The Review of Modern Logic
Volume 10 Numbers 3 & 4 (March 2005–May 2007) [Issue 32], pp. 213–217.

Ricardo Caferra, Alexander Leitsch, and Nicholas Peltier
Automated Model Building
Applied Logic Series, Vol. 31
Dordrecht/Boston/London: Kluwer Academic Publishers, 2004
xi + 341 pp. ISBN 1402026528

REVIEW

VLADIK KREINOVICH

Automated deduction was traditionally concentrated on trying to find out whether a given statement A can be deduced from a given theory T. Some automated deduction techniques start with the theory and try to get as many conclusions as possible—in the hope that the query A will be among these statements. Other automated deduction techniques—including resolution, probably the most well known of these techniques—start with adding the negation $\neg A$ to the theory, and proceed by deducing as many conclusions as possible in the hope that one of these conclusions will be a contradiction; if T is not compatible with $\neg A$, this means that T implies A.

Sometimes, the conclusion is that the formula A cannot be deduced from the theory T. In many practical applications, it is desirable not just to show that A cannot be deduced from T, but also to produce a model explaining a situation when T holds but A does not hold. For example, one possible application of automated deduction is to prove that a robot is safe. In this case, T is a theory describing the motion of a robot under the given design, and A is a statement describing that this robot is safe (*e.g.*, that a mobile robot never leaves the safe zone where it is supposed to stay). In this example, if we prove that the robot is not safe, *i.e.*, that A cannot be deduced from T, then the robot designer would like not only to know the *fact* that the design is un-safe, but also to see an explicit detailed *example* where this design fails—*i.e.*, a model of $T \cup \{\neg A\}$. Such a specific example is, in general, much more useful in correcting the design than simply an indication that the design is wrong.

In pure mathematics, the need for models is also well known. For example, in geometry, it was an interesting theoretical achievement to

prove that Euclid's Vth postulate cannot be deduced from the other axioms of elementary geometry. However, real applications of this research endeavor started only when researchers formulated and analyzed explicit non-Euclidean geometries—which led not only to interesting new mathematics, but also to a breakthrough in our understanding of space-time (that came with Einstein's General Relativity).

Ideally, instead of building *a* model, it is desirable to design a general scheme for generating *all* possible models—*i.e.*, in effect, building a universal parametric model, that covers all possible models when we change the values of the parameters. For example, in the robot design, it would be excellent to learn not only of a situation where this robot design is unsafe, but of all such situations. Similarly, in geometry, the application of non-Euclidean geometries was greatly enhanced when mathematicians moved from describing individual models of non-Euclidean geometry (such as Lobachevsky space) to Riemann's description of general non-Euclidean geometries—a description that made it possible for Einstein to find classes of space-time models which are consistent with observations.

All these examples show that it is often very important to be able to automatically build models of consistent theories.

At first glance, automated model building may seem to be a completely different problem from automated deduction:

- In automated deduction, our goal is to deduce A as fast as possible; thus, we try to deduce the truth values of as few additional statements as possible.
- In model building, conversely, we are interested in assigning truth values to as many statements as possible.

However, in practice, the difference is not as large as it may seem at first glance. Yes, in the ideal world, the search for a proof of A should be guided and focused, and it should result in proving only statements in a short chain from the axioms of T to the desired statement A. However, in real life, the search problems are known to be computationally difficult—even with linear-size proofs, the search for these proofs requires, in many cases, at least exponential time. Thus, no matter how much we try to focus our search, the search for a proof of a given statement usually results in proving a lot of additional statements unrelated to A. In many situations, we deduce so many such statements that we are, in effect, halfway towards building a full *model* for the original theory. As a result of this connection, many automated model building techniques are based on the corresponding automated deduction ones.

The book under review is the first book on automated model building. It mainly describes the approaches developed by the authors, but it also provides an overview (in Chapter 6) of other existing approaches and techniques. From the technical viewpoint, the authors' main objective is to teach the reader new methods and techniques; however, the authors also have a more fundamental objective that makes this book really interesting and exciting: to convince the reader that the existing automated model building systems are actually capable of automatically building non-trivial models.

How can we build a model? A model means that we assign truth values to all possible logical statements. Of course, to describe a model, it is sufficient to assign truth values to all *atomic* statements—then, the logical rules will enable us to automatically assign truth values to composite logical statements as well. Therefore, one possible approach to model building is an *enumeration* (*non-symbolic*) approach in which we enumerate all atomic formulas and assign them truth values one-by-one.

It is well known that it is often more efficient to first assign truth values to complex formulas—*e.g.*, if we succeed in deducing $\forall x\, P(x)$, then we can immediately assign "true" to all the atomic formulas of the type $P(t)$, without the need to individually analyze all of them. The resulting *deduction-based* (*symbolic*) approach is especially important for infinite models—where there are infinitely many atomic formulas and thus, it is impossible to enumerate all of them. (In practice, a proper combination of these two approaches is often the most efficient way to model building.)

The book starts with the introduction to the problem (Chapter 1), and a brief overview of the basic technical ideas like resolution that will be used in the main part of the book (Chapter 2). Chapter 3 explains how resolution techniques (especially the ideas of hyper-resolution that speed up resolution) can be extended from their traditional use in automated deduction to their new use in automated model building.

Chapter 4 describes automated model building techniques motivated by constraints. In constraint satisfaction, usually the objective is either to show that the constraints are inconsistent or to find a design or an object that satisfies all the given constraints. Because of this objective, algorithms for solving such problems simultaneously try to find a solution and to prove that the solution is not possible—*i.e.*, simultaneously search for refutations and models. The authors generalize this idea from the usual number-based constraints to the most general logical constraints corresponding to general theories. To be able to efficiently look both for models and for refutations, the authors

add new *dis-inference* rules to the standard inference rules underlying automated deduction.

For example, constraint techniques corresponding to strict inequalities can be naturally extended to *dis-equation* constraints $t_1 \neq t_2$ between the terms t_1 and t_2. Such dis-equations can be very useful for describing a model: *e.g.*, if we have a system of equations, then its most general solution can be described by a most general unifier (that provides unification to all the pairs). In the presence of negation, such representation is no longer possible: *e.g.*, the solutions to the formula $x = f(u, v) \wedge u \neq v$ cannot be described by a finite disjunction of unification problems, we need to explicitly represent dis-equations.

The authors show that with the new dis-inference rules and new substitution rules which are only valid under constraints, we can often efficiently describe models for complex theories—theories in which some axioms contain negations, disjunctions, and quantifiers applied to equations.

Once we have built a model, *i.e.*, once we have selected a sequence of formulas that, *in principle*, uniquely determine the truth values of all the formulas, it is then necessary to develop an efficient procedure that would tell us, for each given formula A, whether this formula is true or false in this model. This topic is discussed in Chapter 5.

Chapter 6 describes the specifics of building *finite* models. At first glance, building a finite model may seem like an easier task than building an infinite model—after all, for each model size, there are only finitely many possible truth assignments, so, in principle, we can try all of them and find the model by exhaustive search. In practice, however, building a finite model is more difficult than building an infinite one. One of the main reasons for this difficulty is that many known properties of models, properties which are efficiently used in automated deduction, are no longer valid if we restrict ourselves to finite models only. One such property is a compactness theorem: if every subset of a theory is consistent, then the theory as a whole is also consistent—*i.e.*, has a model. This compactness property does not hold for finite models: *e.g.*, the sequence of infinitely many statements

$$\exists x_1 \ldots \exists x_n (x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \ldots \wedge x_{n-1} \neq x_n)$$

(meaning that there are at least n different objects) does not have a finite model, but each finite subset has one. Finite models is where most of the current work in automated model building was done. Because of this, in the finite-model Chapter 6, the authors overview other approaches to automated model building and their known applications—in AI, in mathematics, and in operations research.

The authors have made a lot of effort to make this book generally accessible. The book starts with the preliminaries that go as far as explaining the notion of a clause, the basic ideas behind resolution and natural deduction, *etc.* Of course, the authors' brief introduction cannot completely replace serious textbooks; however, it enables the readers who are not very familiar with automated deduction to be able to follow all the technical details—and those who have forgotten some of the automated deduction ideas have a chance to recall them. In spite of this introduction, in my opinion, this book is mostly beneficial to readers who are somewhat familiar with automated deduction.

Motivated readers who want to learn more about automated model building—*e.g.*, those who want to creatively apply the authors' techniques to their practical problems like robotics—are also strongly encouraged to read this book. For these readers, it will not be easy reading, but first, they can always use an AI or automated deduction textbook if necessary; second, they can consult technical books and papers (rarely an easy reading); and third, this is the only book so far, so interested readers do not really have a choice.

It is also, in my opinion, a very good book for a special topics course.

In short, I would encourage readers to study this book. The area is new, there are many interesting open problems and potential improvements. Because of this newness, motivated readers have a unique opportunity not only to learn interesting ideas but also, hopefully, to meaningfully contribute to this area—by providing new applications, new ideas, and new results.

Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA

E-mail address: vladik@utep.edu