ASSOCIATION FOR AUTOMATED REASONING
NEWSLETTER

No. 31 December 1995

From the AAR President, Larry Wos...

This issue includes several articles on new techniques and new attacks on long-standing problems. I find it most satisfying to end this year with one of our largest AAR newsletters.

Many of our readers may be familiar with the special series on automated reasoning that is published by Kluwer. Included in this *AAR Newsletter* is an announcement of a forthcoming book (with a description of its contents) in this series. The cited book, entitled *Piton: A Mechanically Verified Assembly-Level Language*, will appeal to a wide audience. Kluwer is contemplating offering AAR members a special discount for the book. If you are interested, contact our AAR secretary, Robert Veroff, at veroff@cs.unm.edu.

Another Crack in a Tough Nut

William McCune, Argonne National Laboratory
(mccune@mcs.anl.gov)

John McCarthy's famous memo "A Tough Nut for Proof Procedures" [1,2] contains a challenge to automatically prove that a checkerboard (8×8), with two squares in opposite corners removed, cannot be covered with dominoes. He gives a set of first-order predicate logic sentences stating that the mutilated board can be covered; the challenge is to show the set unsatisfiable.

These days, most interest in the puzzle is for $2n \times 2n, n > 0$ boards and in automatic discovery of the parity argument; several human and human-guided-mechanical proofs have been reported (see [2] for a summary). Also, the 8×8 theorem has been encoded directly into propositional logic and proved by several propositional provers (see [2]).

Our proof is for the 8×8 board and is essentially a propositional proof. We use MACE [3], a program that searches for finite models of first-order statements. To show McCarthy's set unsatisfiable, we do a bit of meta-reasoning, then MACE automatically does the main part of the proof. The meta-reasoning arises because MACE shows that the set has no model of size 8 rather than showing that it has no model of any size.

Here is the MACE input we use.

```
list(usable).
S(0,1). S(1,2). S(2,3). S(3,4). S(4,5). S(5,6). S(6,7). % 1
-S(x,y) | x < y.                                         % 2
```

```
-(x < y) | -(y < z) | -S(x,z).                              % 3b
G1(x,y) | G2(x,y) | G3(x,y) | G4(x,y) | G5(x,y).          % 6
-G1(x,y) | -G2(x,y).  % 7
-G1(x,y) | -G3(x,y).  % 7
-G1(x,y) | -G4(x,y).  % 7
-G1(x,y) | -G5(x,y).  % 7
-G2(x,y) | -G3(x,y).  % 8
-G2(x,y) | -G4(x,y).  % 8
-G2(x,y) | -G5(x,y).  % 8
-G3(x,y) | -G4(x,y).  % 9
-G3(x,y) | -G5(x,y).  % 9
-G4(x,y) | -G5(x,y).  % 10
-S(x1,x2) | -G1(x1,y) |  G3(x2,y).       % 13
-S(x1,x2) |  G1(x1,y) | -G3(x2,y).       % 13
-S(y1,y2) | -G2(x,y1) |  G4(x,y2).       % 14
-S(y1,y2) |  G2(x,y1) | -G4(x,y2).       % 14
-G3(0,y). -G1(7,y). -G2(x,7). -G4(x,0).  % 15
end_of_list.

formula_list(usable).
all x all y (G5(x,y) <-> (x=0 & y=0) | (x=7 & y=7)).  % 11, 12
end_of_list.
```

(The comments refer to the numbering in McCarthy's memo; we don't need axioms 3a, 4, and 5, because "=" and "<" are built in.) The intended domain is 0–7, and the squares are indexed (row,column). `S` is the successor relation, `G1(x,y)` means that squares (x, y) and $(x + 1, y)$ are are covered by a domino; similarly, `G2(x,y)` is for (x, y) and $(x, y + 1)$, `G3(x,y)` is for (x, y) and $(x - 1, y)$, and `G4(x,y)` is for (x, y) and $(x, y - 1)$. `G5(x,y)` means that (x, y) is not covered.

The constants 0–7 must be distinct by axioms 1, 2, and the relations "=" and "<". Therefore, any model of the set must have size at least 8. In fact, MACE assumes that 0–7 are elements of the domain, and thus are distinct.

Also it can be shown that if there is a model, then there is a model of size at most 8. Or, to simplify things, we can assume the statement

$$\forall x \ (x = 0 \mid x = 1 \mid x = 2 \mid x = 3 \mid x = 4 \mid x = 5 \mid x = 6 \mid x = 7).$$

Then it is clear that all models have size 8.

The meta-reasoning aside, MACE shows (in a minute or two) that the above set has no model of size 8, and the proof is complete. (Of course, this is one of those questionable proofs by exhaustive search; I estimate that a corresponding propositional resolution proof has more than 5 million of steps.) MACE works by reducing, for a given domain size, the first order statement into a propositional statement, then a propositional decision procedure checks for unsatisfiability.

Although there is nothing really new here, the proof I offer seems to be closest yet to answering the challenge precisely as put forth in McCarthy's memo.

MACE is freely available for most UNIX systems [3]. Thanks to Peter Andrews for reminding me of this problem.

References

1. J. McCarthy, "A Tough Nut for Proof Procedures", Memo No. 16, Stanford Artificial Intelligence Project, 1964.
2. J. McCarthy, `http://www-formal.stanford.edu/jmc/nut.html`, 1995.
3. W. McCune, MACE, `http://www.mcs.anl.gov/home/mccune/ar/mace/`, 1994.

The TPTP Problem Library, Release v1.2.0

Geoff Sutcliffe
Department of Computer Science, James Cook University, Australia
geoff@cs.jcu.edu.au
and
Christian Suttner
Institut fuer Informatik, TU Muenchen, Germany
suttner@informatik.tu-muenchen.de

The TPTP (Thousands of Problems for Theorem Provers) Problem Library is a library of test problems for automated-theorem proving (ATP) systems, using the clause normal form of first-order logic. The TPTP supplies the ATP community with the following:

- A comprehensive library of the ATP test problems that are available today, in order to provide an overview and a simple, unambiguous reference mechanism.
- A comprehensive list of references and other interesting information for each problem.
- New generalized variants of problems whose original presentation is hand-tailored towards a particular automated proof.
- Arbitrary-size instances of generic problems (e.g., the pigeon-holes problem).
- A utility to convert the problems to existing ATP formats. Currently the 3TAP, KIF, leanTAP, METEOR, MGTP, OTTER, PTTP, SETHEO, and SPRFN formats are supported, and the utility can easily be extended to produce any format required.
- General guidelines outlining the requirements for ATP system evaluation.

Release v1.2.0 of the TPTP is now available. It contains 2,758 problems in 25 domains. Here's what's new in v1.2.0 (after v1.1.3):

- 267 new problems, in the domains BOO COL GRP MSC PUZ ROB SYN.
- 49 bug fixes done, in the domains COL LCL PUZ ROB SYN.
- Generic problems (e.g., the N-queens problem) are now handled by problem generators, which allow the automatic generation of any desired problem size.
- The % syntax field has been reordered in all files.
- The tptp2X utility has been extended and improved in various ways:
  - Installation and use of tptp2X have been simplified.
  - tptp2X now updates the % Syntax field after transformations.
  - There are three new output formats: 3TAP, KIF, and leanTAP.
  - Problem generation has been integrated into tptp2X.
  - The syntax for specifying equality axiom removal has changed.
  - The syntax for specifying OTTER format output has changed.
  - A new syntax conversion option for the SETHEO system has been added.
- The TPTP technical report has been substantially revised.

The TPTP is regularly updated with new problems, additional information, and enhanced utilities. To register as a TPTP user, or to receive information on obtaining TPTP by anonymous ftp, please contact one of the following: Geoff Sutcliffe, geoff@cs.jcu.edu.au (Fax: +61-77-814029), or Christian Suttner, suttner@informatik.tu-muenchen.de (Fax: +49-89-526502).

Implementing Subsumption in Prolog

Geoff Sutcliffe, Department of Computer Science, James Cook University
geoff@cs.jcu.edu.au

Prolog is a convenient language in which to implement ATP systems for first-order logic because the formulae to be manipulated can be represented directly as Prolog terms. In the logic programming community there is some debate as to the desirability of using Prolog variables to represent the logic variables in that data, but doing so does make life easy in many respects. In particular, clauses are easily represented as lists of Prolog terms, using some prefix operators to indicate the signs and Prolog terms for the atoms of the literals. For example, in the TPTP problem library a sample clause (from PUZ001-1.p) is

```
[--lives(X),++richer(X,agatha),++hates(butler,X)]
```

Given an ATP system that represents clauses this way, subsumption between clauses is (always?) required for acceptable performance. I have played with various Prolog implementations of subsumption. Below is the most elegant version I have come up with (the most elegant is not necessarily the fastest, but Prolog programmers know that "elegance is not optional"). I would be interested to hear of anything neater.

```
%-----------------------------------------------------------------------
%----Check every member of the first list is a member of the second
all_members([],_).

all_members([H|T],L):-
    member(H,L),
    all_members(T,L).
%-----------------------------------------------------------------------
%----Check if a list of literals subsumed another
literals_subsume(SubsumingLiterals,SubsumedLiterals):-
%----Enforce theta rule for length, if you want
%   length(SubsumingLiterals,SubsumingLength),
%   length(SubsumedLiterals,SubsumedLength),
%   SubsumedLength >= SubsumingLength,
%----Verify to avoid instantiating variables
    \+ \+ (
        numbervars(SubsumedLiterals,0,_),
        all_members(SubsumingLiterals,SubsumedLiterals)).
%-----------------------------------------------------------------------
```

Call for Papers

RTA-96

The Seventh Conference on Rewriting Techniques and Applications will take place on July 27–30, 1996, at Rutgers University, New Jersey. Papers are solicited in any of the following or related areas:

Term rewriting systems	Symbolic and algebraic computation
Constrained rewriting and deduction	Equational programming languages
String and graph rewriting	Completion techniques
Rewrite-based theorem proving	Unification and matching algorithms
Conditional and typed rewriting	Constraint solving
Higher-order rewriting	Architectures for rewriting

In addition to full research papers (15 pages), descriptions of new working systems (4 proceedings pages) and problem sets that provide realistic, interesting challenges in the field of rewriting techniques are also welcome. Papers on new applications of rewriting techniques are particularly encouraged. Submissions must reach the program chair no later than January 15, 1996: Harald Ganzinger, RTA96, Max Planck Institute for Computer Science, Im Stadtwald, D-66123 Saarbruecken, Germany; e-mail: rta96@mpi-sb.mpg.de; fax: +49 681 302-5401; telephone: +49 681 302-5360.

CADE-13

The Thirteenth International Conference on Automated Deduction will be held at Rutgers University, in New Brunswick, New Jersey, on July 30–August 3, 1996.

The CADE conferences are the major forum for the presentation of new research in all aspects of automated deduction. Original research papers, descriptions of working reasoning systems, and problem sets that provide innovative, challenging tests for automated reasoning systems, are solicited.

CADE conferences cover all aspects of automated deduction, including first vs. higher-order logics, classical vs. non-classical logics, special vs. general-purpose inference, and interactive vs. automatic systems. Specific topics of interest include resolution, sequent calculus, decision procedures, unification, rewrite rules, and mathematical induction. Also of interest are any applications of automated deduction, including deductive databases, logic and functional programming, commonsense reasoning, and software and hardware development. Papers on commercial or industrial applications of automated deduction are especially encouraged.

CADE-13 will be held from Tuesday, July 30, to Saturday, August 3. It will be held as part of the Federated Logic Conference (FLoC'96) to be held at Rutgers University, New Brunswick, New Jersey, USA, from Saturday, July 27, to Saturday, August 3. As well as CADE, other conferences participating in FLoC'96 will be CAV (Conference on Computer-Aided Verification), LICS (IEEE Symposium on Logic in Computer Science), and RTA (Conference on Rewriting Techniques and Applications). The goal of FLoC is to battle fragmentation of the technical community by bringing together synergetic conferences that relate logic to computer science.

The Proceedings of CADE-13 will be published by Springer-Verlag in their Lecture Notes in Artificial Intelligence Series. Research papers should not exceed 15 (fifteen) proceedings pages. System descriptions and problem sets should not exceed 5 (five) proceedings pages. Springer style files should be used if possible. These can be obtained early September from http://www.research.att.com/lics/FLoC.

Submission deadline is January 12, 1996. Authors should send four copies of their submission to the program co-chairs. Further information about the conference may be obtained from the CADE-13 World Wide Web site: http://www.research.att.com/lics/FLoC.

Program Co-Chairs: Michael McRobbie and John Slaney, Centre for Information Science Research, The Australian National University, ACT 0200, Australia Tel: [+61] 6-2492035, Fax: [+61] 6-2490747, e-mail: cade13@cisr.anu.edu.au.

GOEDEL'96

The GOEDEL'96 conference will be held on August 25–29, 1996, in Brno, Czech Republic (birthplace of Kurt Goedel). The aim of the conference is to hold tribute to Kurt Goedel by arranging a scientific event presenting a forum for papers relevant to foundational aspects of logic in mathematics, computer science, philosophy, and physics—areas influenced by Kurt Goedel's work. Both original scientific papers are sought for the conference, as well as research work in history connected with Goedel's work. For further information, see the World Wide Web

http://www.fi.muni.cz/ zlatuska/goedel96.html.

Announcing the Availability of More Nqthm-Checked Theorems

Bob Boyer and J Moore

In addition to the theorems in the "examples" directory distributed with Nqthm-1992. the following Nqthm-checked theorems are available:

- Much of the 'Clinc Stack'
  - The FM9001 microprocessor (Brock & Hunt, with contributions from Kaufmann) [fm9001-piton/fm9001-replay.events]
  - The Piton assembler (Moore) [fm9001-piton/piton.events]
  - The "big-add" Piton example (Moore) [fm9001-piton/big-add.events]
  - A Piton program that wins at Nim (Wilding) [fm9001-piton/nim-piton.events]
- A Paris-Harrington Ramsey theorem (Kunen) [kunen/paris-harrington.events]
- An illustration of the surprising power of EVAL$ (Kunen) [kunen/induct.events] (surprising to Boyer and Moore anyway)
- The arithmetic-geometric mean theorem (Kaufmann & Pecchiari) [numbers/arithmetic-geometric-mean.events]
- The mutilated checkerboard theorem in the general $N \times N$ case (Subramanian) [subramanian/mutilated-checkerboard.events]
- A simple real-time system, the classic train example (Young) [young/train.events]
- A theorem about coin tossing probabilities (Kaufmann) [numbers/tossing.events]
- A proof of correctness of a real-time scheduling algorithm (Wilding) [numbers/scheduler.events]

Some documentation for some of the above proof efforts may be found as follows:

FM9001 microprocessor	http://www.cli.com/hardware/fm9001.html
Piton assembler	http://www.cli.com/reports/files/022.ps
Nim playing program in Piton	http://www.cli.com/reports/files/078.ps
Paris-Harrington Ramsey	http://www.cs.wisc.edu/ kunen/ramsey.ps
EVAL$	http://www.cs.wisc.edu/ kunen/nqthm.ps
Arithmetic-geometric mean	http://www.cli.com/reports/files/100.ps
Real-time train	http://www.cli.com/reports/files/093.ps
Mutilated checkerboard	ftp://ftp.cli.com/pub/nqthm/nqthm-1992/ examples/subramanian/mutilated-checkerboard.ps
Real-time scheduling	ftp://ftp.cli.com/home/wilding/scheduler-proof.ps

The source files for these theorems, named within square brackets above, may be obtained individually from the directory
ftp://ftp.cli.com/pub/nqthm/nqthm-1992/examples/
or altogether in the single file
ftp://ftp.cli.com/pub/nqthm/nqthm-1992/1995-examples.tar.Z.

Also included on the tar file are new "driver" files for doing a replay of all the examples under Nqthm-1992, both these new examples and those previously distributed with Nqthm-1992. A Gnu Emacs TAGS file for all the event commands in all the examples is also provided.

For information on obtaining the Nqthm prover itself, see
ftp://ftp.cli.com/pub/nqthm/nqthm-1992/nqthm-1992.announcement.

An Erratum for Some Errata to Automated Theorem Proving Problems

Francis Jeffrey Pelletier
Department of Computing Science, University of Alberta,
Edmonton, Alberta, Canada T6G 2H1. e-mail jeffp@cs.ualberta.ca
Geoff Sutcliffe
Department of Computer Science, James Cook University,
Townsville, Queensland, Australia 4811. e-mail geoff@cs.jcu.edu.au

1 Introduction

In 1986 Pelletier [7] published an annotated list of logic problems, intended as an aid for students, developers, and researchers to test their automated theorem proving (ATP) systems. The 75 problems in the list are subdivided into propositional logic (Problems 1–17), monadic-predicate logic (Problems 18–34), full predicate logic without identity and functions (Problems 35–47), full predicate logic with identity but without functions (Problems 48–55), full predicate logic with identity and arbitrary functions (Problems 56–70), and problems to use in studying computational complexity of ATP systems (Problems 71–75). The problems were chosen partially for their historical interest and partially for their abilities to test different aspects of ATP systems. The problems were also assigned an intuitive "degree of difficulty", relativized to the type of problem. All the problems are presented in a "natural form" (which is here also called the "first-order form" or FOF), and most of them are also given in an equivalent negated conclusion clause normal form (CNF). The CNF versions of the problems are all in the TPTP Problem Library [12, 13], and are thus conveniently available to ATP system developers who use the CNF form.[1]

[1] The TPTP Problem Library can be obtained through the World Wide Web (WWW) URLs:
http://www.cs.jcu.edu.au/ftp/users/GSutcliffe/TPTP.HTML
http://wwwjessen.informatik.tu-muenchen.de/~suttner/tptp.html

2 The Erratum

Shortly after the publication of [8], Art Quaife and John Pollock sent Jeff Pelletier lists of errors occurring in the problems. These errata (and some others) were published in [9]. (Researchers thinking of using the problems should most definitely consult the Errata!) In particular, Problem 62 was "corrected". Since then a few researchers have written to Pelletier questioning the validity of some of the problems (usually not saying whether they were working with the natural FOF or the CNF), but he always maintained that they were valid, since they were provable with THINKER [6, 7], his natural deduction-based ATP system. So he said that these researchers either had an inferior ATP system or were not looking at the Errata. Recently, however, Geoff Alexander reported to Geoff Sutcliffe that he was unable to find a proof of the "corrected" CNF version of Problem 62. This report caused Geoff Sutcliffe to attempt, and fail, to prove it with OTTER [5]. Geoff Alexander then pointed out to Geoff Sutcliffe that the "corrected" CNF version of Problem 62 in [9] does not contain any negative clauses. Geoff used Geoff's insight to convince Jeff that not all was well with even the "corrected" version of Problem 62. Since the "corrected" natural FOF of Problem 62 is provable by THINKER (and by John Pollock's system, Oscar [10, 11]), but the CNF version of the problem is satisfiable, the obvious conclusion is that this CNF version was constructed incorrectly.

There are various reasons why it is not possible to establish exactly went wrong: Pelletier's original clausifier is no longer available, and other clausifiers (e.g., Sutcliffe's and OTTER's) apparently use algorithms different from Pelletier's and from each other, and generate different sets of clauses. In the end it was concluded that the CNF version of Problem 62 in the Errata was generated from the unnegated-conclusion of the natural form. As with the original errata, Pelletier again takes responsibility for this further erratum.

In discovering this erratum in the Errata, yet another flaw was discovered. The natural form of Problem 62 in [9] has an $\rightarrow$ (implication) as the main connective, whereas the original version in [8] has an $\leftrightarrow$ (equivalence). Further, Problem 62 is one of four variations on the same problem, and all the others (Problems 17, 33, 38) have an $\leftrightarrow$ as main connective in their natural forms. It is clearly intended that Problem 62 should also have an $\leftrightarrow$ as its main connective in the natural form. Thus both the CNF and FOF versions of Problem 62 in [9] are incorrect (although the FOF, unlike the CNF, is nonetheless still a theorem).

3 The Correction

The correct version of Problem 62 is (following the format of [8], + is disjunction, & is conjunction, u, v, w, x, y, and z are variables, (Ax) is a universally quantified x, (Ex) is an existentially quantified x, f is a function symbol, P is a predicate letter, a is a constant, and skN are Skolem constants generated in the clausification):

Problem 62

Natural FOF	**Negated Conclusion CNF**
$(Ax)[(Pa\&(Px \rightarrow Pf(x))) \rightarrow Pf(f(x))]$	Pa
$\leftrightarrow$	$Px + Pf(f(x)) + \neg Pa$
$(Ax)[(\neg Pa + Px + Pf(f(x)))\&$	$Pf(f(x)) + \neg Pa + \neg Pf(x)$
$(\neg Pa + \neg Pf(x) + Pf(f(x)))]$	$\neg Pf(f(sk1)) + \neg Pf(f(sk2))$
	$Pf(sk1) + Pf(sk2) + \neg Psk1 + \neg Psk2$
	$Pf(sk1) + \neg Psk1 + \neg Pf(f(sk2))$
	$Pf(sk2) + \neg Psk2 + \neg Pf(f(sk1))$

(The Negated Conclusion CNF is TPTP problem SYN084-2.p.)

4 The Related Problems

Recall that the four problems (Problems 17, 33, 38, 62) are variations on a theme. The theme was set in [2, p. 59]. In discussing their natural deduction system IMPLY, they say:

> IMPLY is incomplete in many ways. For example, although it can prove the skolemized formula
>
> $(P0\&(Px \rightarrow Pf(x)) \rightarrow Pf(f(x)))$
>
> it cannot handle the following equivalent formula
>
> $(\neg P0 + Px + Pf(f(x)))\&(\neg P0 + \neg Pf(x) + Pf(f(x)))$
>
> because the substitution [0/x] satisfying the first conclusion does not satisfy the second.

Now, it is difficult to know exactly what points the authors intend to make in this quotation, especially with their use of "can prove" and "cannot handle", but at least they seem to be saying that IMPLY would not be able to prove the equivalence of the two formulas. It is exactly this equivalence (adding explicit universal quantifiers and using the constant a rather than 0) that Problem 62 addresses.

The quote also says that the first formula is "skolemized". Again, it is difficult to know what the authors mean by this, since it is not possible to interpret the f's occurring in it as Skolem functions: the $f(f(x))$ would not be interpretable that way. However, it is possible to remove the functions by using equivalent formulas. For example, $Pf(x)$ can be replaced by

(1) $\exists y(Rxy\&\forall z(Rxz \rightarrow y = z)\&Py)$

where R is a new predicate not mentioned elsewhere in the problem. [8] does not present a version of Problem 62 using this substitution, but instead gives a 'weaker' version in which the uniqueness of the function is sacrificed by omitting the middle conjunct of (1). Rather than saying that the *unique* thing to which x is R-related is a P, the weaker version says instead that there is something to which x is R-related and that thing is a P. In this scheme, $Pf(x)$ is replaced by

(2) $\exists y(Rxy\&Py)$

It turns out that the effects of the uniqueness presuppositions of the function symbols in Problem 62 are the same on both sides of the equivalence sign in the natural form, so these functions can be uniformly replaced by relations in accordance with (2) and still yield a theorem. This was done in [8] in order to provide a relational predicate logic problem without identity and without function symbols. This is Problem 38 in [8], where only the natural form is given.

Problem 38 Natural FOF

$$(Ax)[(Pa\&(Px \rightarrow (Ey)(Py\&Rxy))) \rightarrow (Ez)(Ew)(Pz\&Rxw\&Rwz)]$$
$$\leftrightarrow$$
$$(Ax)[(\neg Pa + Px + (Ez)(Ew)(Pz\&Rxw\&Rwz))\&$$
$$(\neg Pa + \neg(Ey)(Py\&Rxy) + (Ez)(Ew)(Pz\&Rxw\&Rwz))]$$

The shortest Negated Conclusion CNF we know for Problem 38 contains 46 clauses, and is relegated to the Appendix below.

Further examination of the natural FOF of Problem 38 reveals that the logical contribution of the parts of this formula that say there is something to which x is R-related is the same on each side of the equivalence. Occurrences of these sub-formulas can therefore be uniformly replaced by constants, and the result will still be a theorem. Thus the same problem can be stated as a theorem in monadic predicate logic, which is Problem 33 in [8]:

Problem 33

Natural FOF	**Negated Conclusion CNF**
$(Ax)[Pa\&(Px \rightarrow Pb) \rightarrow Pc]$	Pa
$\leftrightarrow$	$\neg Pa + Px + Pc + Py$
$(Ax)[(\neg Pa + Px + Pc)\&$	$\neg Pa + \neg Pb + Pc$
$(\neg Pa + \neg Pb + Pc)]$	$\neg Pc$
	$Pb + \neg Pd + \neg Pe$
	$\neg Pa + Pb + Pc + Px + \neg Pd$
	$\neg Pa + Pb + Pc + Px + \neg Pe$

(The Negated Conclusion CNF is TPTP problem SYN063-1.p.)

A better clausification is produced by Sutcliffe's clausifier, which reduces the natural form to a trivial Negated Conclusion CNF:

Problem 33 Negated Conclusion CNF

$$Pa$$
$$Pc + \neg Pa$$
$$\neg Pc$$

(These clauses are TPTP problem SYN063-2.p.)

It is well known [1] (see 3, pp. 174–180] for an exposition) that a monadic logic formula is a theorem so long as it is not falsifiable with a domain of size 2^N, where N is the number of distinct predicates (constants being treated as predicates also). This means that monadic logic problems are really propositional logic problems. But *this* monadic logic problem is a particularly easy problem: it is (modulo the translation of constants in Problem 33 to predicates) a substitution instance of the following propositional logic theorem, which is Problem 17 in [8].

Problem 17

Natural FOF	**Negated Conclusion CNF**
$(p\&(q \rightarrow r) \rightarrow s]$	p
$\leftrightarrow$	$\neg p + q + s$
$(\neg p + q + s)$	$\neg p + \neg r + s$
$(\neg p + \neg r + s))$	$\neg s$
	$\neg q + r$

(The Negated Conclusion CNF is TPTP problem SYN047-1.p.)

It therefore seems that the IMPLY system of [2] is unable to solve a simple propositional equivalence.

5 Conclusion

It is interesting to note the varying degrees of difficulty that provers have with Problem 38 (i.e., the relational predicate logic problem without identity and without function symbols). The natural FOF version of Problem 38 is proved quite easily by THINKER. In contrast, OTTER is unable to find a proof of the natural FOF version (which OTTER clausifies to 55 clauses) or of the 46 clause Negated Conclusion CNF, after one hour of DEC station 5000 CPU time, i.e., using fairly large resources. SETHEO [4] is similarly unsuccessful with the 46 clause version. These observations confirm that an essentially easy FOF problem can become a difficult CNF problem.

A Negated Conclusion CNFs for Problem 38

Pa

$Px + Py + Psk1(y) + Psk3(x) + \neg Pa$

$Px + Py + Psk1(y) + Rx, sk4(x) + \neg Pa$

$Px + Py + Psk1(y) + Rsk4(x), sk3(x) + \neg Pa$

$Px + Py + Psk3(x) + Ry, sk2(y) + \neg Pa$

$Px + Py + Rx, sk4(x) + Ry, sk2(y) + \neg Pa$

$Px + Py + Ry, sk2(y) + Rsk4(x), sk3(K) + \neg Pa$

$Px + Psk1(x) + Psk5(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk1(x) + Ry, sk6(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk1(x) + Rsk6(y), sk5(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk5(y) + Rx, sk2(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Ry, sk6(y) + Rx, sk2(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Rx, sk2(x) + Rsk6(y), sk5(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Py + Psk3(x) + Rsk2(y), sk1(y) + \neg Pa$

$Px + Py + Rx, sk4(x) + Rsk2(y), sk1(y) + \neg Pa$

$Px + Py + Rsk2(y), sk1(y) + Rsk4(x), sk3(x) + \neg Pa$

$Px + Psk1(y) + Psk3(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk3(x) + Ry, sk2(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk1(y) + Rx, sk4(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Rx, sk4(x) + Ry, sk2(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk3(x) + Rsk2(y), sk1(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Rx, sk4(x) + Rsk2(y), sk1(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk1(y) + Rsk4(x), sk3(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Ry, sk2(y) + Rsk4(x), sk3(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Rsk2(y), sk1(y) + Rsk4(x), sk3(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Psk5(y) + Rsk2(x), sk1(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Ry, sk6(y) + Rsk2(x), sk1(x) + \neg Pz + \neg Pa + \neg Ry, z$

$Px + Rsk2(x), sk1(x) + Rsk6(y), sk5(y) + \neg Pz + \neg Pa + \neg Ry, z$

$Psk1(x) + Psk5(y) + \neg Pz + \neg Pu + \neg Pa + \neg Ry, z + \neg Rx, u$

$Psk5(x) + Ry, sk2(y) + \neg Pz + \neg Pu + \neg Pa + \neg Rx, z + \neg Ry, u$

$Psk1(x) + Ry, sk6(y) + \neg Pz + \neg Pu + \neg Pa + \neg Ry, z + \neg Rx, u$

$Rx, sk6(x) + Ry, sk2(y) + \neg Pz + \neg Pu + \neg Pa + \neg Rx, z + \neg Ry, u$

$Psk5(x) + Rsk2(y), sk1(y) + \neg Pz + \neg Pu + \neg Pa + \neg Rx, z + \neg Ry, u$

$Rx, sk6(x) + Rsk2(y), sk1(y) + \neg Pz + \neg Pu + \neg Pa + \neg Rx, z + \neg Ry, u$

$Psk1(x) + Rsk6(y), sk5(y) + \neg Pz + \neg Pu + \neg Pa + \neg Ry, z + \neg Rx, u$

$Rx, sk2(x) + Rsk6(y), sk5(y) + \neg Pz + \neg Pu + \neg Pa + \neg Ry, z + \neg Rx, u$

$Rsk2(x), sk1(x) + Rsk6(y), sk5(y) + \neg Pz + \neg Pu + \neg Pa + \neg Ry, z + \neg Rx, u$

$Psk10 + Psk8 + \neg Psk7 + \neg Psk9$

$Psk8 + Rsk9, sk10 + \neg Psk7 + \neg Psk9$

$Psk10 + Rsk7, sk8 + \neg Psk7 + \neg Psk9$

$Rsk7, sk8 + Rsk9, sk10 + \neg Psk7 + \neg Psk9$

$Psk8 + \neg Px + \neg Psk7 + \neg Ry, x + \neg Rsk9, y$

$Rsk7, sk8 + \neg Px + \neg Psk7 + \neg Ry, x + \neg Rsk9, y$

$Psk10 + \neg Px + \neg Psk9 + \neg Ry, x + \neg Rsk7, y$

$Rsk9, sk10 + \neg Px + \neg Psk9 + \neg Ry, x + \neg Rsk7, y$

$\neg Px + \neg Py + \neg Rz, x + \neg Ru, y + \neg Rsk7, u + \neg Rsk9, z$

(These clauses are TPTP problem SYN067-3.p.)

References

1. P. Bernays and M. Schönfinkel. Zum Entscheidungsproblem der mathematischen Logik. *Mathematische Annalen*, 99:342–372, 1928.

2. W. Bledsoe, R. Boyer, and W. Henneman. Computer proofs of limit theorems. *Artificial Intelligence*, 3:27–60, 1972.

3. D. Kalish, R. Montague, and G. Mar. *Logic: Techniques of Formal Reasoning*. Harcourt Brace Jovanovich, 1980.

4. R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *J. Automated Reasoning*, 8(2):183–212, 1992.

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

6. F. J. Pelletier. Completely non-clausal, completely heuristically driven, automated theorem proving. Department Computing Science TR82-7, University of Alberta, 1982.

7. F. J. Pelletier. Thinker. In J. H. Siekmann, editor, *Proceedings of the 8th International Conference on Automated Deduction*:701–702. Springer Verlag, Lecture Notes in Artificial Intelligence 230, 1985.

8. F. J. Pelletier. Seventy-five problems for testing automatic theorem provers. *J. Automated Reasoning*, 2:191–216, 1986.

9. F. J. Pelletier. Errata. *J. Automated Reasoning*, 4:235–236, 1988.

10. J. L. Pollock. *How to Build a Person*. Bradford/MIT Press, 1989.

11. J. L. Pollock. Interest driven suppositional reasoning. *J. Automated Reasoning*, 6(4):419–461, 1990.

12. G. Sutcliffe, C. B. Suttner, and T. Yemenis. The TPTP Problem Library. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*:252–266. Springer Verlag, Lecture Notes in Artificial Intelligence 814, 1994.

13. C. B. Suttner and G. Sutcliffe. The TPTP Problem Library (TPTP v1.2.0). Technical Report AR-95-03 (Technical Report 95/6), Institut für Informatik, Technische Universität München (Department of Computer Science, James Cook University), 1995.

New Book Available

Piton: A Mechanically Verified Assembly-Level Language, by J Strother Moore, is about how to prove that one practical programming language is correctly implemented on a conventional microprocessor. The programming language, called Piton, is a stack-based assembly-level language with recursive subroutine calls, symbolic names, and several different data types, including numbers, arrays, and addresses. The microprocessor on which Piton is implemented is the FM9001, a 32-bit general purpose microprocessor with 16 general purpose 32-bit wide registers, four 1-bit condition code registers, designated "carry," "zero," "overflow," and "negative," and up to 2^{32} words of memory. The FM9001 is interesting in part because it has been formally verified at the register-transfer level and it has been fabricated.

Piton is implemented via a compiler that maps a system of Piton programs and data into a binary image for the FM9001. The compiler works in several passes, including one that converts a symbolic relocatable image to absolute binary. The compiler is written as a recursive function in the Lisp-like Boyer-Moore logic. A theorem is proved establishing that the compiler is "correct" in the sense the same answers can be gotten by either of two routes: interpreting the assembly code or executing the binary on FM9001.

The book is unconventional because all of the foregoing is ultimately couched in a mathematical formalism. That is, the semantics of both Piton and the binary machine are presented as systems of mathematical equations describing the operations of the two abstract machines. The correctness formula was proved mechanically with the Boyer-Moore theorem prover.

The formal logic is explained informally before much use is made of it. Everything else is explained informally as well. Nevertheless, the mathematical formalization is offered as the definitive expression of the specification. Thus, this is really two intertwined books, one written in English and the other written in the formal logic.

For whom is this book written? The first prerequisite is an open mind with respect to the question of what mathematics can bring to the production of reliable computing systems. No theorem can be proved about a physical device. If one is inclined toward the view that this work is irrelevant because it proves a theorem about two mathematical models, then obviously this book isn't for that reader.

The informal book has been written for the computer scientist or computer science student. Knowledge of fundamental programming concepts is taken for granted. Thus, the reader should be familiar with such concepts as registers, memory, program counters, addresses, push down stacks, arrays, trees, lists, atomic symbols, jumps, conditionals, and subroutine call. Also, it is assumed that the reader is familiar the elementary mathematical concept of function. Even in the informal parts, it is assumed that the reader is willing to deal with some formal notation, namely, that for function application, including expressions built up from nested function applications. Finally, it would be helpful to be comfortable with the notion of recursively defined mathematical functions.

As for the formal book, virtually no computer science background is required. After all, the main theorem has been proved by a machine. Thus, except for the Nqthm logic (which is explained here only informally), everything one needs to understand Piton, the FM9001, the compiler, and

the theorem is explicitly presented in complete detail. Nothing is taken for granted beyond the ability to read the formulas in the logic (and a good memory for details).

About ordering the book: The price is not yet certain, but it is expected to be about $160. We are exploring with Kluwer the possibility of offering a *discount* to readers; such a discount will depend on the number of orders. Interested readers should send e-mail to veroff@cs.unm.edu.

Call for Papers: Special Issue of the Journal of Automated Reasoning

Deepak Kapur and Dongming Wang have issued a call for papers for a special issue on computer algebra (CA) and automated theorem proving (ATP), to be published in the *Journal of Automated Reasoning*. Original research papers describing recent advances and new insights on all aspects of coupling CA and ATP are solicited. Specific topics of interest include incorporating ATP mechanisms and tools into CA systems, combining CA and ATP systems, and applications of combined systems. The deadline for submission is **May 15, 1996**. All submitted papers will be refereed according to the JAR refereeing process. The special issue is expected to appear in 1997.

Authors are invited to submit 3 copies of their manuscripts to Prof. Deepak Kapur, Department of Computer Science, State University of New York, Albany, NY 12222, Fax: (1) 518 442 5638, e-mail: kapur@cs.albany.edu, or to Dr. Dongming Wang, LIFIA–IMAG–CNRS, 46, avenue Félix Viallet, 38031 Grenoble Cedex, France; Fax: (33) 76 57 46 02, e-mail: wang@lifia.imag.fr.

The Passing of Three Pioneers

Larry Wos

With deep sadness, I report the passing this year of three of the pioneers of the field. Within the recent weeks, Woody Bledsoe (with whom I shared many gratifying discussions) succumbed to a long illness. No replacement can be found.

Before him, Alonzo Church and Hao Wang passed away. Church died August 11, 1995, in Hudson, Ohio at the age of 92. Memorial contributions may be made to the Association for Symbolic Logic (1409 West Green Street, Urbana, IL 61801) and marked "For the Alonzo Church Fund".

Hao Wang died in the spring of this year. He will be remembered for his work in mathematical logic and his contributions to the beginnings of automated theorem proving in a significant manner. A session at the GOEDEL '96 conference (see elsewhere in this issue) is planned in honor of Hao Wang.