CS402 Introduction to Logic for Computer Science

Critical Review

Satisfiability Modulo Theory and SMT solvers

20160042 Inyong Koo

1 Introduction

Satisfiability modulo theories (SMT) problem is a decision problem for logical formulas with respect to combinations of background theories such as arithmetic, bit-vectors, arrays, and uninterpreted functions. The most well-known constraint satisfaction problem is propositional satisfiability, or SAT, aiming to decide whether a formula over Boolean variables, formed using logical connectives, can be made true by choosing true/false values for its variables.

A **SMT solver** reports whether a formula is satisfiable, and if so, may provide a model of this satisfaction. SMT solvers are useful for verification (assertion checking), extended static checking, and software testing. Moreover, recent enhancements of SMT solvers enabled their use in diverse applications including scheduling, program synthesis, and run-time analysis [1].

Since the assignment specification has restricted the length of the review to be at most 3 pages (excluding bibilography), this review does not contain full details of the concepts, but rather introduces what the concept is about and where you can find more information about the topic.

We will first look into various decision procedures for individual theories, starting from familiar DPLL algorithm (Section 2). Next, we will address the Nelson-Oppen combination method to understand how the theory solvers can be combined in order to decide satisfiability of formulas involving multiple theories (Section 3). Lastly, we will conclude this review by addressing some recent approaches in SMT problem (Section 4).

2 Various Theory Solvers

The *Davis-Putnam-Logemann-Loveland* (DPLL) algorithm decides a propositional formula in CNF by using three main operations: decision, propagation, and backtracking. Its performance can be considerably increased by applying clause learning, yielding constraint-driven clause learning (CDCL) algorithms [2]. These famous SAT solving approaches are immensely important, since a SMT instance can be viewed as a generalization of a Boolean SAT instance in which various sets of variables are replaced by predicates from a variety of underlying theories. In fact, most SMT solvers follow the DPLL(T) architecture: combining a CDCL satisfiability solver and a decision procedure for conjunctions of propositions from theory T [3].

The *simplex algorithm* and its variants are typical theory solver for linear real arithmetic (LRA) [4]. Most SMT solvers implement the simplex algorithm using rational arithmetic, performed using machine integers (numerator, denominator). However, this approach is very inefficient in rare cases where the solver goes a lot into extended precision. And thus the linear programming in floating-point arithmetic was proposed [5].

In the case of linear integer arithmetic (LIA), decision procedure also follows the DPLL(T) scheme, including traditional *branch-and-bound* and *Gomory cuts* approaches [6] and their variants. An alternative to linear programming plus branching and/or cuts is Pugh's Omega test [7].

Decision procedures for other theories, such as bit vectors, arrays, and quantified formulas are introduced with sufficient details on Kroening's book (e-copy available from KAIST library) [8].

3 Nelson-Oppen Combination Method

In Section 2, we addressed decision procedures for theories that each formalize just one data type. But in many cases, we need to decide satisfiability of formulas involving multiple theories. Using the **Nelson-Oppen combination method**, we can construct decision procedures for union theroies from decision procedures for individual theories.

Nelson-Oppen combination method imposes following restrictions.

i) The two combining theories T_1, T_2 should be quantifier-free.

ii) The signatures of each theories Σ_1 and Σ_2 can only share equality. $\Sigma_1 \cap \Sigma_2 = \{=\}$

iii) Theories T_1 and T_2 must be *stably infinite*[1].

The first phase of Nelson-Oppen method is purification. The goal of purification is to seperate formula F in $T_1 \cup T_2$ into formulas F_1 and F_2 such that F_i belongs only to T_i. (F_i is "pure") where $F_1 \wedge F_2$ is equisatisfiable as F. This can be easily done by finding a pure subterm t, replacing it with a new variable v, adding the equation $v = t$ to the set, and then repeating this process until all literals are pure.

After purification, we seperated $(\Sigma_1 \cup \Sigma_2)$-formula F into two formulae, Σ_1-formula F_1, and Σ_2-formula F_2. F_1 and F_2 are linked by a set of shared variables V. We need to get the component satisfiability procedures to agree on the values assigned to the shared variables.

The next phase can be done in several approach.

i) Guess-and-check method

Guess an assignment of V, a set $arr(V)$ of equations and disequations encoding an equivalence relation on over V. If each satisfiability procedure finds its respective input $F_i \cup a(V)$ satisfiable, report the original set F to be satisfiable. Otherwise, try different guess and check. If no suitable assignment exists, report F to be unsatisfiable.

ii) Optimized incremental construction

Unfortunately, the number of equivallence relations (possible guess) increases significantly with the number of shared variables. The guess-and-check method is impractical.

However, there is no need to guess the entire equivalence relation at once; instead, we can construct it incrementally, with various optimization methods as we did in DPLL.

iii) Equality propagation

Methods above were non-deterministic. The deterministic version of Phase 2 asks the decision procedures P_1 and P_2 to propagate information in the form of new equalities. We may deduce and propagate, from one component decision procedure to the other, disjunctions of shared equalities entailed by F_1 or F_2. This is particularly effective when T_1 and T_2 are both *convex*[2] over the sorts they share.

If both T_1 and T_2 are convex, it is enough for completeness to consider only individual entailed equalities. If either F_1 or F_2 is unsat, F is unsatisfiable. Note that it doesn't mean that F_1 and F_2 being satisfiable implies F to be satisfiable. In such case, theories have to exchange all implied equalities.

With non-convex theories, or convex theories for which computing entailed equalities is expensive, another approach is to check the T_i-satisfiability of F_i alone and once a model A_i is found, make the optimistic assumption that $F_j \cup a(V)$ is T_j-satisfiable, where $j \neq i$ and $a(V)$ is the assignment of V induced by A_i. If $F_j \cup a(V)$ is unsatisfiable because of some of the literals in $a(V)$, a new model for F_i with different truth values for those literals must be found. This heuristic approach is highly effective in practice.

I referred to [9, 10, 12, 11] to understand Nelson-Oppen methods.

[1] A theory T with signature Σ is stably infinite if for every quantifier-free Σ-formula F is T-satisfiable, then there exists some T-interpretation that satisfies F and has a domain of infinite cardinality.

[2] A Theory T is called convex if for every conjunctive formula F:
If $F \Rightarrow \vee_{i=1}^{n} x_i = y_i$ for finite n, then $F \Rightarrow x_i = y_i$ for some $i \in [1, n]$

4 Remarks

So far, we've been addressing decidability of quantifier-free theories. There are several approaches to handle SMT problems beyond quantifier-free theories. One approach is to eliminate quantifiers by substituting expressions into the quantified variables. Examples of substitution-based methods include Cooper's [13] for LIA, and Loos and Weisfpenning's [14] methods for LRA. Other approaches, such as quantifier elimination by projection, instantiation heuristics, and Craig interpolation are introduced on [3].

SMT solving problem involves the most fundamental areas of computer science and symbolic logic. It is an ongoing topic; There are efforts to support standrd library (SMT_LIB) for the SMT solvers, and new improved SMT solvers (such as z3 SMT solver [15]) are emerging from the annual competitions for SAT(`http://www.satcompetition.org`) and SMT (`http://www.smtcomp.org`).

I was able to learn some decision procedures for diverse theories, not limited to propositional logic, and how to apply those procedures for individual theory to the case involving multiple theories. There are also few unresolved/newly emerged questions for me. For instance, there were many approaches regarding the learning procedure after a model failed to satisfy the formula. *Is there a way to select a 'most probable' assignments in advance?* or in the case involving multiple convex theories, *is there an preferred order of checking satisfiability of individual theories to make the SMT solving process efficient?* Further research is needed. But for now, I conclude my review.

References

[1] De Moura, L., & Bjørner, N. Satisfiability modulo theories: introduction and applications. Communications of the ACM, 54(9), 69-77. 2011.

[2] Joao P. Marques-Silva, Ines Lynce, and Sharad Malik. "Conflict-driven clause learning SAT solvers". In: *Handbook of Satisfiability*. Ed. by Armin Biere et al. Vol. 185. IOS Press, 2009. Chap. 4, pp. 131–153.

[3] David Monniaux. A Survey of Satisfiability Modulo Theory. Computer Algebra in Scientific Computing, Sep 2016, Bucharest, Romania.

[4] George B. Dantzig and Mukund N. Thapa. *Linear programming 1: Introduction*. Springer-Verlag. 1997.

[5] David Monniaux. "On using floating-point computations to help an ex- act linear arithmetic decision procedure". In: *Computer-aided verification (CAV)*. LNCS 5643. Springer, 2009, pp. 570–583.

[6] Alexander Schrijver. *Theory of linear and integer programming*. Wiley, 1998.

[7] William Pugh. "The Omega test: a fast and practical integer programming algorithm for dependence analysis". In: Supercomputing. New York, NY, USA: ACM, 1991, pp. 4–13.

[8] Kroening, Daniel, and Ofer Strichman. *Decision procedures*. Springer-Verlag Berlin Heidelberg, 2016.

[9] http://www.cs.utexas.edu/ isil/cs389L/lecture16-6up.pdf

[10] http://gauss.ececs.uc.edu/Courses/c626/lectures/SMT/nelson-oppen.pdf

[11] Barrett, Clark, and Cesare Tinelli. "Satisfiability modulo theories." *Handbook of Model Checking*. Springer, Cham, 2018. 305-343.

[12] Bradley, A. R., & Manna, Z. *The calculus of computation: decision procedures with applications to verification*. Springer Science & Business Media. 2007.

[13] D. C. Cooper. "Theorem proving in arithmetic without multiplication". In: Machine Intelligence 7. Edinburgh University Press, 1972, pp. 91–100.

[14] Rüdiger Loos and Volker Weispfenning. "Applying linear quantifier elim- ination". In: The Computer Journal 36.5 (1993). Special issue on compu- tational quantifier elimination, pp. 450–462.

[15] Programming Z3. Nikolaj Bjørner, Leonardo de Moura, Lev Nachmanson, and Christoph Wintersteiger, https://theory.stanford.edu/ nikolaj/programmingz3.html