ASSOCIATION FOR AUTOMATED REASONING
NEWSLETTER

No. 30 August 1995

From the AAR President, Larry Wos...

Amazing to me is the fact that this is the thirtieth issue of the *AAR Newsletter.* I am delighted to report that—more than ever—the newsletter is stimulating both discussion and research. With regard to discussion, the dialogue on CADE continues, with David Plaisted's response to Alan Bundy's article in the May 1995 *AAR Newsletter* and with my own comments on the possible CADE reorganization. With regard to research, Uwe Egly and Thomas Rath have written an article on the halting problem, using a formulation presented by Li Dafa in the October 1994 *AAR Newsletter*; and Geoff Sutcliffe has devised an elegant version of sound unification in Prolog.

Working on such challenging problems can be exciting—and rewarding. I offer my congratulations to Chou, Zhang, Gao, and Yang for winning a first prize in the Natural Science Awards of the Chinese Academy of Sciences. The future of automated reasoning is indeed promising!

A Response to Alan Bundy

David A. Plaisted

I wanted to briefly respond to some of the points made in Alan Bundy's article in the May 1995 *AAR Newsletter.* First let me emphasize that my January 1995 *AAR Newsletter* article proposing a democratic system for CADE was endorsed and supported by a number of others in the community. Currently there are about 28 supporters.

Alan disputes my assertion that "there is no democratic element at all in the current system," on the basis that it is possible to change the by-laws democratically at a CADE meeting. But I mentioned this possibility in my *AAR Newsletter* article and was referring to the current by-laws, as my article made clear. In the current system, PC chairs are chosen by the trustees, who then become new trustees. The trustees have stated that they will consider program committee recommendations for program chair very seriously, and this is a good feature. Still, the program committee and the members have no direct input, except as the trustees choose to consider it.

Alan says that I propose having both a board of trustees and an executive committee. He also says that I propose that the trustees be a much larger board than the current nine members. Both assertions are incorrect. CADE officers are not necessarily distinct from the trustees in our (democratic) proposal. There is no need for a president, vice president, et cetera; the trustees can choose a president from among themselves, and they can perform their duties without a vice president. It may be necessary to choose one or two other officers, who may or may not be trustees. My proposal was designed to be very simple and avoid creating complexity. The trustees would

be elected and would choose program chairs. There would be no executive committee. Program chairs would not become trustees unless they were elected.

Alan says further that if the center of gravity of CADE were to shift away from the wishes of its members, the conference would die, and the trustees are unlikely to let this happen. This reasoning is too simplified. In the first place, this would not necessarily cause CADE to die. It may just become weak. Furthermore, this may be unlikely, but it is not impossible. Conferences are born and die all the time. In addition, it does not take much imagination to see that the trustees can make suboptimal choices for a number of reasons. In the current system, each program chair, whether he or she was good or bad, becomes a trustee and helps to choose future chairs and trustees. There is a mechanism for removing bad program chairs, but this would probably be used only in the most extreme cases because of the embarrassment it would cause. A democratic system provides a better way to correct such (hypothetical) problems. And of course, a democratic system gives everyone a direct voice in the choice of the trustees, a feature not provided for in the current system.

I would like to note that the supporters of the CADE proposition have changed their emphasis to the RTA bylaws as a possible model for CADE, because of their simplicity, though both proposals are reasonable, and some people may still prefer the original one. The RTA system is actually quite similar to the one proposed in the January issue of the *AAR Newsletter*.

For a fuller version of this response, please send e-mail to plaisted@cs.unc.edu. This longer version was posted to the rewriting and theorem proving mailing lists.

Proposed CADE By-Laws

David A. Plaisted

I would like to make a copy of a proposed new set of by-laws for CADE available to the public, in order to get as much input as possible before finalizing them. In the current system, CADE is administered by a committee of trustees composed primarily of past and future program chairs, and program chairs are chosen by the trustees, with input from the program committee. The purpose of these proposed new by-laws is to introduce a democratic system for electing the CADE trustees. There will likely be a vote on these by-laws at CADE 96. These by-laws have been developed in response to discussions with many people. The by-laws and all of their features also received an overwhelmingly favorable response from those 36 people who responded to our recent CADE survey. Before the survey, 28 people had indicated their support for a democratic system in CADE. Now this number has grown to 46. However, we would also like your input.

In general, these proposed new by-laws stipulate that CADE will be administered by a group of nine trustees, three elected each year for three-year terms. The elections will occur at CADE business meetings, where other matters also can be discussed. The trustees will choose CADE program chairs, local arrangements chairs, and conference sites. The constituency for CADE will be (approximately) the registered CADE attendees, plus members of the Association for Automated Reasoning. Trustees will not be permitted to serve three consecutive terms, but otherwise can serve any number of terms. These by-laws will be subject to revisions based on

the comments received. They are very similar to those contained in our recent survey. We may publish a final version of the bylaws in the next issue of the AAR Newsletter. For an on-line copy of the (nearly) most recent version of these by-laws, use the world-wide web at URL http://www.cs.unc.edu/mi/mi.html or send e-mail to me (plaisted@cs.unc.edu) or write to me at the following address:

David A. Plaisted
Department of Computer Science
CB# 3175, 352 Sitterson Hall
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27599-3175

You may also send comments about the by-laws to me at these addresses. Your comments will be kept confidential.

I would like to comment on some related matters. Another system that has been proposed is to have six trustees with six-year terms, one elected each year. This would mean that the program chair would likely be elected each year, and so we would have a system not much different from the current one. I would also like to respond to the claim that we should not change the current system because CADE has been successful under it. The current trustee system is relatively new and significantly different from the former system, under which the program chair was chosen by the previous program committee, and each program chair had a relatively short term. Therefore the current trustee system cannot claim responsibility for the past success of CADE. I would also like to emphasize the importance of a secret vote when this issue is (likely) voted on at CADE 96, and to make public my request that if proxies are permitted, each individual only be permitted to have one proxy, that is, to cast his or her own vote and at most one vote by proxy.

Thank you for considering these issues.

Another View of the Proposed Changes to CADE

Larry Wos

At the quality of automated reasoning programs that now exist, I am awed; at the successes obtained with them, I am astounded. The cited growth and importance can be traced to a large degree to the manner in which CADE has functioned. Hence I now feel it imperative to express my opinion—both as a researcher and as president of AAR—about the proposed revisions to the CADE organization.

For many years, CADE has automatically included past chairmen in the planning of subsequent conferences. The cited practice—formalized in the by-laws—has, without question, contributed to the success of CADE. The experience and knowledge of these individuals have provided essential guidance to the CADE organization. It seems so clear to me that altering such a policy—and thereby risking the possibility of losing this expertise—is a step fraught with pitfalls.

Another key factor in the success of CADE has been, I believe, the avoidance of politics. Too often over the past thirty years, I have seen whole organizations destroyed because of politics.

The proposed alteration to the CADE bylaws raises the spectre of politicking—of small groups lobbying for votes or pressuring others to support a particular candidate. The price, to researchers in automated reasoning and to the field itself, could be enormous!

In regard to the issue of voting requirements, I confess to puzzlement, for logic presents only one choice. CADE is a subcorporation of AAR. If one is to vote on an issue affecting CADE, one must be a member of AAR. Mere attendance at a conference is not sufficient.

Significant changes to CADE are being proposed. What I advise is that we proceed with the greatest of care—even if a final decision must be delayed until after the next CADE. What I wish is that these matters be weighed with wisdom, with care, and with a realization of possible consequences.

The FM9001 Microprocessor: Its Formal Specification and Mechanical Correctness Proof

Bishop C. Brock (brock@cli.com) and Warren A. Hunt, Jr. (hunt@cli.com)

We are releasing the mechanically checked proof scripts for the FM9001 microprocessor. The FM9001 is a general-purpose 32-bit microprocessor which has been implemented as a CMOS ASIC. The proof being released rigorously connects the expression of the FM9001 as a netlist with the characterization of the FM9001 at the machine-code programmer's level. (The FM9001 is the foundation of the 'CLI Stack', which also includes several verified compilers and applications all running on the FM9001. Other parts of the 'CLI Stack' are separately released.)

To obtain information about the FM9001 microprocessor and proof, please examine the URL

http://www.cli.com/hardware/fm9001.html

To obtain the FM9001 system, connect to Internet site ftp.cli.com by anonymous ftp, giving your e-mail address as the password, get the file /pub/fm9001/README, and follow the instructions therein. Or get the URL

ftp://ftp.cli.com/pub/fm9001/README

via a WWW browser.

Researchers Receive Prestigious First Prize

Shang-Ching Chou, Jiag-Zhong Zhang, Xiao-Shan Gao, and Lu Yang have been awarded a first prize in the Natural Science Awards of the Chinese Academy of Sciences. This is one of the highest awards in China and is bestowed only once every two years. It includes all areas in natural sciences, from mathematics and physics to agriculture and geology. Chou, Zhang, Gao, and Yang were the only researchers to receive first prize in computer science. The award is a tribute to their outstanding work in geometry, mechanics, and theorem proving and is good news for the whole discipline of automated reasoning.

Call for Papers

Symposium on Artificial Intelligence and Mathematics

The Fourth International Symposium on Artificial Intelligence and Mathematics will take place on January 3–5, 1996, in Fort Lauderdale, Florida. The goal is to foster interactions among mathematics, theoretical computer science, and artificial intelligence. The meeting includes paper presentation, invited speakers, and special topic sessions. Topic sessions in the past have covered computational learning theory, nonmonotonic reasoning, and computational complexity issues in artificial intelligence.

The deadline for submissions is September 1, 1995. Authors should submit extended abstracts (up to 10 double-spaced pages) by e-mail (postscript) selman@research.att.com or send five copies to Bart Selman, AT&T Bell Laboratories, Room 2T-414, 600 Mountain Avenue, Murray Hill, NJ 07974. Accepted papers will be included in a thoroughly refereed volume of the series *Annals of Mathematics and Artificial Intelligence,* J. C. Baltzer Scientific Publishing Co.

The Symposium is partially supported by the Annals of Math and AI, Florida Atlantic University, and the Florida- Israel Institute. Partial travel subsidies may be available to junior researchers. For further information and future announcements contact Frederick Hoffman, Florida Atlantic University, Department of Mathematics, PO Box 3091, Boca Raton, FL 33431; e-mail: hoffman@acc.fau.edu or hoffman@fauvax.bitnet.

JSC Special Issue on Executable Temporal Logics

The *Journal of Symbolic Computation* invites authors to submit papers on all aspects relating to the foundations, implementation techniques and applications of languages based upon temporal logic. The research described must not only incorporate an adequate level of technical detail, but must also provide a clear indication of both the utility and the applicability of the results. Topics of interest include theoretical issues in executable temporal logics, design of executable temporal logics, relationship between execution and temporal theorem-proving, operational models and implementation techniques, programming support and environments, comparative studies of languages, relationship of executable temporal logics to (temporal) databases, applications, and case studies.

In addition to longer papers, short papers (5 to 10 pages) describing specific features or novel applications of executable temporal logic. Submissions should follow the JSC style guide available from ftp://ftp.risc.uni-linz.ac.at/pub/jsc. LaTeX users are encouraged to use the jsc.sty file. Electronic submission is strongly encouraged (either as self-contained LaTeX or postscript). Submissions, either electronic or a paper copy of the full paper, should arrive no later than October 15, 1995, and should be sent to the principal guest editor Michael Fisher, Department of Computing, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom; tel: +44 161 247 1488; fax: +44 161 247 1483; e-mail: M.Fisher@doc.mmu.ac.uk.

FroCoS'96

The first international workshop on Frontiers of Combining Systems will take place on March 26–29, 1996, in Munich, Germany.

In various areas of logic, computation, language processing, and artificial intelligence there is an obvious need for using specialized formalisms and inference mechanisms for special tasks. In order to be usable in practice, these specialized systems must be combined with each other, and they must be integrated into general purpose systems. The development of general techniques for the combination and integration of special systems has been initiated in many areas. The workshop "Frontiers of Combining Systems" intends to offer a common forum for these research activities. Furthermore, it gives the possibility to present results on particular instances of combination and integration, and on their practical use.

Topics of interest for the workshop include

- combination of logics (e.g., modal logics, logics in AI)
- combination of constraint solving techniques (unification and matching algorithms, general symbolic constraints, numerical constraints) and combination of decision procedures
- integration of equational and other theories into deductive systems (e.g., theory resolution, constraint resolution, constraint paramodulation)
- combination of term rewriting systems
- integration of data structures (e.g., sets, multisets, lists) into CLP formalisms and deduction processes
- hybrid systems in computational linguistics, knowledge representation, natural language semantics, and human computer interaction
- logic modeling of multi-agent systems.

A PostScript version of the full paper (preferable LaTeX format), not exceeding 15 pages (incl. title page and references), should be received via e-mail by October 16, 1995. In addition, one hard copy of the paper should be received by the same date. Selected papers will be published in the Kluwer series on "Applied Logic".

Please send submissions to the local organizer K. U. Schulz, CIS, University of Munich, Wagmuellerstr. 23, D-80538 Muenchen, Germany; e-mail: schulz@cis.uni-muenchen.de.

Information on FroCoS'96 is available by WWW: `http://www.cis.uni-muenchen.de` (under "events").

ELP'96

The Fifth International Workshop on Extensions of Logic Programming will take place on March 28–30, 1996, in Leipzig, Germany. ELP aims at stimulating research on extensions of logic programming languages, especially those based on proof theory, and seeks to disseminate insights into the relations between the logics of those languages, implementation techniques, and the use of these languages in applications. Deadline for submissions is September 15, 1995. For more details see the WWW page http://www-theory.dcs.st-and.ac.uk/elp96.html or send e-mail to elp96@informatik.uni-leipzig.de. Submissions should be sent to ELP'96/Heinrich Herre, Institut fuer Informatik, Universitaet Leipzig, Augustusplatz 10-11, D-04109 Leipzig, Germany. Phone: +49 341 973 2201, Fax: +49 341 973 2209.

FLAIRS-96

Two recent advances in automated reasoning research have led the field closer to its goal of providing effective procedures for deductive reasoning and automated theorem proving. Many novel techniques have been proposed in the literature for controlling the amount of search required by an automated reasoning system to find its desired result. Also, there is a new emphasis on experimental evaluation of proposed systems, facilitated by Sutcliffe and Suttner's TPTP library, a collection of thousands of problems in first order logic. The synergy between these trends has increased the need for forums where developers of implemented automated reasoning systems can discuss their techniques and empirical findings. Automated reasoning systems that deal with first-order logic serve as the deductive engine for disjunctive deductive database systems (DDDB). Thus advances in controlling search in these automated reasoning systems also lead to improved strategies for answering queries in DDDB.

The track "Controlling Search in Automated Reasoning Systems" will take place on May 20–22, 1996, in Key West, Florida. This track will attract developers and users of implemented automated reasoning and DDDB systems. Of interest are papers that describe novel techniques that limit the amount of search required by automated reasoning and DDDB systems; implemented automated reasoning and DDDB systems; and empirical results on problems, such as those in TPTP.

Authors are encouraged to submit electronically by October 13, 1995, by mailing an encoded compressed PostScript file to bspencer@unb.ca (instructions below). Submissions may also be physically mailed by October 15, 1995, by sending four copies to the Program Committee Chair (address below). All accepted papers will be published with the FLAIRS-96 proceedings. The submitted paper should not exceed 4000 words, including abstract, references and figures.

Program chair: Bruce Spencer, Faculty of Computer Science, Gillin Hall, GWE-126, University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, CANADA E3B 5A3. FAX: 506-453-3566; phone: 506-453-4566; e-mail: bspencer@unb.ca;
http://www.cs.unb.ca/profs/bspencer/home.html

Electronic submission information: To create a compressed form of the PostScript file my_paper.ps as my_paper.ps.Z, do the following: compress my_paper.ps. To encode the binary my_paper.ps.Z into the ASCII mailfile, suitable for mailing:

```
uuencode my\_paper.ps.Z my\_paper.ps.Z > mailfile
```

Include mailfile in an electronic mail message to bspencer@unb.ca, along with a letter that identifies to whom correspondence should be addressed.

Coq V5.10 Release

The Coq Proof Assistant, Version 5.10, has been released.

For users who are already using one of the beta-test versions of V5.10, this final release has improved documentation, streamlining of tactics (some adaptation of scripts may be necessary because the names of inversion tactics have changed), and compressed theory files (.vo) which keep the size of the installation much smaller and are uniform across architectures. This substantial improvement necessitates, however, the installation of the latest release of Caml Light 0.7, which is available by anonymous ftp on ftp.inria.fr, in INRIA/lang/caml-light. Users also need to install the Caml Light contribution 'libunix'.

The new system has been completely rebuilt. Among its new features are authorization of mutually recursive inductive families, compiled theory modules, extensible parsers and pretty-printers, user-programmable tactics, and synthesis of implicit type arguments. The distribution also includes numerous new user-contributed libraries.

The current release works under most modern Unix platforms. A specialized interface with the Centaur environment, called CTCoq, is under completion in Sophia-Antipolis and should be available soon in beta test. A Macintosh version will be available next fall.

The release may be taken from machine ftp.inria.fr, directory INRIA/coq/V5.10, archive V5.10.tar.z and README file. Anomalies should be reported to

coq@pauillac.inria.fr

and general questions to

coq-club@pauillac.inria.fr.

Implementing Sound Unification 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 such data, but doing so does make life easy in many respects. Given this approach, sound unification (i.e., with occurs check) of Prolog terms is required for implementing ATP inference operations. Some Prolog implementations now provide sound unification directly, and the problem is solved. In other cases, sound unification is a procedure that ATP implementors still have to code up.

I have played with various Prolog implementations of sound unification. 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!

```
%----------------------------------------------------------------------
%----Identical things unify. This is a short cut, and also used by pairs
%----of empty lists.
unify(Term1,Term2):-
    Term1 == Term2,
    !.

%----There's a variable about
unify(Variable,Term):-
    var(Variable),
    !,
%----Do occurs check
%----Copy the term. This should be replaced by a built in copy_term/2, if
%----available, because asserting and retracting is slow.
    asserta(saved_term(Term)),
    retract(saved_term(TermCopy)),
%----Count how many variables in the copy
    numbervars(TermCopy,0,NumberOfVariables),
    \+ \+ (
%----Instantiate the variable
        Variable = dummy_value_no_one_will_use,
%----Make sure the original has the same number of variables still. If it
%----doesn't then the Term contains the variable.
        numbervars(Term,0,NumberOfVariables)
        ),
```

```
    Variable = Term.

unify(Term,Variable):-
    var(Variable),
    !,
    unify(Variable,Term).

%----Lists of terms
unify([H1|T1],[H2|T2]):-
    !,
    unify(H1,H2),
    unify(T1,T2).

%----Functions
unify(Function1,Function2):-
    Function1 =.. [Functor|Arguments1],
    Function2 =.. [Functor|Arguments2],
    unify(Arguments1,Arguments2).
%----------------------------------------------------------------------
```

The Halting Problem: An Automatically Generated Proof

Uwe Egly and Thomas Rath
e-mail: {uwe,rath}@intellektik.informatik.th-darmstadt.de

1 Introduction

In the 1994 fall issue no. 27 of the *AAR Newsletter,* Larry Wos asked for resolution-style proofs for the famous halting problem. In the following, we present a proof that was obtained by our theorem prover KoMeT without any interaction or assistance. We use Dafa's formalization presented in [6], which itself is a modification of Burkholder's original formalization in [3]. For these formalizations, several unsuccessful attempts to prove the halting problem by resolution-oriented theorem provers were reported in [2, 5]. There are, however, natural deduction (ND) proofs of Burkholder's original formalization (see, e.g., [5]) that are obtained with support of automated theorem provers, but for the new formalization, there is only one hand-crafted ND proof in [6]. In contrast, our proof was found by KoMeT automatically. KoMeT is based on clausal connection tableaux and therefore requires the input formula being in clausal normal form. The key feature of KoMeT that enables it to find the proof of the halting problem is the integrated definitional (or structure-preserving)

transformation[1] [7, 8, 12] of a formula into clause form.

The paper is organized as follows. In Section 2, the formalization of the halting problem is reconsidered. We describe the way the formula is translated into *disjunctive*[2] normal form and present the normal form consisting of 75 clauses. A presentation of the proof found by KoMeT is given in Section 3. We conclude the paper with some general remarks about the practical value of definitional translations into normal form.

2 The Halting Problem

The formalization is taken from [6]. Table 1 presents the intuitive meaning of predicates used in subsequent sections. The formalization is as follows.

Predicate	Meaning
$a(X)$	X is an algorithm
$c(X)$	X is a computer program in some programming language
$d(X,Y,Z)$	X is able to decide whether Y halts, given input Z
$h_2(X,Y)$	X halts on a given input Y
$h_3(X,Y,Z)$	X halts on given as input the pair $\langle X,Y\rangle$
$o(X,Y)$	X outputs Y

Table 1: Intuitive meaning of the predicates

$$
\begin{aligned}
&(\exists X(a(X) \wedge \forall Y(c(Y) \rightarrow \forall Z d(X,Y,Z)))) \rightarrow \\
&\quad \exists W(c(W) \wedge \forall Y(c(Y) \rightarrow \forall Z d(W,Y,Z))) && (1)\\
&\forall W((c(W) \wedge \forall U(c(U) \rightarrow \forall V d(W,U,V))) \rightarrow \\
&\quad \forall Y,Z((c(Y) \wedge h_2(Y,Z) \rightarrow (h_3(W,Y,Z) \wedge o(W,g)) \wedge \\
&\quad (c(Y) \wedge \neg h_2(Y,Z) \rightarrow (h_3(W,Y,Z) \wedge o(W,b)))))) && (2)\\
&\forall W((c(W) \wedge \\
&\quad \forall Y,Z((c(Y) \wedge h_2(Y,Z) \rightarrow (h_3(W,Y,Z) \wedge o(W,g)) \wedge \\
&\qquad (c(Y) \wedge \neg h_2(Y,Z) \rightarrow (h_3(W,Y,Z) \wedge o(W,b)))) \rightarrow \\
&\qquad \exists V(c(V) \wedge \forall Y(((c(Y) \wedge h_3(W,Y,Y) \wedge o(W,g)) \rightarrow \neg h_2(V,Y)) \wedge \\
&\qquad ((c(Y) \wedge h_3(W,Y,Y) \wedge o(W,g)) \rightarrow (h_2(V,Y) \wedge o(V,b))))))) && (3)\\
&\neg(\exists X(a(X) \wedge \forall Y(c(Y) \rightarrow \forall Z d(X,Y,Z)))) && (4)
\end{aligned}
$$

An explanation of the different formulae (1) to (4) can be found in [6]. The problem is to prove

$$(1) \wedge (2) \wedge (3) \rightarrow (4). \qquad (5)$$

[1] The translation is performed by a program called NFT [13], which is available by ftp.

[2] We adopt the positive affirmative representation [1] of a formula here. Instead of transforming $\neg F$ into conjunctive normal form, F is transformed into disjunctive normal form.

By introducing labels for subformulae, (5) is transformed into clause form. Table 2 shows the labels introduced for the different subformulae. The same label is introduced for different syntactically identical copies of the same subformula. The conjunction of these formulae of the form

label connector subformula

implies d_{40}, the label of (5), whereby *connector* depends on the polarity. If the polarity is `p` (positive), then $\rightarrow$ is used. If the polarity is `n` (negative), then the connector is $\leftarrow$. In the remaining case, the connector is $\equiv$. The resulting implication is transformed into a *disjunctive* normal form. The resulting clause set $\mathcal{H}$ is depicted below.

C_1 `[-(d39),-(d40)].`
C_2 `[-(d5),-(d40)].`
C_3 `[-(d11),d39].`
C_4 `[-(d38),d39].`
C_5 `[d11,d5,-(d10)].`
C_6 `[d4(A),-(d5)].`
C_7 `[d5,-(d4(sc1))].`
C_8 `[d4(A),-(d4(sc1))].`
C_9 `[a(A),d3(A),-(d4(A))].`
C_{10} `[d4(A),-(a(A))].`
C_{11} `[d4(A),-(d3(A))].`
C_{12} `[d2(sf1(A),A),-(d3(A))].`
C_{13} `[d3(A),-(d2(B,A))].`
C_{14} `[d2(sf1(A),A),-(d2(B,A))].`
C_{15} `[-(c(A)),-(d2(A,B))].`
C_{16} `[d1(A,B),-(d2(B,A))].`
C_{17} `[d2(A,B),c(A),-(d1(B,A))].`
C_{18} `[d(A,B,sf2(B,A)),-(d1(A,B))].`
C_{19} `[d1(A,B),-(d(A,B,C))].`
C_{20} `[d(A,B,sf2(B,A)),-(d(A,B,C))].`
C_{21} `[d10,-(d9(sc2))].`
C_{22} `[-(c(A)),d9(A)].`
C_{23} `[-(d8(A)),d9(A)].`
C_{24} `[d8(A),-(d7(B,A))].`
C_{25} `[d7(A,B),c(A),-(d6(B,A))].`
C_{26} `[d6(A,B),-(d(A,B,C))].`
C_{27} `[-(d25),d38].`
C_{28} `[-(d37),d38].`
C_{29} `[d25,-(d24(A))].`
C_{30} `[d24(A),d15(A),-(d23(A))].`
C_{31} `[c(A),d14(A),-(d15(A))].`
C_{32} `[d13(sf3(A),A),-(d14(A))].`
C_{33} `[-(c(A)),-(d13(A,B))].`
C_{34} `[d12(A,B),-(d13(B,A))].`
C_{35} `[d(A,B,sf4(B,A)),-(d12(A,B))].`
C_{36} `[d22(sf5(A),sf6(A),A),-(d23(A))].`
C_{37} `[d23(A),-(d22(B,C,A))].`
C_{38} `[d22(sf5(A),sf6(A),A),-(d22(B,C,A))].`
C_{39} `[d18(A,B,C),d21(A,B,C),-(d22(A,B,C))].`
C_{40} `[d22(A,B,C),-(d18(A,B,C))].`
C_{41} `[d22(A,B,C),-(d21(A,B,C))].`
C_{42} `[-(d16(A,B)),-(d18(A,B,C))].`
C_{43} `[d17(A,B,C),-(d18(B,C,A))].`
C_{44} `[d18(A,B,C),d16(A,B),-(d17(C,A,B))].`
C_{45} `[c(A),h2(A,B),-(d16(A,B))].`
C_{46} `[d16(A,B),-(c(A))].`
C_{47} `[d16(A,B),-(h2(A,B))].`
C_{48} `[h3(A,B,C),o(A,g),-(d17(A,B,C))].`
C_{49} `[d17(A,B,C),-(h3(A,B,C))].`
C_{50} `[d17(A,B,C),-(o(A,g))].`
C_{51} `[-(d19(A,B)),-(d21(A,B,C))].`
C_{52} `[d20(A,B,C),-(d21(B,C,A))].`
C_{53} `[d21(A,B,C),d19(A,B),-(d20(C,A,B))].`
C_{54} `[c(A),-(h2(A,B)),-(d19(A,B))].`
C_{55} `[d19(A,B),-(c(A))].`
C_{56} `[d19(A,B),h2(A,B)].`
C_{57} `[h3(A,B,C),o(A,b),-(d20(A,B,C))].`
C_{58} `[d20(A,B,C),-(h3(A,B,C))].`
C_{59} `[d20(A,B,C),-(o(A,b))].`
C_{60} `[d37,-(d36(A))].`
C_{61} `[d36(A),d26(A),-(d35(A))].`
C_{62} `[c(A),d23(A),-(d26(A))].`
C_{63} `[d35(A),-(d34(sf7(A),A))].`
C_{64} `[-(c(A)),d34(A,B)].`
C_{65} `[-(d33(A,B)),d34(B,A)].`
C_{66} `[d33(A,B),-(d32(C,A,B))].`
C_{67} `[-(d28(A,B,C)),d32(A,B,C)].`
C_{68} `[-(d31(A,B,C)),d32(A,B,C)].`
C_{69} `[d28(A,B,C),d27(A,B),h2(C,A)].`
C_{70} `[c(A),h3(B,A,A),o(B,g),-(d27(A,B))].`
C_{71} `[d31(A,B,C),d29(A,B),-(d30(C,A))].`
C_{72} `[c(A),h3(B,A,A),o(B,b),-(d29(A,B))].`
C_{73} `[-(h2(A,B)),d30(A,B)].`
C_{74} `[-(o(A,b)),d30(A,B)].`
C_{75} `[d40].`

3 A Description of the Obtained Proof

In this section, we present the proof of the definitional translation of (5) found by our theorem prover KoMeT. KoMeT is written in Prolog and consists of three main parts: a module for transforming a first-order formula into a normal form as mentioned above, a module for preprocessing reductions, and a module compiling a formula into a Prolog program. The Prolog program generated for a formula simulates the proof search in a connection tableaux, using several refinements (e.g., regularity, failure lemmata, tautology- and subsumption-constraints) [9, 11].

The basic calculus of KoMeT can be seen as a variant of Loveland's model elimination (ME) or Kowalski and Kuehner's SL-resolution. There are two major inferences, namely, extensions (ext) and reductions (red). An extension step is similar to a linear input resolution step. In contrast to linear input resolution, literals resolved upon are not simply deleted from the center clause but stored as framed (or boxed) literals in the resolvent. Hence, ME works not on clauses but on extended clauses containing two different kinds of literals. These framed literals correspond to literals on the path in Bibel's connection calculi and can be used only in reduction steps. For instance, if $C \vee \boxed{A} \vee B$ is the center clause, the clause $C \vee \boxed{A} \vee \boxed{B} \vee D \vee \neg A$ can be obtained by an extension with the side clause $\neg B \vee D \vee \neg A$. Next, a reduction step is possible; specifically, $\neg A$ can be solved by unifying A with the framed literal $\boxed{A}$ and $C \vee \boxed{A} \vee \boxed{B} \vee D$ is the resulting new center clause.

KoMeT was able to find a proof for Dafa's variant of the halting problem in 22 seconds [3] on a SPARC-Station 20. [4] The shortest proof we found is listed below; it consists of 77 inferences steps (57 extension and 20 reduction steps). Within this listing, we employ the abbreviation 'ext(...)' for extension steps and 'red(...)' for reduction steps. The numbers given in brackets indicate the literals used for the respective proof step. The first and third numbers indicate the number of the clause and the number of the literal within this clause (numbered from left to right). The second and fourth numbers indicate the instance numbers of the used clauses. For example 'ext(75-1, 0, 1-2, 1)' is the abbreviation for an extension step from literal 1 of clause 75 (the unit clause `d40`) with index number 0, to literal 2 of clause 1 (`-d40`) with index number 1. The proof is as follows.

```
ext(75-1,0,1-2,1), ext(1-1,1,3-2,2), ext(3-1,2,5-1,3),
ext(5-3,3,21-1,4), ext(21-2,4,22-2,5), ext(22-1,5,31-1,31),
ext(31-3,31,30-2,32), ext(30-1,32,29-2,33), ext(29-1,33,27-1,34),
ext(27-2,34,4-1,35), red(4-2,35,1-1,1), ext(30-3,32,62-2,2818),
ext(62-3,2818,61-2,2819), ext(61-3,2819,63-1,2820), ext(63-2,2820,64-2,2821),
ext(64-1,2821,45-1,2938), ext(45-3,2938,47-1,2940), ext(47-2,2940,69-3,2967),
ext(69-2,2967,70-4,2968), ext(70-3,2968,50-2,2969), ext(50-1,2969,44-3,2970),
red(44-2,2970,45-3,2938), ext(44-1,2970,40-2,2971), ext(40-1,2971,37-2,2972),
red(37-1,2972,30-3,32), ext(70-2,2968,49-2,2973), ext(49-1,2973,44-3,2974),
red(44-2,2974,45-3,2938), ext(44-1,2974,40-2,2975), ext(40-1,2975,37-2,2976),
red(37-1,2976,30-3,32), red(70-1,2968,64-1,2821), ext(69-1,2967,67-1,2977),
```

[3] This is the time for processing the whole problem including all preparation and compilation steps.

[4] KoMeT has also obtained proofs for Burkholder's original formalization in less than 35 seconds if the definitional translation of the formula is used.

```
ext(67-2,2977,66-2,2978), ext(66-1,2978,65-1,2979), red(65-2,2979,63-2,2820),
ext(45-2,2938,73-1,3041), ext(73-2,3041,71-3,3042), ext(71-2,3042,72-4,3043),
ext(72-3,3043,59-2,3044), ext(59-1,3044,53-3,3045), ext(53-2,3045,54-3,3049),
red(54-2,3049,45-2,2938), red(54-1,3049,64-1,2821), ext(53-1,3045,41-2,3050),
ext(41-1,3050,37-2,3051), red(37-1,3051,30-3,32), ext(72-2,3043,58-2,3059),
ext(58-1,3059,53-3,3060), ext(53-2,3060,54-3,3064), red(54-2,3064,45-2,2938),
red(54-1,3064,64-1,2821), ext(53-1,3060,41-2,3065), ext(41-1,3065,37-2,3066),
red(37-1,3066,30-3,32), red(72-1,3043,64-1,2821), ext(71-1,3042,68-1,3067),
ext(68-2,3067,66-2,3068), ext(66-1,3068,65-1,3069), red(65-2,3069,63-2,2820),
ext(61-1,2819,60-2,3070), ext(60-1,3070,28-1,3071), ext(28-2,3071,4-1,3072),
red(4-2,3072,1-1,1), red(62-1,2818,22-1,5), ext(31-2,31,32-2,3073),
ext(32-1,3073,33-2,3074), ext(33-1,3074,25-2,3159), ext(25-3,3159,26-1,3160),
ext(26-2,3160,35-1,3176), ext(35-2,3176,34-1,3177), red(34-2,3177,32-1,3073),
ext(25-1,3159,24-2,3178), ext(24-1,3178,23-1,3179), red(23-2,3179,21-2,4),
ext(5-2,3,2-1,3180), red(2-2,3180,75-1,0)
```

4 Conclusion

We reported our successful efforts in proving the halting problem for which, up to now, there was no automatically generated proof. The key feature of KoMeT which enables the proof is its definitional translation to normal form. In order to evaluate this assertion, we have translated the resulting clause set (of the definitional translation) into Otter syntax and proved it in about 15 seconds.[5] Hence, we have a realistic example for the practical value of this kind of a translations which are widely neglected in the automated deduction community. The reason for this neglect might be the linear increase (in terms of literal occurrences) of the length of the resulting normal form which is introduced by such definitions, and the difficulty to avoid the deductive generation of the conventional normal form from the definitional normal form. Moreover, in most cases, only clause sets are considered as the theorem prover's input. But dealing with a problem is more than simply refuting a given clause set. The consideration has to start with the predicate logic formula and has to take all possible manipulations like antiprenexing, optimal Skolemization, and definitional translations into account. By simply reducing the given formula into normal form without any further intelligent manipulation, a chance is lost to drastically improve the overall efficiency of Automated Deduction systems.

Acknowledgment

The authors thank W. Bibel and K. Genther for their constructive criticism and for their useful comments on an earlier draft of this paper. This research was partially supported by the DFG under grant Bi228/6-3.

[5] For the original formalization, a proof was found in less than 2 minutes if the definitional translation was used.

References

[1] W. Bibel. *Deduction: Automated Logic*. Academic Press, London, 1993.

[2] M. Bruschi. The Halting Problem. *AAR Newsletter*, pages 7–12, March 1991.

[3] L. Burkholder. The Halting Problem. *SIGACT News*, 18(3):48–60, 1987.

[4] C. L. Chang and R. C. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

[5] L. Dafa. A Mechanical Proof of the Halting Problem in Natural Deduction Style. *AAR Newsletter*, pages 4–9, June 1993.

[6] L. Dafa. The Formulation of the Halting Problem Is Not Suitable for Describing the Halting Problem. *AAR Newsletter*, pages 1–7, October 1994.

[7] E. Eder. An Implementation of a Theorem Prover Based on the Connection Method. In W. Bibel and B. Petkoff, editors, *AIMSA 84, Artificial Intelligence - Methodology, Systems, Applications, Varna, Bulgaria*, Amsterdam, September 1984. North-Holland.

[8] E. Eder. *Relative Complexities of First Order Calculi*. Vieweg, Braunschweig, 1992.

[9] R. Letz. *First-Order Calculi and Proof Procedures for Automated Deduction*. Ph.D. thesis, TH Darmstadt, 1993.

[10] D. W. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science*. North–Holland Publishing Company, Amsterdam, New York, Oxford, 1978.

[11] K. Mayr. Integrating Antilemmata into Model Elimination. Technical report, TU München, 1993.

[12] D. A. Plaisted and S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[13] Tran van Thanh Liem and S. Merker. NFT: Der Normalformtransformator V 1.0, 1994. System description.

label	subformula	polarity
$d_1(X,Y)$	$\forall Z\ d(X,Y,Z)$	p/n
$d_2(Y,X)$	$c(Y) \rightarrow d_1(X,Y)$	p/n
$d_3(X)$	$\forall Y\ d_2(Y,X)$	p/n
$d_4(X)$	$a(X) \wedge d_3(X)$	p/n
d_5	$\exists X\ d_4(X)$	p/n
$d_6(W,Y)$	$\forall Z\ d(W,Y,Z)$	n
$d_7(Y,W)$	$c(Y) \rightarrow d_6(W,Y)$	n
$d_8(W)$	$\forall Y\ d_7(Y,W)$	n
$d_9(W)$	$c(W) \wedge d_8(W)$	n
d_{10}	$\exists W\ d_9(W)$	n
d_{11}	$d_5 \rightarrow d_{10}$	n
$d_{12}(W,U)$	$\forall V\ d(W,U,V)$	p
$d_{13}(U,W)$	$c(U) \rightarrow d_{12}(W,U)$	p
$d_{14}(W)$	$\forall U\ d_{13}(U,W)$	p
$d_{15}(W)$	$c(W) \wedge d_{14}(W)$	p
$d_{16}(Y,Z)$	$c(Y) \wedge h_2(Y,Z)$	p/n
$d_{17}(W,Y,Z)$	$h_3(W,Y,Z) \wedge o(W,g)$	p/n
$d_{18}(Y,Z,W)$	$d_{16}(Y,Z) \rightarrow d_{17}(W,Y,Z)$	p/n
$d_{19}(Y,Z)$	$c(Y) \wedge \neg h_2(Y,Z)$	p/n
$d_{20}(W,Y,Z)$	$h_3(W,Y,Z) \wedge o(W,b)$	p/n
$d_{21}(Y,Z,W)$	$d_{19}(Y,Z) \rightarrow d_{20}(W,Y,Z)$	p/n
$d_{22}(Y,Z,W)$	$d_{18}(Y,Z,W) \wedge d_{21}(Y,Z,W)$	p/n
$d_{23}(W)$	$\forall Y Z\ d_{22}(Y,Z,W)$	p/n
$d_{24}(W)$	$d_{15}(W) \rightarrow d_{23}(W)$	n
d_{25}	$\forall W\ d_{24}(W)$	n
$d_{26}(W)$	$c(W) \wedge d_{23}(W)$	p
$d_{27}(Y,W)$	$c(Y) \wedge h_3(W,Y,Y) \wedge o(W,g)$	p
$d_{28}(Y,W,V)$	$d_{27}(Y,W) \rightarrow \neg h_2(V,Y)$	n
$d_{29}(Y,W)$	$c(Y) \wedge h_3(W,Y,Y) \wedge o(W,b)$	p
$d_{30}(Y,V)$	$h_2(V,Y) \wedge o(V,b)$	n
$d_{31}(Y,W,V)$	$d_{29}(Y,W) \rightarrow d_{30}(Y,V)$	n
$d_{32}(Y,W,V)$	$d_{28}(Y,W,V) \wedge d_{31}(Y,W,V)$	n
$d_{33}(W,V)$	$\forall Y\ d_{32}(Y,W,V)$	n
$d_{34}(V,W)$	$c(V) \wedge d_{33}(W,V)$	n
$d_{35}(W)$	$\exists V\ d_{34}(V,W)$	n
$d_{36}(W)$	$d_{26}(W) \rightarrow d_{35}(W)$	n
d_{37}	$\forall W\ d_{36}(W)$	n
d_{38}	$d_{25} \wedge d_{37}$	n
d_{39}	$d_{11} \wedge d_{38}$	n
d_{40}	$d_{39} \rightarrow \neg d_5$	p

Table 2: The labels and subformulae.