ASSOCIATION FOR AUTOMATED REASONING
NEWSLETTER

No. 27 October 1994

From the AAR President, Larry Wos...

Highlighted in this issue is an article by Li Dafa, who recently found a significantly shorter proof to the well-known halting problem. As many of you know, one of my favorite topics of research has been using an automated reasoning program to find shorter proofs. Not only are such proofs often more aesthetically pleasing, but they may have practical application, for example, in program synthesis.

With his enhanced program ANDP, Dafa more than halved the previous automated proof of the problem (reducing the 93-step proof to 43 steps), coming remarkably close to the 38-step manual proof obtained by Burkholder in 1987. I encourage researchers to attempt to obtain (not by hand) an even shorter proof in resolution style—and to submit the results to the *AAR Newsletter*.

I also encourage readers to test their programs on the problem submitted by Zhang regarding finite model generators, as well as on the two challenge problems I've offered in Robbins algebra.

New Journal

A new journal, *Multiple-Valued Logic*, has been established and has issued a call for papers. The aim is to publish and disseminate knowledge internationally in the area of multiple-valued logic. Specific topics include mathematical aspects of MVL, engineering aspects of MVL, MVL and automated reasoning, computer science and MVL, theoretical and practical aspects of fuzzy logic, and philosophical aspects of MVL. For further information, contact the managing editors Dan A. Simovici, e-mail: dsim@cs.umb.edu, and Ivan Stojmenovic, e-mail: ivan@csi.uottawa.ca.

The Formulation of the Halting Problem Is Not Suitable for Describing the Halting Problem

Li Dafa
Department of Applied Mathematics, Tsinghua University, Beijing 100084, China
Fax: (861) 2562768

L. Burkholder gave a first-order formulation of the halting problem in [2]. It includes four premises (given in the appendix). Let P_i represent the ith premise, i = 1,2,3,4. The conclusion

is that no algorithm exists for solving the halting problem. It is difficult, however, to find an automated proof of the problem in resolution style; applications of OTTER (Argonne's theorem prover) and ENprover were unsuccessful [1]. In 1993, we presented a 93-step mechanical proof of the problem [4] in natural deduction style, which we obtained by using the ANDP system. Since then, we have improved the program in both power and efficiency and recently obtained a 43-step proof of the problem. (Note: Burkholder obtained a 38-step manual proof in 1987 [2].)

P_1, P_3, and P_4 are implications (see the appendix). We list P_4 as follows:

$$(\exists v)[Cv \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow [H_2vy \wedge Ovg]] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_2vy \wedge Ovb]]]] \rightarrow$$
$$(\exists u)[Cu \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow \neg H_2uy] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_2uy \wedge Oub]]]]$$

Let P_1 be $A_1 \rightarrow S_1$; let P_3 be $A_3 \rightarrow S_3$. Then P_4 is $S_3 \rightarrow S_4$ (note that the consequence of P_3 is the antecedent of P_4; see the appendix), where S_4 is

$$(\exists u)[Cu \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow \neg H_2uy] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_2uy \wedge Oub]]]].$$

Clearly, the conclusion is $\neg A_1$ (see the appendix).

Analyzing Steps 17–29 of our 43-step proof, we can easily see that the consequence of S_4 of P_4 is a contradiction. (Burkholder gave a 17-step manual proof of the fact that the consequence S_4 is a contradiction [2].) First, eliminate the top quantifier $(\exists v)$ of S_4, substitute a new constant c for all occurrences of v in the scope of $(\exists v)$, and obtain $Cc \wedge (\forall y)[[Cy \wedge H_2yy \rightarrow H_2cy] \wedge [Cy \wedge \neg H_2yy \rightarrow H_2cy \wedge Ocb]]$. Then, eliminate the universal quantifier $(\forall y)$. Substitute c for all occurrences of y in the scope of $(\forall y)$. Finally, obtain $Cc \wedge [Cc \wedge H_2cc \rightarrow \neg H_2cc] \wedge [Cc \wedge \neg H_2cc \rightarrow H_2cc \wedge Ocb]$, which is a propositional contradiction. Since the consequence S_4 is a contradiction, P_4 is equivalent to the negation of its antecedent S_3, that is,

$$P_4 = \neg S_3 = \neg[(\exists v)[Cv \wedge (\forall y)[[Cy \wedge H_2yy \rightarrow H_2vy \wedge Ovg] \wedge [Cy \wedge \neg H_2yy \rightarrow H_2vy \wedge Ovb]]].$$

P_3 says that if program w decides the termination of a program y on y as an input, there is a program v that can terminate only if a program halts on itself (see the appendix). But the premise P_4 asserts that there is no program v that can terminate only if a program halts on itself. Why? Where does the premise P_4 come from? Clearly P_4 cannot be asserted; P_3 is not necessary. Therefore, the formulation in [2] is not suitable for describing the halting problem. Moreover, given that the consequence S_4 is a contradiction, the halting problem as the 76th automated theorem-proving problem [3] becomes trivial. Let us summarize as follows.

S_4 is a contradiction; therefore, $S_3 \rightarrow S_4 = \neg S_3$. We wish to draw the conclusion $\neg A_1$ from $P_1 \wedge P_2 \wedge P_3 \wedge P_4$. Let us negate the conclusion. We then derive a contradiction from the negation of the conclusion and the four premises. Clearly,

$$A_1 \wedge P_1 \wedge P_2 \wedge P_3 \wedge P_4 = A_1 \wedge P_1 \wedge P_2 \wedge P_3 \wedge \neg S_3 =$$
$$A_1 \wedge [A_1 \rightarrow S_1] \wedge P_2 \wedge P_3 \wedge \neg S_3 \Rightarrow S_1 \wedge P_2 \wedge P_3 \wedge \neg S_3 =$$
$$S_1 \wedge P_2 \wedge [A_3 \rightarrow S_3] \wedge \neg S_3 \Rightarrow S_1 \wedge P_2 \wedge \neg A_3,$$

which is a contradiction trivially.

Analyzing the proofs of the halting problem [5], we suggest a formulation of the halting problem as follows. It includes the first two premises P_1, P_2 in [2] and the following formula P_3'.

P_3' says that if program w decides the termination of any program on any input, then there is a program v such that for all programs y, if w halts given an input pair $< y, y >$ and prints out "g" (for good), then v does not halt on given input y. If w halts given an input pair $< y, y >$ and prints out "b" (for bad), then v halts on given input y and prints out "b".

P_3' is as follows:

$$(\forall w)[Cw \wedge (\forall y)(\forall z)[[Cy \wedge H_2yz \rightarrow H_3wyz \wedge Owg] \wedge [Cy \wedge \neg H_2yz \rightarrow H_3wyz \wedge Owb]]$$
$$\rightarrow (\exists v)[Cv \wedge (\forall y)[[Cy \wedge H_3wyy \wedge Owg \rightarrow \neg H_2vy] \wedge [Cy \wedge H_3wyy \wedge Owb \rightarrow H_2vy \wedge Ovb]]]].$$

It is easy to get a manual proof of the new formulation of the halting problem as follows. First, negate the conclusion $\neg A_1$. Then, from the negation and P_1, obtain S_1 by the MP rule. We need only to derive a contradiction from $S_1 \wedge P_2 \wedge P_3'$.

Eliminate the top existential quantifier of S_1. The next step is

(1) $Ca \wedge (\forall y)[Cy \rightarrow (\forall z)Dayz]$

Apply US to premise P_2, and substitute a for w. Then

(2) $Ca \wedge (\forall y)[Cy \rightarrow (\forall z)Dayz] \rightarrow$
$(\forall y)(\forall z)[[Cy \wedge H_2yz \rightarrow H_3ayz \wedge Oag] \wedge [Cy \wedge \neg H_2yz \rightarrow H_3ayz \wedge Oab]]$

Apply MP to Steps 1 and 2. Then

(3) $(\forall y)(\forall z)[[Cy \wedge H_2yz \rightarrow H_3ayz \wedge Oag] \wedge [Cy \wedge \neg H_2yz \rightarrow H_3ayz \wedge Oab]]$

Apply US to premise P_3', and substitute a for w. Then

(4) $[Ca \wedge (\forall y)(\forall z)[[Cy \wedge H_2yz \rightarrow H_3ayz \wedge Oag] \wedge [Cy \wedge \neg H_2yz \rightarrow H_3ayz \wedge Oab]]$
$\rightarrow (\exists v)[Cv \wedge (\forall y)[[Cy \wedge H_3ayy \wedge Oag \rightarrow \neg H_2vy] \wedge [Cy \wedge H_3ayy \wedge Oab \rightarrow$
$H_2vy \wedge Oab]]]]$

Apply the simplification rule to Step 1. Then

(5) Ca

First obtain the conjunction of Steps 1 and 3. Then apply MP to the conjunction and Step 4.

(6) $(\exists v)[Cv \wedge (\forall y)[[Cy \wedge H_3ayy \wedge Oag \rightarrow \neg H_2vy] \wedge$
$[Cy \wedge H_3ayy \wedge Oab \rightarrow H_2vy \wedge Ovb]]]]$

Eliminate the top existential quantifier $(\exists v)$ of Step 6, and substitute the constant d for the existential variable v. Then

(7) $Cd \wedge (\forall y)[[Cy \wedge H_3ayy \wedge Oag \rightarrow \neg H_2dy] \wedge [Cy \wedge H_3ayy \wedge Oab \rightarrow H_2dy \wedge Odb]]]$

Apply the simplification rule to Step 7. Then

(8) Cd
(9) $(\forall y)[[Cy \wedge H_3ayy \wedge Oag \rightarrow \neg H_2dy] \wedge [Cy \wedge H_3ayy \wedge Oab \rightarrow H_2dy \wedge Odb]]]$

Apply US to Step 9, and substitute d for y. Apply the simplification rule. Then

(10) $Cd \wedge H_3add \wedge Oag \rightarrow H_2dd$
(11) $Cd \wedge H_3add \wedge Oab \rightarrow H_2dd \wedge Odb$

Apply US to Step 3, and substitute d for y and z. Next, use the simplification rule. Then

(12) $Cd \wedge H_2dd \rightarrow H_3add \wedge Oag$
(13) $Cd \wedge \neg H_2dd\text{–} \rightarrow H_3add \wedge Oab$

We know that $F_1 \wedge F_2 \rightarrow F_3 = F_1 \rightarrow [F_2 \rightarrow F_3]$. Apply the technique to Steps 10–13. Next, apply MP to the results and Step 8. Then

(14) $H_3add \wedge Oag \rightarrow \neg H_2dd$
(15) $H_3add \wedge Oab \rightarrow H_2dd \wedge Odb$
(16) $H_2dd \rightarrow H_3add \wedge Oag$
(17) $\neg H_2dd \rightarrow H_3add \wedge Oab$

Apply implication transitivity to Steps 16 and 14, 17 and 15, respectively. Then

(18) $H_2dd \rightarrow \neg H_2dd$
(19) $\neg H_2dd \rightarrow H_2dd \wedge Odb$

Then

(20) $\neg H_2dd \rightarrow H_2dd$

From Steps 18 and 20 we can conclude that there is a computer program d such that d halts on given input d itself iff d does not halt on given input d itself. This program is self-contradictory; cf. Minsky [5].

Unfortunately, so far ANDP cannot draw the conclusion from the new formulation of the halting problem in natural deduction style.

Acknowledgment: We thank Prof. Zhang Ming-Hua for helpful discussions about our 43-step proof.

Proof. The proof of the formulation of the halting problem in [2] in natural deduction style is as follows.

Let P_1, P_2, P_3, and P_4 represent the four premises in [2]. Let $\neg A_1$ represent the conclusion: $\neg(\exists x)[Ax \wedge (\forall y)[Cy \rightarrow (\forall z)Dxyz)]$.

```
1.  P1 & P2 & P3 & P4                                        ASSUMED PREMISE
2.  (Ex)[Ax & (Ay)[Cy -> (Az)Dxyz)]
  -> (Ew)[Cw & (Ay)[Cy -> (Az)Dwyz]]                                 SIMP 1
3.  (Aw)[Cw & (Au)[Cu -> (Av)Dwuv]
 -> (Ay)(Az)[[Cy & Pyz -> Qwyz & Owg] &
            [Cy & ~Pyz -> Qwyy & Owb]]]                              SIMP 1
4.  (Ew)[Cw & (Ay)[[Cy & Pyy -> Qwyy & Owg] &
                   [Cy & ~Pyy -> Qwyy & Owb]]]
 -> (Ev)[Cv & (Ay)[[Cy & Pyy -> Pvy & Ovg] &
                   [Cy & ~Pyy -> Pvy & Ovb]]]                        SIMP 1
5.  (Ev)[Cv & (Ay)[[Cy & Pyy -> Pvy & Ovg] &
                   [Cy & ~Pyy -> Pvy & Ovb]]
 -> (Eu)[Cu & (Ay)[[Cy & Pyy -> ~Puy] &
                   [Cy & ~Pyy -> Puy & Oub]]]                        SIMP 1
6.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                              CASE2  2
7.  (Ew)[Cw & (Ay)[Cy -> (Az)Dwyz]]                               CASE1  2
8.  Cal & (Ay)[Cy -> (Az)Dalyz]                                    HYPO  7
9.  Cal                                                            SIMP  8
10, (Ay)[Cy -> (Az)Dalyz]                                          SIMP  8
11. Cal & (Au)[Cu -> (Av)Daluv]
 -> (Ay)(Az)[[Cy & Pyz -> Qalyz & Oalg] &
             [Cy & ~Pyz -> Qalyz & Oalb]]                    US (al w)  3
12.  ~Cal v [~(Au)[Cu -> (Av)Daluv] v
 (Ay)(Az)[[Cy & Pyz -> Qalyz & Oalg] &
          [Cy & ~Pyz -> Qalyz & Oalb]]]                    IMPLICATION  11
13.  ~(Au)[Cu -> (Av)Daluv] v
 (Ay)(Az)[[Cy & Pyz -> Qalyz & Oalg] &
          [Cy & ~Pyz -> Qalyz & Oalb]]                          LDS 9  12
14.  (Ay)(Az)[[Cy & Pyz -> Qalyz & Oalg] &
              [Cy & ~Pyz -> Qalyz & Oalb]]                     LDS 13  10
15.  ~(Ew)[Cw & (Ay)[[Cy & Pyy -> Qwyy & Owg] &
                     [Cy & ~Pyy -> Qwyy & Owb]]]                 CASE2  4
16.  (Ev)[Cv & (Ay)[[Cy & Pyy -> Pvy & Ovg] &
                    [Cy & ~Pyy -> Pvy & Ovb]]]                   CASE1  4
17.  (Eu)[Cu & (Ay)[[Cy & Pyy -> ~Puy] &
                    [Cy & ~Pyy -> Puy & Oub]]]                     MP  5 16
18.  Ca2 & (Ay)[[Cy & Pyy -> ~Pa2y] &
                [Cy & ~Pyy -> Pa2y & Oa2b]]                       HYPO  17
19.  Ca2                                                          SIMP  18
20.  (Ay)[[Cy & Pyy -> ~Pa2y] & [Cy & ~Pyy -> Pa2y & Oa2b]]
21.  [Ca2 & Pa2a2 -> ~Pa2a2] & [Ca2 & ~Pa2a2 -> Pa2a2 & Oa2b]
                                                            US (a2 y)  20
22.  Ca2 & Pa2a2 -> ~Pa2a2                                        SIMP  21
```

```
23.  Ca2 & ~Pa2a2 -> Pa2a2 & Oa2b                                  SIMP  21
24.  ~Ca2 v [Pa2a2 v Pa2a2 & Oa2b]                            IMPLICATION  23
25.  Pa2a2 v Pa2a2 & Oa2b                                        LDS  24  19
26.  ~Ca2 v [~Pa2a2 v ~Pa2a2]                                 IMPLICATION  22
27.  ~Pa2a2                                                      LDS  26  19
28.  Pa2a2 & Oa2b                                                LDS  25  27
29.  Pa2a2                                                      SIMP  28
30.  ~[Cal & (Ay)[[Cy & Pyy -> Qalyy & Oalg] &
                  [Cy & ~Pyy -> Qalyy & Oalb]]]
                                                             US (al w)  15
31.  ~Cal v ~(Ay)[[Cy & Pyy -> Qalyy & Oalg] &
                  [Cy & ~Pyy -> Qalyy & Oalb]]
                                                            DE.MORGAN  30
32.  ~(Ay)[[Cy & Pyy -> Qalyy & Oalg] &
           [Cy & ~Pyy -> Qalyy & Oalb]]                          LDS  31  9
33.  ~[[Ca4 & Pa4a4 -> Qala4a4 & Oalg] &
       [Ca4 & ~Pa4a4 -> Qala4a4 & Oalb]]                        HYPO  32
34.  ~(Av8)(Av7)[[Cv8 & Pv8v7 -> Qalv8v7 & Oalg] &
                 [Cv8 & ~Pv8v7 -> Qalv8v7 & Oalb]]
                                                             EG&EG  33
35.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                  ~-ELIMINATION 14 14
36.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                  ~-ELIMINATION 29 27
37.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                          EE  17  36
38.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                          EE  32  35
39.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                      CASES  4 37 38
40.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                          EE  7  39
41.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                        SAME  6
42.  ~(Ex)[Ax & (Ay)[Cy -> (Az)Dxyz]]                      CASES  2 40 41
43.  P1 & P2 & P3 & P4 -> ~A1                                 CP 42
```

Appendix

L. Burkholder presented the halting problem in [3] as the 76th automated theorem proving problem. This is an important theorem in computer science. We use $(\exists x)$, $(\forall x)$ to stand for existential quantifier and universal quantifier, respectively. Details about the English statement for the halting problem are given in [1,2], from which we take the following formulas. The key is as follows:

Ax	x is an algorithm
Cx	x is a computer program in some programming language
$Dxyz$	x is able to decide whether y halts, given input z
H_2xy	x halts on given input y
H_3xyz	x halts on given as input the pair $< y, z >$
Oxy	x outputs y

Four premises are given. The first premise says that if an algorithm that solves the halting problem exists, then it can be written as a computer program in some language.

P_1 (for premise 1):

$$(\exists x)[Ax \wedge (\forall y)[Cy \rightarrow (\forall z)Dxyz]] \rightarrow (\exists w)[Cw \wedge (\forall y)[Cy \rightarrow (\forall z)Dwyz]]$$

The second premise states what is meant when a program w is able to decide whether a program y halts or does not halt on input z. Program w does the following: If program y halts given input z, then program w halts given as input the pair $< y, z >$ and prints out "g" (for good). On the other hand, if program y does not halt given input z but goes on forever, then program w halts on $< y, z >$ and prints out "b" (for bad).

P_2 (for premise 2):

$$(\forall w)[[Cw \wedge (\forall u)[Cu \rightarrow (\forall v)Dwuv]] \rightarrow \\ (\forall y)(\forall z)[[[Cy \wedge H_2yz] \rightarrow [H_3wyz \wedge Owg]] \wedge [[Cy \wedge \neg H_2yz] \rightarrow [H_3wyz \wedge Owb]]]]$$

Now, program y may be given itself as input. The third premise says that if program w decides the termination of a program y on y as an input, then there exists a slightly different program v that can determine only whether a program halts on itself.

P_3 (for premise 3):

$$(\exists w)[Cw \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow [H_3wyy \wedge Owg]] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_3wyy \wedge Owb]]]] \rightarrow \\ (\exists v)[Cv \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow [H_2vy \wedge Odg]] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_2vy \wedge Odb]]]]$$

The final premise states that if a program such as v exists, then so does a yet slightly different program that goes into a loop just in those cases when program v would halt and print "g".

P_4 (for premise 4):

$$(\exists v)[Cv \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow [H_2vy \wedge Odg]] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_2vy \wedge Odb]]]] \rightarrow \\ (\exists u)[Cu \wedge (\forall y)[[[Cy \wedge H_2yy] \rightarrow \neg H_2uy] \wedge [[Cy \wedge \neg H_2yy] \rightarrow [H_2uy \wedge Oub]]]]$$

The conclusion is that an algorithm to solve the halting problem does not exist:

$$: \neg(\exists x)[Ax \wedge (\forall y)[Cy \rightarrow (\forall z)Dxyz]]$$

References

1. Bruschi, M., The halting problem, *AAR Newsletter* 17, March 1991.

2. Burkholder, L., The halting problem, *SICACT News* 18, no. 3, spring 1987.

3. Burkholder, L., A 76th automated theorem-proving problem, *AAR Newsletter* 8, April 1987.

4. Li Dafa, A mechanical proof of the halting problem, *AAR Newsletter* 23, June 1993.

5. Minsky, M. L., *Computation: Finite and Infinite Machines.* Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

A Test Problem for Finite Model Generators

Jian Zhang
University of Iowa
jzhang@cs.uiowa.edu

In the theory, there are three binary operators: m (multiplication), ld (left division), and rd (right division). The axioms are

$$rd(m(x,y),y) = x$$
$$m(rd(x,y),y) = x$$
$$ld(y,m(y,x)) = x$$
$$m(y,ld(y,x)) = x$$
$$rd(x,x) = ld(y,y)$$
$$m(w,m(m(z,x),z)) = m(m(m(w,z),x),z)$$
$$m(rd(m(m(z,x),y),m(x,y)),y) = rd(m(z,m(y,x)),x)$$
$$m(m(y,x),m(y,x)) = m(m(m(x,y),y),x)$$
$$m(rd(m(m(w,y),x),m(y,x)),rd(m(m(z,y),x),m(y,x))) = rd(m(m(m(w,z),y),x),m(y,x))$$

All variables are universally quantified. The problem is to find finite models of this theory that are not Abelian groups.

With my program FALCON, I found one such model of cardinality 8 and another of cardinality 16.

Two Challenge Problems

Larry Wos
Argonne National Laboratory, Argonne, IL 60439
e-mail: wos@mcs.anl.gov

For the two challenge problems we offer, we turn to Robbins algebra whose axioms are the following three, where one can interpret the function n as complement and the function + as union.

```
EQ(+(x,y),+(y,x)).    %  commutativity
EQ(+(+(x,y),z),+(x,+(y,z))).    %  associativity
EQ(n(+(n(+(x,y)),n(+(x,n(y))))),x).    %  Robbins axiom
```

Although it is known that every finite Robbins algebra is in fact a Boolean algebra, still open is the question of whether Robbins implies Boolean in all cases. Thanks to S. Winker [2,3], we know that the adjunction to the three axioms for a Robbins algebra of any one of a number of well-known properties of a Boolean algebra suffices to produce a Boolean algebra. For example, easily proved without induction or AC-unification, the adjunction of the property asserting that $c + c = c$ for some constant c suffices.

For the first challenge problem, we ask for a proof that the adjunction of the property asserting that $c+d = c$ suffices, where one is required to avoid the use of both induction and AC-unification. We have in fact found such a proof with McCune's program OTTER [1] running on a SPARC-10, in approximately 9770 CPU-seconds, of length 78 and level 16, with retention of clause (48308). We find that important advances sometimes occur by imposing some seemingly arbitrary constraint (such as blocking the use of AC-unification) on the program's attack and then attempting to formulate a method that nevertheless finds a proof.

For the second challenge problem, we note that the adjunction of $n(n(x)) = x$ suffices to prove Boolean; the theorem is not difficult to prove. However, still open is the question of whether the adjunction of $n(n(c)) = c$ (for a constant c) suffices.

References

1. McCune, W. W., "OTTER 3.0 Reference Manual and Guide," Technical Report ANL-94/6, Argonne National Laboratory, Argonne, Illinois, 1994

2. Winker, S., "Absorption and idempotency criteria for a problem in near-Boolean algebras," *J. Algebra* 153, no. 2 (December 1992) 414–423

3. Winker, S., "Robbins algebra: Conditions that make a near-Boolean algebra Boolean," *J. Automated Reasoning* 6, no. 4 (December 1990) 465–489

Call for Papers

Fourth Workshop on Theorem Proving with Analytic Tableaux and Related Methods

The Fourth Workshop on "Theorem Proving with Analytic Tableaux and Related Methods" will be held on May 7–10, 1995, in St. Goar am Rhein, Germany.

The workshop will bring together people who develop tableaux-like calculi for classical and nonclassical logic and people who are involved in the practical implementation of tableaux-like calculi. Topics of interest include analytic tableaux, model elimination, connection method, and sequent calculi.

Authors are invited to submit papers in one of the following categories: (1) original research papers of at most 15 pages or (2) position papers or work in progress, not necessarily original, with up to 5 pages. Authors should send four copies, preferably in LaTeX. Electronic or fax submissions are not acceptable. Deadline for submissions is November 22, 1994. It is intended to publish all accepted papers from category 1 in the *LNAI* series of Springer.

For further information, contact Reiner Hähnle, Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, 76128 Karlsruhe, Germany; fax: ++49-721-43 29; phone: ++49-721-608-39 19; e-mail: reiner@ira.uka.de.

Third Logic Programming and Nonmonotonic Reasoning Conference

The Third Conference on "Logic Programming and Nonmonotonic Reasoning" will take place June 26–28, 1995, in Lexington, Kentucky. Topics include the semantics for logic programs, default logic and its versions, modal nonmonotonic logics, nonmonotonic rule systems, abduction, nonmonotonic reasoning in databases, theory of updates and belief revision, constraint satisfaction, algorithms and complexity, implementations, and applications. Submissions (four copies, double-spaced, 12-point font) of a full paper of twenty pages or less should be sent to the program chair: Anil Nerode, Mathematical Sciences Institute, Cornell University, 407 College Ave., Ithaca, NY 14850; e-mail: nerode@msiadmin.cit.cornell.edu; tel: +1-607-255-7752; fax: +1-607-255-9003.

LICS

The Tenth Annual IEEE Symposium on Logic in Computer Science will be held on June 26–29, 1995, in San Diego, California. The symposium aims to attract original papers of high quality on theoretical and practical topics in computer science that relate to logic in a broad sense, including algebraic, categorical, and topological approaches.

Suggested topics include automated deduction, constraint programming, logics of knowledge, lambda and combinatory calculi, linear logic, logical aspects of computational complexity, logics in artificial intelligence, logic programming, modal and temporal logics, rewriting, logical aspects of symbolic computing, and verification.

Authors should send twelve copies of an extended abstract to the program chair to be received by December 7, 1994. For further information, contact Moshe Y. Vardi, Department of Computer Science, Rice University, Houston, Texas 77251-1892; vardi@cs.rice.edu.

International KRUSE Symposium

An international symposium on "Knowledge Retrieval, Use, and Storage for Efficiency" will take place at the University of California, Santa Cruz, August 11–13, 1995.

The symposium will provide a forum for exploring current research in artificial intelligence, cognitive science, and databases that pertains to the organization, encoding and retrieval of logical and complex objects. The symposium will draw together researchers from diverse disciplines as well as practitioners engaged in developing real object-oriented term classification systems. Mathematical and graph-theoretic approaches will be favored over those approaches based on analogy with human cognitive processes, though mathematical discussions of such processes will be appropriate. The basic questions to be addressed include

- classification of objects in a taxonomy: systemic classification, semantic indexing, partial-order sorting, description identification, and taxonomy maintenance.
- efficient order, lattice, graph, and code theoretic operations on objects: subsumption, generalization, specialization, least common generalization, and greatest common specialization.
- advanced uses of taxonomies: knowledge compression, knowledge compilation, and knowledge evolution.
- using classified knowledge: classification as problem solving, classification as constraint satisfaction, and exploiting abstraction.
- scalable techniques for large object databases
- integration of data and knowledge base technologies

The symposium will maintain a balance between theoretical issues and descriptions of implemented systems providing a balance between theory and practice. The focus of the symposium is on efficiency of retrieval, use, and storage.

Manuscripts up to 15 pages may be submitted; shorter, substantive papers are welcome. Authors are requested to submit five copies of their paper or to send postscript output electronically. Papers must be postmarked on or before February 13, 1995. For further information, contact KRUSE, c/o Gerard Ellis, Computer Science Dept., RMIT, GPO Box 2476V, Melbourne, VIC 3001, Australia; e-mail: ged@cs.rmit.edu.au ph:61-3-660-5090 fax:61-3-662-1617.

International Congress of Logic, Methodology and Philosophy of Science

The Tenth International Congress of "Logic, Methodology and Philosophy of Science" will take place on August 19–25, 1995, in Florence, Italy. Of especial interest to AAR members are the sections on proof theory and categorical logic; model theory, set theory and formal systems; recursion theory and constructivism; logic and computer science; logic, mathematics, and computer science; and cognitive sciences and artificial intelligence. Abstracts of contributed papers (max. 2 pages) should be submitted by February 1, 1995. For further information contact LMPS,

Centro Servizi di Segreteria, via A. Lapini, 1, I-50136 Florence, Italy; phone: ++39 (0)55 670369; fax: ++39 (0)55 660236.

CONCUR '95

The Sixth International Conference on Concurrency Theory will be held in Philadelphia, Pennsylvania, on August 21–24, 1995.

Submissions are invited in all areas of semantics, logics, and verification techniques for concurrent systems. Potential topics include process algebras, Petri nets, true concurrency, shared-memory and message-passing formalisms, operational and denotational models, programming language semantics, probabilistic and real-time processes, hybrid systems, concurrent logic and constraint programming, fairness, temporal logics, compositional analysis techniques, and verification tools.

Uuencoded dvi or postscript files should be e-mailed to concur95-submit@cs.sunysb.edu by March 1, 1995. When electronic submission is not possible, five hardcopies of the paper should be sent to CONCUR '95, Attn: Scott Smolka, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794-4400; telephone: +1 516 632 8453; fax: +1 516 632 8334. Submissions should contain a draft of a full paper of no more than fifteen typed pages, accompanied by a one-page abstract.

KARP - 95

The Second International Symposium on "Knowledge Acquisition, Representation and Processing" (KARP-95) will take place on September 27–30, 1995, at the Auburn University Conference Center in Auburn, Alabama.

KARP is an international forum for presentation and discussion of interdisciplinary research on knowledge acquisition, representation, and processing, as well as recent advances on discipline independent tools for realizing large-scale applications of knowledge-based systems. The purpose of the symposium is to identify challenging problems common to many disciplines that must be solved to realize future knowledge and information systems, and to shape future directions of research by soliciting and reviewing high-quality applied and theoretical research findings. An important part of the symposium is the provision for one-on-one interactions provided by an intimate setting, poster sessions, and demonstrations of operational systems.

The topics of interest include artificial intelligence—knowledge acquisition approaches and tools, knowledge representation, integration of heterogeneous knowledge representations, knowledge bases and models, search and planning approaches, inference methodologies, and interdisciplinary artificial intelligence applications.

Authors are invited to submit extended abstracts by December 1, 1994, to the Program Chair Chuck Karr, U.S. Bureau of Mines, The University of Alabama Campus, P.O. Box L, Tuscaloosa, AL 35486-9777; tel: +1 (205) 759-9432; fax: +1 (205) 759-9440; e-mail: karr@ai.usbm.gov. Six paper copies or one e-mail copy should be submitted. More information on KARP-95 can be obtained automatically by sending e-mail to karp-info@eng.auburn.edu.