The Field of Automated Reasoning*

Larry Wos

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439-4801

Abstract

The term *automated reasoning* (first introduced in 1980) accurately describes the objective of the field, the automation of logical reasoning. This article introduces scientists to the field and then briefly describes the papers found in this special issue.

Key words. Automated reasoning, inference rules, logical reasoning, redundancy, strategy.

1. Introduction and Background

This issue is devoted entirely to the field of automated reasoning. Although the field is equally concerned with theory, implementation, and application, the papers in this issue emphasize the latter two aspects. The term *automated reasoning* was first coined in 1980 to reflect the broadening of its predecessor, automated theorem proving. Indeed, rather than focusing almost exclusively on proving theorems from mathematics and logic, by 1980 the applications had expanded to include program verification, circuit design and validation, hypothesis testing, conjecture formulation, and puzzle solving.

The name automated reasoning accurately suggests the main objective of the field: the design of computer programs that reason logically, drawing conclusions that follow inevitably from the hypotheses supplied by the user of such a program. In other words, if an automated reasoning program is sufficiently free of bugs, the reasoning it applies is sound. Although not the concern in any of the papers presented here, some automated reasoning programs also permit the user to ask for probabilistic reasoning, usually by a suitable modification of those statements of the input that characterize the question or problem under study.

Of course, from a practical perspective, merely designing a computer program that reasons logically offers little, unless the program can be used to answer questions and solve problems that interest people outside of automated reasoning. The papers in this issue strongly suggest that the field of automated reasoning has in fact arrived. Indeed, among other successes, reasoning programs have been used to answer open questions from mathematics and logic and to validate a chip that is now in use. The achievements are most impressive when one realizes that eminent logicians such as Lukasiewicz asserted in 1948 that a formalized proof cannot be ‘‘discovered mechanically’’ but can only be ‘‘checked mechanically’’ [2].

Clearly, one could not have expected foresight sufficient to predict what would occur beginning in the early 1960s and culminating in the early 1990s with the availability of powerful and versatile automated reasoning programs. However, and in contrast to the understandable lack of foresight exhibited in 1948, even today various researchers are more than skeptical (see, especially, [1]) regarding the use of a computer program to produce proofs in which numerical calculation is not the crux, proofs such as one finds in group theory, for example. A charitable explanation for the current skepticism—and worse—might be inadequate dissemination of information. The papers of this issue provide

*This work was supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

substantial evidence refuting the naysayers regarding the possible automation of logical reasoning (see also [4]).

The source of the impressive power offered by the better reasoning programs rests in part with the use of *strategy* [3,7]. Some programs offer the user a choice of strategies to restrict the reasoning, preventing certain paths of inquiry from being explored. Of a quite different character, some programs offer the user a choice of strategies to direct the reasoning, providing a means for the program to decide where next to focus its attention. Evidence strongly suggests that, for a reasoning program to be of substantial use for answering diverse and deep questions, its menu must include a number of strategies from each class.

For some programs, the basis of power also rests with mechanisms to control redundancy [6]. To be effective, a reasoning program must cope with two types of redundancy, especially evident when conclusions are retained, but an obstacle even when they are not retained [5]. The more obvious type of redundancy concerns drawing the same conclusion repeatedly, from different paths of reasoning, and drawing pairs of conclusions one of which is less general than is the other. Some programs offer the procedure *subsumption* to cope with this type of redundancy by purging less general information when more general information is present. The less obvious form of redundancy, which might be termed semantic redundancy, can be illustrated with two examples. For the first example, in many cases, semantic redundancy is present when a program retains both the fact that $a+b = c$ and the fact that $0+(a+b) = c$. For the second example, ordinarily, semantic redundancy is present when the program retains the two equalities $(a+b)+c = d$ and $a+(b+c) = d$. To cope with this type of redundancy, some programs offer the procedure *demodulation*, a procedure that can be used to canonicalize and simplify expressions. For a thorough treatment of subsumption and of demodulation, see [3,7].

As for the language for presenting questions and problems, variants of first-order predicate calculus are frequently used. Regarding the reasoning itself—although some programs offer induction and some offer an approach based on complete sets of reductions—typically, the inference rules used to draw conclusions are tailored to the field of automated reasoning, rather than emulating the reasoning people apply. For example, in contrast to offering instantiation as an inference rule—permitting the program to deduce that $(yz)(yz) = e$ from the hypothesis that $xx = e$, where e is the identity of a group—inference rules such as *paramodulation* [3,7] are sometimes offered. Paramodulation generalizes the usual notion of equality substitution and is an excellent example of an inference rule that is both difficult for a person to apply and trivial for a computer program to apply. Indeed, one finds that, rather than emulating the mind of a researcher, many automated reasoning programs complement the researcher, often acting as an automated reasoning assistant.

The effectiveness of today's automated reasoning assistants is attested to by the various papers in this issue. Let us therefore turn now to a brief review of those papers.

2. Brief Review of the Papers

The papers in this special issue on automated reasoning fall loosely into two groups. In the first group, the papers focus on the use of automated reasoning programs to answer previously open questions in mathematics or logic. In the second group, the papers discuss in depth some specific automated reasoning programs. One paper, the last in this issue, spans both groups: The paper discusses the application of a new strategy and also presents an overview of OTTER, a powerful automated reasoning program used in several of the applications reported in this issue. To aid the reader, we now summarize the papers, following the order in which they appear.

2.1. K. Kunen — Groups of Exponent 4 and OTTER

In his paper ''The Shortest Single Axioms for Groups of Exponent 4'', K. Kunen successfully searches for single axioms for a particular variety of groups, those in which the fourth power of every element x is the identity e. Kunen uses the program OTTER to prove that three such axioms exist, eliminating the rest with the use of models. He also presents an improvement on B. H. Neumann's scheme for single axioms for varieties of groups.

2.2. R. Padmanabhan and W. McCune — Ternary Boolean Algebra and OTTER

In their paper ''Single Identities for Ternary Boolean Algebras'', R. Padmanabhan and W. McCune offer for ternary Boolean algebra the first known single axioms. They use a method of Padmanabhan to find a single axiom, then show how the automated reasoning program OTTER was used to obtain a simpler axiom.

2.3. R. Padmanabhan and W. McCune — Algebraic Geometry and OTTER

In their paper ''Automated Reasoning about Cubic Curves'', R. Padmanabhan and W. McCune study algebraic geometry by enhancing McCune's program OTTER with a new inference rule gL. They use the enhanced program to prove various incidence theorems on projective curves and without any reference to the geometry or the topology of the curves. Their application provides a new use for automated reasoning programs.

2.4. R. Boyer, M. Kaufmann, and J Moore — Nqthm and Verification

In their paper ''The Boyer-Moore Theorem Prover and Its Interactive Enhancement'', R. Boyer, M. Kaufmann, and J Moore describe the Boyer-Moore program Nqthm and its extension. These two programs, mainly used for verification of both hardware and software, appear to be among the best currently available for these applications. The authors introduce the logic in which theorems are proved and present a detailed example of how the programs can be used. To show the breadth of the two programs, the authors also give short descriptions of numerous and diverse applications.

2.5. S. Chou and X.-S. Gao — Theorems in Pascal Conics

S. Chou and X.-S. Gao, in their paper ''The Computer Searches for Pascal Conics'', discuss an approach to discovering new theorems in geometry, an approach that relies on numerical examples and that is based on Wu's method. Four theorems related to Pascal conics have been discovered, two of which may be new.

2.6. C. Brink, D. Gabbay, and H. J. Ohlbach — Automating Dualities in Mathematics and OTTER

Among the dualities that can occur between different theories in mathematics and in logic, C. Brink, D. Gabbay, and H. J. Ohlbach focus on structures and power structures in their paper Towards Automating Duality''. They show how properties of one theory can be automatically translated into corresponding properties of the other theory by using quantifier-elimination algorithms and a theorem-proving program relying on first-order predicate logic. A particular instance of the duality between structures and power structures is the duality between an axiom of a Hilbert-style specification of a logic and the corresponding semantic property. Their approach is illustrated with examples treated with the automated reasoning program OTTER.

2.7. D. Kapur and H. Zhang — RRL and Rewrite Rules

In their paper ''An Overview of Rewrite Rule Laboratory RRL'', D. Kapur and H. Zhang provide a brief historical account of the development of their program RRL. Among the automated reasoning programs based on the use of rewrite rules, their program may be the most powerful, as evidenced by its successful use to solve hard mathematical problems. The authors include an overview of the capabilities of RRL and discuss various applications.

2.8. J. Slaney, M. Fujita, and M. Stickel — Quasigroups and Program Comparisons

In their paper, ''Automated Reasoning and Exhaustive Search'', J. Slaney, M. Fujita, and M. Stickel study and compare the use of three reasoning programs in the context of solving problems in the theory of quasigroups. Each of the three programs successfully answered previously open questions in this area of mathematics. The authors also include a philosophical discussion focusing on proofs that are essentially computational.

2.9. L. Wos — Resonance Strategy, Shorter Proofs, and OTTER

The volume concludes with the paper ‘‘The Resonance Strategy’’, which defines and applies the resonance strategy discovered by L. Wos. The paper shows how the resonance strategy was used to find shorter proofs than were previously known and to prove theorems that were previously out of reach of an automated reasoning program. The successes (obtained with the aid of the automated reasoning program OTTER) are taken from group theory, Robbins algebra, and various logic calculi. Included in the paper is an introduction to the features of the program OTTER.

3. The Power of Automated Reasoning

The nine papers in this issue give a tantalizing taste of the power of today’s automated reasoning programs. Both special-purpose programs such as Nqthm and general-purpose programs such as OTTER and RRL are now successfully used to answer open questions and solve hard problems from mathematics, logic, and software and hardware verification. If these papers are indicative of the future, then the outlook is indeed bright, for access to a powerful automated reasoning assistant of the type described in this volume will play a key role in both research and applications of a wide variety.

References

1. Kolata, G., ‘‘Computers Still Can’t Do Beautiful Mathematics’’, *The New York Times*, Section E, p. 4 (14 July 1991).

2. Lukasiewicz, J., ‘‘The Shortest Axiom of the Implicational Propositional Calculus’’, *Proc. of the Royal Irish Academy 52A*, 3 (1948) 25-33.

3. Wos, L., *Automated Reasoning: 33 Basic Research Problems*, Prentice-Hall: Englewood Cliffs, N.J., 1987.

4. Wos, L., ‘‘Automated Reasoning Answers Open Questions’’, *Notices of the AMS*, 5, no. 1 (January 1993) 15-26.

5. Wos, L., and McCune, W., ‘‘Automated Theorem Proving and Logic Programming: A Natural Symbiosis’’, *Logic Programming*, 11, no. 1 (July 1991) 1-53.

6. Wos, L., Overbeek, R., and Lusk, E., ‘‘Subsumption, A Sometimes Undervalued Procedure’’, pp. 3-140 in *Festschrift for J. A. Robinson*, ed. J.-L. Lassez and Gordon Plotkin, MIT Press: Cambridge, Mass., 1991.

7. Wos, L., Overbeek, R., Lusk, E., and Boyle, J., *Automated Reasoning: Introduction and Applications*, 2nd ed., McGraw-Hill: New York, 1992.