

Automated Theorem Proving: Theory and Practice

A Review

Geoff Sutcliffe

Automated Theorem Proving (ATP) deals with the development of computer programs that show that some statement (the conjecture) is a logical consequence of a set of statements (the axioms and hypotheses). ATP systems are used in a wide variety of domains: A mathematician might use the axioms of group theory to prove the conjecture that groups of order two are commutative; a management consultant might formulate axioms that describe how organizations grow and interact and, from these axioms, prove that organizational death rates decrease with age; or a frustrated teenager might formulate the jumbled faces of a Rubik's cube as a conjecture and prove, from axioms that describe legal changes to the cube's configuration, that the cube can be rearranged to the solution state. All these tasks can be performed by an ATP system, given an appropriate formulation of the problem as axioms, hypotheses, and a conjecture. Most commonly, ATP systems are embedded as components of larger, more complex software systems, and in this context, the ATP systems are required to autonomously solve subproblems that are generated by the overall system. To build a useful ATP system, several issues have to carefully be considered, independently and in relation to each other, and addressed in a synergetic manner. These issues include the choice of logic that will be used to represent the problems, the calculus that will be used for deduction, the programming language that will be used to write the ATP system, the data structures that will be used to hold the statements of the problem and the

statements deduced by the system, the overall scheme for controlling the deduction steps of the system, and the heuristics that will control the fine-grained aspects of the deduction steps (the heuristics are most likely to determine the success or failure of an ATP system because they most directly

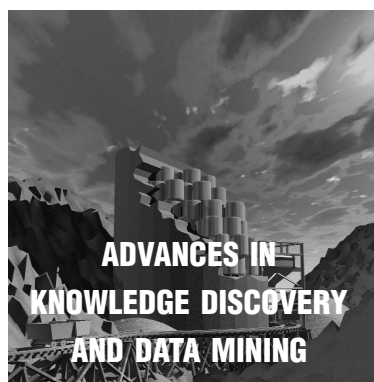
Automated Theorem Proving: Theory and Practice, Monty Newborn, Berlin, Springer-Verlag, 231 pp., \$54.95. ISBN 0-387-95075-3.

control the system's search for a solution). Current state-of-the-art ATP systems, such as E, VAMPIRE, E-SETHEO, and WALDMEISTER, have been developed with the benefit of years of experimentation and effort (information about these systems can easily be found on the World Wide Web). For a newcomer to the field, it is a daunting, if not seemingly impossible, task to start on the road to building a decent ATP system. It is almost imperative that a budding ATP system developer should start with the benefit of previous experience. Monty Newborn's book, *Automated Theorem Proving: Theory and Practice*, can contribute to this learning process.

Monty Newborn is a professor of computer science at McGill University, Canada. Newborn is probably better known for his involvement with computer chess than with ATP. In particular, he is the long-standing chairperson of the ACM Computer Chess Committee, which organized the famous 1997 match in which IBM's DEEP BLUE program defeated the human

world champion, Gary Kasparov. It is Newborn's background in the search issues of computer chess that led to his more recent interest in ATP. His book provides an introduction to some of the basic logic, calculi, heuristics, and practicalities of first-order ATP. Rather than working rigorously through the theory of mathematical logic, the book focuses on just the necessary foundations for understanding how first-order ATP systems operate and links these foundations to the implementation of two ATP systems, HERBY and THEO. The book comes with a CD containing the source code for HERBY and THEO and several suites of ATP problems for HERBY and THEO to attempt.¹ Thus, the reader is able to experiment as both a user and a developer of ATP systems. The provision of the ATP systems and problems sets this book apart from most other introductory books on ATP that provide a more in-depth treatment of the theory but fail to get readers over the initial hurdles of using an ATP system (a notable exception is Wos et al.'s book, *Automated Reasoning: Introduction and Applications* [McGraw-Hill, 1992], which comes with the well-known ATP system OTTER). Newborn's book is suitable as an introduction to ATP for undergraduate university students and independent, interested readers.

After an introductory chapter explaining the structure of the book and software installation, chapters 2 to 4 introduce first-order logic (in the syntax used by HERBY and THEO) and the basic mechanics and semantics of resolution-based ATP. Chapters 5 and 6 then provide the underlying theory and describe the calculi for the two ATP systems, chapter 5 corresponding to HERBY and chapter 6 corresponding to THEO. The architecture, use, and implementation of HERBY are described in chapters 7, 8, and 11, respectively, and the same information is provided for THEO in chapters 9, 10, and 12. The last chapter steps aside to briefly discuss the CADE ATP System Competition (CADE [the Conference on Automated Deduction] is the major forum for the presentation of new research in all aspects of automated deduction). Variants of HERBY and THEO participated in the competitions in 1997 and 1998.



**Usama M. Fayyad,
Gregory Piatetsky-Shapiro,
Padhraic Smyth, and
Ramasamy Uthurusamy,
editors**

ISBN 0-262-56097-6 632 pp., index.

The AAAI Press

Distributed by The MIT Press

Massachusetts Institute of Technology,
5 Cambridge Center Cambridge,
Massachusetts 02142

To order, call toll free:

(800) 356-0343 or (617) 625-8569.

MasterCard and VISA accepted.

Each chapter ends with a set of exercises that are useful for testing the reader's understanding of the chapter material.

Many first-order ATP systems and calculi, including those described in this book, use the clause normal form (CNF) of first-order logic statements. A procedure for converting a problem in first-order form to CNF is described in chapter 3, along with instructions for using the conversion program supplied on the CD. The use of CNF is largely motivated by the resolution inference rule, which generalizes the intuitively understandable modus ponens rule: If A is true, and it is true that A implies B , then B is true. THEO is a resolution-based ATP system. Chapter 4 describes the binary-resolution and factoring inference rules, which, in combination, implement the full-resolution inference rule. Chapter 4 also introduces the very important subsumption rule, which is used to eliminate redundant clauses that might be inferred (although the book admits that HERBY and THEO implement only a weakened form of subsumption, called s-subsumption).

The underpinning for resolution-

based ATP is Herbrand's theorem, which shows that logical consequence can be established by considering instances of a set of clauses. The semantic tree calculus implemented in HERBY is a direct application of Herbrand's theorem. Herbrand's theorem and the semantic tree calculus are presented in chapter 5. The semantic tree calculus is appropriate for an introductory book on ATP because it provides a way to see the immediate consequences of Herbrand's theorem in action. It also provides a clear framework in which to illustrate some issues of heuristic control in an ATP system, as is done in chapter 7. Unfortunately, the semantic tree calculus fails to benefit from the abstraction inherent in the resolution inference rule, thus making it hard to develop a high-performance ATP system based on this calculus. The reader should note that the examples in chapter 5 all use a finite Herbrand universe and that a semantic tree can obviously get a lot deeper when the Herbrand universe is infinite, as exemplified by the example in chapter 8.

The resolution calculus, the basis for THEO, is described in chapter 6 with some nice examples. The explanation of how a resolution proof can be extracted from a closed semantic tree provides an excellent link to the preceding chapter and Herbrand's theorem. Linear resolution, a refinement of the resolution calculus, is introduced next. Linear resolution is significant as the basis for the PROLOG programming language (where the format of the problem is restricted to Horn clauses, providing completeness for linear-input resolution) and its use in various high-performance ATP systems, for example, PTPP, PROTEIN, and METEOR. Linear resolution also parallels the tableau calculus used successfully by the E-SETHEO system. Interestingly, Newborn chooses to extend linear resolution to linear-merge resolution. Historically, resolution with merging was developed separately but around the same time (late 1960s) as linear resolution. The link between them was noticed later, and several ATP systems exploited the link until the late 1970s when interest in developing their combination seemed to disappear. The THEO system is thus uniquely interesting

among the linear resolution-based systems available now. Chapter 9, as well as describing the architecture of THEO, describes some strategies that can be used by a broad range of resolution-based systems to improve performance. Some of these strategies come at the cost of completeness; that is, if such a strategy is used, then some theorems can no longer be proved. However, it is well known that almost all contemporary high-performance ATP systems use incomplete strategies (sometimes even unintentionally).

The chapters describing the implementations of HERBY and THEO will be too technically messy for a reader who is primarily interested in the deductive issues and will also be inadequate for a programmer who wants to understand the internal workings of the systems (although the programmer does have the option of examining the source code provided on the CD). These chapters, however, will give all readers a feel for the complexity of source code and data structures that are required to implement an ATP system in an imperative programming language such as C.

Overall, Newborn's book meets the needs of its intended audience as a straightforward and practical introduction to ATP. If it whets your appetite, you can take advantage of the bibliography of appropriate further material. If not, you'll walk away with at least a basic understanding of ATP.

Note

1. The first-order problems in the WFF directory on the CD use the keyword theorem to mark the conjecture to be proved, but the compile program, for converting problems to clause normal form, expects to find the keyword conclusion. To use the compile program, it is necessary to edit either the note.WFF files or the compile source code file lexwff.c in the COMPSC directory.

Geoff Sutcliffe is a faculty member in the Department of Computer Science at the University of Miami. His research focuses on the evaluation and appropriate application of automated theorem-proving (ATP) systems, including the development of parallel and distributed ATP systems, and easy-to-use ATP system interfaces. His e-mail address is geoff@cs.miami.edu.