Boosting Satisfiability Testing Using Boolean Rings

Daher Kaiss* and Nachum Dershowitz†
Department of Computer Science, Tel-Aviv University, Tel Aviv, Israel
Email: {daher,nachumd}@tau.ac.il

Given a Boolean formula A, the *satisfiability problem* is concerned with finding an assignment of truth values (0 and 1) to the propositional variables in A that gives the formula the value 1 (true), or—when there is no such satisfying assignment—proving that the A is equivalent to the constant 0 (false). Alternatively, to establish validity of A, one can refute its negation B by inferring a contradiction $1 = 0$ from it. In general, a proof will require some form of case splitting. Suppose B contains one or more occurrences of a propositional variable x. From B, one may infer the disjunction $B_0 \vee B_1$, where B_0 and B_1 denote the formula B after making the assignment $x = 0$ or $x = 1$, respectively. If both alternatives lead to a contradiction, then B is unsatisfiable and A is a tautology. Tracking "necessary" assignments is crucial for reducing the number of backtracks that must be performed during the search.

Boolean rings are an algebraic structure that provides an alternative to Boolean algebra. By using exclusive-or ($+$) instead of ($\vee$), there is no need for a negation operator (x' is $x + 1$). The Boolean-ring (exclusive-or) normal-form is a sum of monomials, called a *Zhegalkin polynomial*. The normal form of tautologies is 1, and is 0 for contradictions. The use of this representation in automated deduction was first suggested in [1]. We explore the range of *simplifications* that can be employed efficiently (that is, polynomially) on Boolean ring formulæ, and show how such simplifications can be used to boost performance of satisfiability algorithms. We avoid the potential exponential size increase associated with the use of the distributive law needed to compute Boolean-ring normal forms, splitting on variables instead.

We have implemented a mixed *binary-linear* representation of Zhegalkin polynomials [4], which can help optimize operations like unit propagation and hypothesis testing. Extra variables are used to decompose each polynomial equation $M_0 + \cdots + M_n = 0$ ($n > 1$) into a linear equation $y_0 + \cdots + y_n = 0$ and a set of binomial equations $M_i = y_i$. Satisfiability of binomial equations is (polynomially) equivalent to Horn-clause satisfiability; Gaussian elimination solves the linear part. Cross-fertilization between the two parts adds inferential power. Judicious use of simplification, exploiting algebraic properties of the representation, reduces the necessity for case splitting.

Finally, we present a method for testing validity that adapts Stålmarck's method [3] to the Boolean-ring framework. Stålmarck's algorithm (and the similar method of *recursive learning* [2]) is a novel technique for tautology-checking that has been used successfully for some industrial-scale problems. "Simple" inference rules generate consequences of a given formula. New consequences *shared* by both B_0 and B_1 are then added to the set B. This merging of consequences is iterated as long as possible, after which the allowed nesting depth of case splitting is incremented. In the binary-linear framework, checking whether the binary (or linear part) of B_0 entails a newly derived binomial (or linear, respectively) equation in B_1 (or vice-versa) is polynomial.

* Also with Intel, Haifa, leading a development team for formal verification tools.

† Supported in part by the Israel Science Foundation (grant no. 254/01).

References

[1] J. Hsiang and N. Dershowitz (1983). "Rewrite methods for clausal and non-clausal theorem proving." In: *Proc. of the Tenth International Colloquium on Automata, Languages and Programming* (Barcelona, Spain), Lecture Notes in Computer Science **154**, pp. 331–346, Springer-Verlag, Berlin.

[2] W. Kunz (1994). "Recursive learning: A new implication technique for efficient solutions to CAD problems — Test, verification, and optimization." *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **13**(9): 1143–1158.

[3] M. Sheeran and G. Stålmarck (1998). "A tutorial on Stålmarck's proof procedure for propositional logic." In: *Proc. of the Second Intl. Conference on Formal Methods in Computer-Aided Design*, Lecture Notes in Computer Science **1522**, pp. 82–99. Springer-Verlag, Berlin.

[4] G. S. Shieng (1999). "Search reduction techniques and applications to problems in combinatorics." Doctoral dissertation, Department of Computer Science and Information Engineering, National Taiwan University.