Propositional Satisfiability Algorithms based on Inference of Constraints

Inês Lynce and João Marques-Silva
Technical University of Lisbon, IST/INESC/CEL, Lisbon, Portugal
{ines,jpms}@sat.inesc.pt

Abstract

The area of Propositional Satisfiability (SAT) has been the subject of intensive research in recent years, with significant theoretical and practical contributions. From a practical perspective, a large number of very effective SAT solvers has recently been proposed, most of which based on improvements made to the original Davis-Putnam-Logemann-Loveland (DPLL) backtrack search SAT algorithm. In addition, these recent algorithms utilize advanced conflict analysis procedures, recording the cause of failure and therefore allowing the search to backjump. The new solvers are capable of solving very large, very hard real-world problem instances, which more traditional SAT solvers are totally incapable of.

Future research directions for SAT certainly include devising new paradigms and integrating already used techniques within backtrack search SAT algorithms. With this work we propose the utilization of *hypothetical reasoning* (HR), a new propositional reasoning engine based on inference of constraints. Hypothetical reasoning is an inference mechanism based on formulating and analyzing hypothetical scenarios regarding the propositional formula, that can be used for identifying necessary assignments and for inferring new constraints. Moreover, we envision the integration of HR within intelligent backtrack search SAT algorithms.

1 Introduction

Propositional Satisfiability (SAT) is a well-known NP-complete problem, with extensive applications in many fields of Computer Science and Engineering. In recent years several competing solution strategies for SAT have been proposed and thoroughly investigated. Local search algorithms have allowed solving extremely large satisfiable instances of SAT. These algorithms have also been shown to be very effective in randomly generated instances of SAT. On the other hand, several improvements to the backtrack search Davis-Putnam-Logemann-Loveland algorithm [3] have been introduced. These improvements have been shown to be crucial for solving large instances of SAT derived from real-world applications [1, 7, 10, 11].

Furthermore, formula simplification techniques are also known to be competitive for propositional satisfiability. The ability to reduce either the number of variables or clauses in instances of SAT impacts the expected computational effort of solving a given instance. This ability can actually be essential for specific and hard classes of instances. Interestingly, the ability to infer *new* constraints also impacts the expected computational effort of SAT solvers. As a result, even though new clauses are inferred and added to the CNF formula, the task of solving a target problem instance can become significantly simplified.

2 Hypothetical Reasoning

Hypothetical reasoning (HR) [8] is a generalized propositional reasoning engine that can be used as a proof procedure for SAT. By proposing the integration of assignment probing techniques in a single unified framework, HR captures and generalizes existing proof procedures.

The idea of probing consists in identifying assignments that are deemed necessary, usually called *implied necessary assignments*. Hence, probing can be viewed as a form of reduction of the domain of each variable. Interestingly, these techniques can also be interpreted as a mechanism for identifying suitable resolution operations.

In SAT algorithms, the most often used procedure for identifying necessary assignments is Boolean Constraint Propagation (BCP). Given a set of variable assignments, BCP identifies necessary assignments due to the unit clause rule. However, BCP does not identify *all* necessary assignments given a set of variable assignments and a CNF formula. Indeed, the backtrack search procedure can be augmented with other value probing techniques, namely variable probing [12] and clause probing [6, 9].

Stålmark's Method (SM) [12] is a variable probing technique, which consists of recursively applying the *branch-merge rule* to each variable [5]. Consequently, common assignments to variables are identified, by detecting and merging equivalent branches in the search tree.

For example, consider the formula φ_{SM} containing the following clauses: $\omega_1 = (x_1 \vee x_2)$, $\omega_2 = (\neg x_1 \vee x_3)$, $\omega_3 = (\neg x_2 \vee x_4)$ and $\omega_4 = (\neg x_3 \vee x_4)$. Let us assign variable x_1 both truth values. If we assign $\langle x_1, 0\rangle$, and then apply BCP, we obtain the sequence of implications $\langle x_1, 0\rangle \Rightarrow \langle x_2, 1\rangle \Rightarrow \langle x_4, 1\rangle$, which means $BCP(\langle x_1, 0\rangle) = \{\langle x_1, 0\rangle, \langle x_2, 1\rangle, \langle x_4, 1\rangle\}$. On the other hand, if we assign $\langle x_1, 1\rangle$, we obtain the sequence of implications $\langle x_1, 1\rangle \Rightarrow \langle x_3, 1\rangle \Rightarrow \langle x_4, 1\rangle$, that is $BCP(\langle x_1, 1\rangle) = \{\langle x_1, 1\rangle, \langle x_3, 1\rangle, \langle x_4, 1\rangle\}$. Therefore, we have $BCP(\langle x_1, 0\rangle) \cap BCP(\langle x_1, 1\rangle) = \{\langle x_4, 1\rangle\}$, concluding that any assignment to variable x_1 implies $\langle x_4, 1\rangle$. Hence, $\langle x_4, 1\rangle$ is a necessary assignment.

Recursive Learning (RL) [6, 9] is a form of clause probing, consisting in the recursive evaluation of clause satisfiability requirements for identifying common assignments to variables. Common assignments are deemed

necessary for the clause to become satisfied and consequently for the formula to be satisfied.

Finally, note that SM and RL can be applied using recursion of arbitrary depth, in order to guarantee completeness. The depth defines the number of assumptions to be recursively evaluated.

A recursive version of the SM algorithm can be obtained from [5]. At each depth of the recursive procedure, a set C of conclusions (e.g. unit clauses denoting necessary assignments) is maintained. For every variable x the two possible assignments are evaluated and, depending on the outcomes, the problem instance can be deemed satisfied, or currently unsatisfiable. It is plain to conclude that SM with arbitrary depth identifies all necessary assignments. Moreover, if the problem instance is unsatisfiable, SM is able to conclude so. Moreover, the same reasoning can be applied to RL.

Observe that both SM and RL are based on similar reasoning, despite SM being based on variables and RL on clauses. Finally, even though each of these methods is a complete proof procedure *per se*, each can also be integrated into backtrack search algorithms (possibly with recursion of limited depth).

Hypothetical Reasoning is based on analyzing conditions relating sets of assignments. Given an assignment α, the result of applying BCP to a CNF formula φ as the result of α is denoted by $\text{BCP}(\varphi, \alpha)$. Assignment α is referred to as the *triggering assignment* of the assignments in $\text{BCP}(\varphi, \alpha)$. Reasoning conditions are then analyzed based on a tabular representation of triggering assignments, i.e. the *table of assignments*, where each row represents a triggering assignment, and each column represents a possible implied assignment.

Interestingly, once a table of assignments is filled, a few reasoning conditions can be established, therefore allowing the inference of new constraints [8]. Such conditions entail the branch-merge rule and recursive learning, as well as the failed literal rule [2] and literal dropping techniques [4]. For example, the branch-merge rule can be applied by making the intersection between the implied assignment for assigning both values to each variable x (i.e., $\text{BCP}(\langle x, 0\rangle) \cap \text{BCP}(\langle x, 1\rangle)$).

The reasoning conditions described can be organized in an algorithm that can be used to replace the plain BCP algorithm. The HR algorithm simply extends the established reasoning conditions, implementing recursion in order to guarantee completeness.

3 Current Research Status

This paper introduces hypothetical reasoning (HR), a new complete algorithm for SAT that is based on the inference of constraints. The HR algorithm can be used as an independent algorithm, or as a preprocessing tool for backtrack-search SAT solvers, among other possible applications.

Currently, the hypothetical reasoning algorithm is implemented as a prototype. In a near future, we will be able to incorporate hypothetical reasoning within backtrack search SAT algorithms. We believe that by combining these two algorithms we will achieve interesting improvements in current state of the art SAT solvers, and will allow solving challenging hard instances of SAT.

In addition, a more comprehensive evaluation of the actual effectiveness of HR on real-world problem instances is required. Nevertheless, the promising results obtained in the recent past with assignment probing techniques, suggest that more sophisticated forms of assignment probing may prove crucial for solving different classes of problem instances.

References

[1] R. Bayardo Jr. and R. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the National Conference on Artificial Intelligence*, pages 203–208, July 1997.

[2] J. Crawford and L. Auton. Experimental results on the cross-over point in satisfiability problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 22–28, 1993.

[3] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the Association for Computing Machinery*, 5:394–397, July 1962.

[4] O. Dubois and G. Dequen. A backbone-search heuristic for efficient solving of hard 3-sat formulae. In *Proceedings of the International Joint Conference on Artificial Intelligence*, August 2001.

[5] J. Groote and J. Warners. The propositional formula checker heerhugo. In I. Gent, H. van Maaren, and T. Walsh, editors, *SAT 2000*, pages 261–281. IOS Press, 2000.

[6] W. Kunz and D. Stoffel. *Reasoning in Boolean Networks*. Kluwer Academic Publishers, 1997.

[7] C. M. Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pages 341–355, October 1997.

[8] I. Lynce and J. P. Marques-Silva. Hypothetical reasoning in propositional satisfiability. Technical Report RT/01/2002, INESC, March 2002.

[9] J. P. Marques-Silva and T. Glass. Combinational equivalence checking using satisfiability and recursive learning. In *Proceedings of the ACM/IEEE Design, Automation and Test in Europe Conference*, pages 145–149, March 1999.

[10] J. P. Marques-Silva and K. A. Sakallah. GRASP: A new search algorithm for satisfiability. In *Proceedings of the ACM/IEEE International Conference on Computer-Aided Design*, pages 220–227, November 1996.

[11] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Engineering an efficient SAT solver. In *Proceedings of the Design Automation Conference*, pages 530–535, June 2001.

[12] G. Stålmarck. A system for determining propositional logic theorems by applying values and rules to triplets that are generated from a formula, 1989. Swedish Patent 467 076 (Approved 1992), US Patent 5 276 897 (approved 1994), European Patent 0 403 454 (approved 1995).