-negative. This implies that every side parent clause is in $S-T$ and that the refutation is an input one. An input version of the derivation system is thus complete for Horn clauses, where the top clause is restricted to be a negative clause. Note that in such input derivation systems negative clauses need not be considered for selection as side parent clauses.

This result provides immediate proof of the refutation completeness of an input derivation system for Horn clauses if a corresponding completeness result is known for a linear derivation system for general clauses. For example, the two complete linear derivation systems given as examples conform to the restrictions listed. Hence, input versions of the systems are complete for Horn clauses.

References

Chang, C. L., and Slagle, J., Completeness of linear refutation for theories with equality, J. ACM 18, no. 1 (1971) 126-136.

Henschen, L., and Wos, L., Unit refutations and Horn sets, J. ACM 21, no. 4 (1974) 590-605.

Kuehner, D., Some special purpose resolution systems, *Machine Intelligence 7*, Edinburgh University Press, 1972, pp. 117-128.

Luckham, D., Refinement theorems in resolution theory, Symposium on Automatic Demonstration, Versailles, 1968, pp. 163-190.

Wos, L., Robinson, G. A., and Carson, D. F., Efficiency and completeness of the set of support strategy in theorem proving, J. ACM 12, no. 4 (1965) 536-541.

Another Term Rewriting-based Proof of the Non-Obvious Theorem

F. Baj, M. P. Bonacina, M. Bruschi, and A. Zanzi
Universita degli Studi di Milano

A term rewriting proof of a "non-obvious" theorem was given in *AAR Newsletter* No. 11 by Wilkerson and Smith. Their prover implements the N-strategy method (Hsiang 1985) for first-order logic. We have implemented in Prolog a theorem prover for first-order logic with equality, called ENprover. Our prover provides N-strategy and EN-strategy, the extension of N-strategy to first-order logic with equality defined by Hsiang (1987). In these methods the theorem is proved refutationally by simplification and superposition in a Boolean ring.

We have given the non-obvious theorem to our prover. The problem is defined by the following clauses:

1. $\neg Q(C,D)$
2. $\neg P(A,B)$
3. $Q(x,y) \vee \neg P(x,y)$
4. $Q(x,y) \vee \neg Q(y,x)$
5. $Q(x,y) \vee \neg Q(x,z) \vee \neg Q(z,y)$
6. $P(x,y) \vee \neg P(x,z) \vee \neg P(z,y)$

The input clauses are transformed into the following rewrite rules:

1. $Q(C,D) \rightarrow 0$
2. $P(A,B) \rightarrow 0$
3. $Q(x,y)P(x,y) + P(x,y) + Q(x,y) \rightarrow 1$
4. $Q(x,y)\ Q(y,x)\ Q(z,y) + Q(y,x) \rightarrow 0$
5. $Q(x,y)\ Q(x,z)\ Q(z,y) + Q(x,z)Q(z,y) \rightarrow 0$
6. $P(x,y)P(x,z)P(z,y) + P(x,z)P(z,y) \rightarrow 0$

ENprover gives the following proof using a breadth-first strategy:

1. theorem: $Q(C,D) \rightarrow 0$

```
2. theorem: P(A,B) -> 0
3. theorem: Q(x,y)P(x,y) + P(x,y) + Q(x,y) -> 1
4. theorem: Q(x,y)Q(y,x) + Q(y,x) -> 0
5. theorem: Q(x,y)Q(x,z)Q(z,y) + Q(x,z)Q(z,y) -> 0
6. theorem: P(x,y)P(x,z)P(z,y) + P(x,z)(Pz,y) -> 0
8. 1,5-en: Q(C,x)Q(x,D) -> 0
9. 1,3-en: P(C,D) -> 1
10. 2,6-en: P(A,x)P(x,B) -> 0
11. 2,3-en: Q(A,B) -> 1
14. 8.4-en: Q(x,C)Q(x,D) -> 0
15. 8,3-en: P(C,x)Q(x,D) + Q(x,D) -> 0
16. 10,3-en: Q(A,x)P(x,B) + P(x,B) -> 0
18. 14,16-en: P(C,B)Q(A,D) -> 0
20. 14,3-en: P(x,C)Q(x,D) + Q(x,D) -> 0
21. 10,15-en: Q(B,D)P(A,C) -> 0
23. 21,20-en: Q(A,D)Q(B,D) -> 0
31. 18,6-en: P(C,x)P(x,B)Q(A,D) -> 0
45. 23,5-en/[11]-simp
    : Q(B,D) -> 0
63. 31,16-en/[9]-simp
    : P(D,B) -> 0
73. 45,4-en: Q(D,B) -> 0
81. 63,3-en/[73]-simp
    : 1 -> 0
Proof Length: 22
Rules Generated: 81
Rules Deleted: 45
Rules Simplified: 3
Total Runtime: 221 sec (on an IBM RT-6150)
```

Notes:

r. m,n-en means that rule r has been obtained by an en-superposition step between rules m and n. Notice that, since equality is not present in this theorem, all the en-superposition steps are just n-superposition steps.

r. m,n-en/[s1,...sn] means that the result of en-superposition between rules m and n has been simplified to rule r by rules s1,...,sn.

A rule is deleted when it is simplified by another rule to 0 -> 0.

The proof is made of 22 rules, including the 6 initial ones. When 1 -> 0 was found, there were 33 rules in the database.

The second example is taken from Pelletier (1986). It is the exercise no. 55 (''Who Killed Aunt Agatha,'' L. Shubert). This example shows how ENprover works with equality. Here are the input clauses:

1. L(A1)

2. K(A1,A)
3. L(A)
4. L(B)
5. L(C)
6. ¬L(x) V [x=A] V [x=B] V [x=C]
7. ¬ L (x,y) V H(x,y)
8. ¬ K(x,y) V ¬ R (x,y)
9. ¬ H (A,x) V ¬ H(C,x)
10. [x=B] V H(A,x)
11. R(x,A) V H(B,x)
12. ¬ H(A,x) V H(B,x)
13. ¬ H(x,F1(x))
14. ¬ [A=B]
t15. ¬(A,A)

where t15 is the negation of the theorem.

The following is the entire proof, including the rules for the input clauses.

1. axiom: L(A1) -> 1
2. axiom: K(A1A) -> 1
6. axiom: [x=C][x=A]L(x)[x=B] + [x=A]L(x)[x=B] +
 [x=C][x=A]L(x) + [x=C]L(x) [x=B] + [x=A]L(x) +
 L(x) [x=B] + [x=C]L(x) + L(x) -> 0
7. axiom: H(x,y)K(x,y) + K(x,y) -> 0
8. axiom: R(x,y)K(x,y) -> 0
9. axiom: H(C,x)H(a,x) -> 0
10. axiom: H(A,x) [x=B] + [x=B] + H(A,x) -> 1
11. axiom: H(B,x)R(x,A) + R(x,A) + H(B,x) -> 1
12. axiom: H(B,x)H(A,x) + H(A,x) -> 0
13. axiom: H(x,F1(x)) -> 0
14. axiom: [A=B] -> 0
t15. theorem: K(A,A) -> 0
16. 7.2-en: R(A1,A) -> 0
17. 2,7-p: H(A1,A) -> 1
21. 13,12-en: H(A,F1(B)) -> 0
23. 14,10-en: H(A,A) -> 1
25. 21,10-en: [F1(B)=B] -> 1
30. 23,9-p: H(C,A) -> 0
33. 25,13-para: H(B,B) -> 0
36. 33,11-en: R(B,A) -> 1
39. 3,1-en: [A1=A][A1=C][A1=C] + [A1=B][A1=C] +
 [A1=A][A1=C] + [A1=B] + [A1=C] +
 [A1=B] + [A1=A] -> 1
48. 39,30-para/[17]-simp:
 [A1=B][A1=A] + [A1=A] + [A1=B] -> 1
93. 48,36-para/[16]-simp:
 [A1=A] -> 1
94. 93,2-simp: K(A1,A1) -> 1
t96. [93,94]/t15-simp: 1 -> 0

Proof Length: 25
Rules Generated: 96
Rules Deleted: 45
Rules Simplified: 9
Total Runtime: 1266 sec. (on an IBM RT-6150)

where
r. **m,n-para** means that rule r has been obtained by a para-superposition step between rules m and n.

r. **m,n-p** means that rule r has been obtained by a p-superposition step between rules n and m.

When the rule 1 -> 0 was found, there were 42 rules in the database.

It should be noted that input axioms 2, 3, and 4 are not really necessary to obtain a contradiction.

To compare the proof given by the EN-strategy method with a resolution+paramodulation proof, we have given this last example to the OTTER theorem prover (McCune 1989). We selected as inference rules the following: para_into, para_from, para_from_left, para_from_right, and hyperresolution. The proof report gave the following statistics:

```
clauses input                          15
clauses given                          88
clauses generated                      5,716
demod and eval rewrites                0
tautologies deleted                    27
clauses forward subsumed
 (subsumed by sos)                     1,405
clauses kept                           631
empty clauses                          1
clauses back subsumed                  465
clauses not processed                  3

times (seconds)--------(on an AMSTRAD PC 2386/65 (80386 20 MHa))
run time                               145.04
para_into time                         5.60
part_from time                         2.96
for_sub time                           95.61
```

It should be noticed that OTTER is much faster than ENprover: a superposition step is intrinsically more time-consuming than a resolution step since the first requires unification and reduction, and the second only unification. Moreover, not much work has been done yet to increase efficiency of ENprover.

The above proofs show the power of ordered inference rules and simplification inference rules. ENprover implements para-superposition, a restricted form of paramodulation, where an equation s = t is applied in the s -> t direction only if s > t in a complete simplification ordering. Para-superposition generates far fewer rules than paramodulation. Furthermore, a large number of generated rules are deleted by simplification.

References

Hsiang, J., Refutational theorem proving using term rewriting systems, Artificial Intelligence 25 (1985) 255-300.

Hsiang, J., Rewrite methods for theorem proving in first order theory with equality, J. Symbolic Computation 3 (1987) 133-151.

McCune, W. W., OTTER 1.0 Users' Guide, Argonne National Laboratory Report, ANL-88-44 (January 1989).

Pelletier, F. J., Seventy-five problems for testing automatic theorem provers, J. of Automated Reasoning 2 (1986) 191-216.

A Solution to the B, M, and S Problem

William McCune, Argonne National Laboratory

The problem, stated here in the notation required by our theorem prover OTTER [McCune 1989], is to refute the following set (x, y, and z are variables, a is the apply function, and f is a Skolem function).

```
(x = x).
(a(a(a(B,x),y),z) = a(x,a(y,z))).
(a(M,x) = a(x,x)).
(a(a(a(S,x),y),x) = a(a(x,z),a(y,z))).
(a(x,f(x)) != a(f(x),a(x,f(x)))) | $Answer(x).
```

The fixed point combinator

```
a(a(S,a(M,a(a(B,a(S,B)),M))),x)    (full notation)
S(M(B(SB)M))x                      (abbreviated notation)
```

along with 848 others was found with the *kernel method* of Wos and McCune (1989).
Seven runs were required, using a total of several minutes CPU time.

References

McCune, W., OTTER 1.0 Users' Guide, Argonne National Laboratory Report, ANL-88-44 (January 1989).

Wos, L., and McCune, W., Searching for fixed point combinators by using automated theorem proving: A preliminary report, Argonne National Laboratory Report, ANL-88-10 (1988).