Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

"Decision Procedures: An Algorithmic Point of View," by Daniel Kroening and Ofer Strichman, Springer-Verlag, 2008

Clark Barrett

Received: date / Accepted: date

The topic of this book is decision procedures for first-order theories, a research area now typically referred to as *Satisfiability Modulo Theories (SMT)*.[1] The book is important if for no other reason than because it is one of the first to capture the essential concepts of SMT in a book.[2] More importantly, most of the content is still relevant to those wishing to understand the area today.

The book is structured as follows. After a high-level introductory chapter, and a chapter on propositional logic and Boolean satisfiability (SAT), the bulk of the content (chapters 3 through 8) deals with decision procedures for specific first-order theories: equality with uninterpreted functions (chapters 3 and 4), linear arithmetic (chapter 5), bit vectors (chapter 6), arrays (chapter 7), and pointer logic (chapter 8). Chapter 9 covers quantifiers (the other chapters focus on quantifier-free formulas); chapter 10 covers theory combination; and chapter 11 describes how to integrate Boolean satisfiability with theory reasoning. In addition, each chapter includes a challenging set of problem exercises. And every chapter except for the first and third concludes with a short but informative "Bibliographic Notes" section which gives some historical context and additional reading for the chapter. Finally, there are two appendices: the first is a single page which mentions the SMT-LIB initiative and some other SMT community activities; and the second is a description of a software library (available online) developed for the book. In this review, I

C. Barrett
251 Mercer Street, New York, NY 10012
E-mail: barrett@cs.nyu.edu

[1] At the time the book was published, the name SMT was not yet as established as it is now.

[2] A related book, "The Calculus of Computation" by Bradley and Manna [1] focuses more on program verification whereas this book covers more theories (such as the theory of bit vectors, missing from Bradley and Manna's book) and includes SMT-specific topics like the *DPLL(T)* framework for combining SAT with theory reasoning. Bradley and Manna's treatment of logic is more formal, however.

will describe each of the chapters briefly and then conclude with some general remarks about the book.

Chapter 1 is an introduction and begins with an important point about the book's perspective: the focus is "on algorithms rather than mathematical logic." Though there is a section on "the theoretical point of view", for the most part, formal foundations are only lightly touched on here (and throughout the book), while algorithms are described in detail. The first chapter defines important terms like satisfiability, decidability, soundness, and completeness. It also has a nice overview of basic normal forms (negation normal form, disjunctive normal form, conjunctive normal form), and Tseitin's encoding for converting arbitrary Boolean formulas into conjunctive normal form.

Chapter 2 covers propositional logic with a focus on Boolean satisfiability (SAT) and binary decision diagrams (BDDs). The section on SAT solvers is quite detailed and includes most of the fundamental concepts and techniques used in modern SAT solvers (so-called conflict-driven clause-learning or CDCL solvers). The section on BDDs is, similarly, an excellent introductory tutorial on the subject. Both sections include helpful diagrams and examples, and the problems at the end of the chapter provide further insight and the opportunity to test one's understanding (though some are quite difficult).

Chapters 3 and 4 cover equality logic and uninterpreted functions. Formally, this is equivalent to satisfiability of quantifier-free first-order logic with equality over a signature with arbitrarily many function symbols. The authors focus on a two-stage method for solving such formulas: first, remove the function symbols; second, reduce the resulting equalities to a SAT problem. Two classic methods are discussed for removing function symbols: Ackermann's method and Bryant's method, and chapter 3 provides a good explanation of each. Most of chapter 4 is concerned with algorithms for reducing formulas with equalities to SAT using graph-based methods. This approach is covered quite thoroughly.

An alternative procedure for equality logic with uninterpreted functions is to reason directly about function symbols and equality together using an algorithm for congruence closure, and, in fact, this is the dominant approach used by modern SMT solvers. Section 4.1 presents one algorithm for congruence closure based on [4]. However, the presentation is brief and, while a more thorough set of references on congruence closure can be found in the bibliography of chapter 4, it would have been nice to see more details on congruence closure discussed in the text of the chapter, especially as this is an important and commonly-used algorithm.

Chapter 5 covers algorithms for reasoning about linear arithmetic (both real and integer) and is a very nice synopsis of the subject. First up is a description of a variant of the simplex algorithm for real arithmetic which is the basis for most implementations in current SMT solvers. Next is a discussion of how to extend this approach to handle integer reasoning using branch-and-bound methods and cutting planes, including a detailed explanation of Gomory cuts. This is followed by a discussion of Fourier-Motzkin variable elimination, and then a detailed description of the omega test, a complete algorithm for

linear integer reasoning which includes a generally useful algorithm for solving linear equations over integers. There is a short section on simplifications that can aid in solving arithmetic problems and finally a brief description of difference logic, a special case in which all formulas are of the form $x - y \bowtie c$ where $\bowtie \in \{<, \leq\}$ and c is a constant. A decision procedure is given via reduction to finding negative cycles in a weighted directed graph. This last section is the only one that seems a bit thin. Generally, this chapter is an excellent tutorial and reference on decision procedures for arithmetic.

Chapter 6 covers the theory of bit-vectors, focusing on solving bit-vector constraints by flattening (also called bit-blasting) them into a pure SAT problem. One of the highlights of the chapter is a concise description of efficient flattening encodings for a fairly complete set of bit-vector operators, making it a nice reference for anyone needing to do bit-vector to SAT encodings. Also briefly covered is "incremental flattening," an abstraction-refinement algorithm in which expensive operators are either treated as variables or uninterpreted functions, in both cases the idea being to delay flattening them as long as possible. There is also a brief section on how to use linear arithmetic solvers to solve bit-vector arithmetic constraints as well as a section on solving fixed-point (as opposed to floating point) constraints via a reduction to SAT. In all, this is another excellent chapter providing a sound tutorial and reference, this time for bit-level reasoning. The only note to add is that this is a very active research area and so the material in the chapter and bibliography are necessarily incomplete and only provide a starting point for understanding recent and current work in the area.

Chapter 7 is a very short chapter about the theory of arrays, where an array is essentially an updatable function. The chapter focuses on a decidable theory fragment identified by Aaron Bradley in 2006 which allows quantifiers but has syntactic restrictions on formulas. This is certainly an important result about arrays. However, what is (surprisingly) missing is a discussion of how to decide arbitrary quantifier-free formulas involving arrays. To find this information, the interested reader will have to consult the references in the bibliography (or more recent work such as [3]).

Chapter 8 discusses the topic of reasoning about pointers. The approach taken is to model memory as a single array and pointers as indices into the array. This model is used to translate formulas in a simple pointer logic into formulas using arithmetic and arrays. Several extensions are covered including: using axioms to further constrain the memory model, modeling structure types (records), and modeling heap-allocated data structures. Several optimizations are also mentioned such as: abstracting terms to variables (called pure variables) when it is clear that the terms can take on any value; and using several arrays instead of a single array to model memory. The chapter concludes with a discussion of rule-based decision procedures for reasoning about reachability in linked data structures. Overall, this is a nice introduction to reasoning about pointers and reachability using a basic but clear model. Some of the more sophisticated techniques for modeling and reasoning about pointers (e.g. Burstall's memory model) are referenced in the bibliographic notes.

Chapter 9 introduces quantifiers and quantifier elimination with an emphasis on quantified Boolean formulas (QBF). It gives examples of QBF problems and gives algorithms for QBF based on both quantifier elimination and binary search. It also describes how Fourier-Motzkin elimination can be used for quantifier elimination in linear real arithmetic. There is obviously much more that could be said about reasoning with quantifiers, but it is considered (for the most part rightly so) beyond the scope of this book. One thing I would have liked to see is a discussion of heuristic instantiation as a technique for proving quantified formulas, something found in many SMT solvers.

Chapter 10 gives an introduction to the theory combination problem: given decision procedures for several individual theories, how can these be used to check formulas in the combined theory. The Nelson-Oppen technique for theory combination is presented, with algorithms for both the convex and the non-convex case, and a proof of correctness for the former. Also mentioned is the abstract Nelson-Oppen procedure which unifies the two cases by introducing the idea of an arrangement of variables.

Chapter 11 explains how a SAT solver and theory-specific decision procedures can work together to determine the satisfiability of formulas with both theory reasoning and non-trivial Boolean structure. This is known as the "lazy" approach to SMT and the standard framework for implementing it is called DPLL(T). The authors do a good job of explaining this framework in considerable detail, including covering several important implementation details such as theory propagation, early conflict detection, and the importance of strong lemmas. The chapter concludes with a discussion of how proof objects can be produced for both lazy and eager frameworks.

Overall, the book touches, at least briefly, on nearly every important topic related to modern SMT solving, a significant accomplishment. As mentioned above, the book's approach is to emphasize algorithms and examples rather than theoretical foundations. This approach works surprisingly well and has the benefit of making the book more accessible to those without a strong background in mathematical logic. Those looking for a presentation rooted in logical foundations, however, will probably want to supplement their reading by looking at the references described in the bibliographic notes.

The book has an accompanying website [2] with additional information including presentation slides and software downloads. There is also a link to *Errata* which should be consulted as part of any serious attempt to study the material in the book, as a number of significant errors are corrected there.

References

1. Bradley, A.R., Manna, Z.: The Calculus of Computation: Decision Procedures with Applications to Verification. Springer Berlin Heidelberg (2007)
2. Kroening, D., Strichman, O.: URL `http://www.decision-procedures.org/`
3. de Moura, L., Bjørner, N.: Generalized, efficient array decision procedures. In: Formal Methods in Computer-Aided Design, 2009, pp. 45–52. IEEE (2009)
4. Shostak, R.E.: An algorithm for reasoning about equality. Communications of the ACM **21**(7), 583–585 (1978)