Integrating Simplication Techniques in SAT Algorithms

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

Abstract

Propositional Satisfiability (SAT) is fundamental in solving many application problems in Artificial Intelligence and in other fields of Computer Science and Engineering. In the past, it has already been shown that large simplifications can be obtained using algebraic simplification techniques. On the other hand, intelligent backtrack search algorithms for SAT have empirically been shown to be highly effective in pruning the amount of search, applying strategies for non-chronological backtracking and procedures for clause recording. In this paper we propose integrating some algebraic simplification techniques — namely two-variable equivalence and clause inference — within intelligent backtrack search algorithms. When integrated in SAT algorithms, these techniques target the simplification of a problem instance, reducing the associated search cost. With the obtained experimental results we show the importance of the proposed techniques, demonstrating that they are indeed applied in hard real-world problem instances.

1 Introduction

Recent years have seen the proposal of several effective approaches for solving propositional satisfiability. These approaches include, among others, local search and its variations [10, 11], backtrack search improved with different search pruning techniques [2, 8], backtrack search with randomization and restarts [3, 1], and algebraic manipulation [4, 6]. These different approaches have allowed efficiently solving different classes of SAT instances.

Algebraic manipulation targets the simplification of a problem instance, reducing the associated search cost. Using simplification rules it is possible to prune the search space. In the end, we want to conclude we save more time in search than loose in obtaining simplifications.

Experimental results have already demonstrated that large simplifications can be obtained using the proposed techniques [7]. Nevertheless, they have only been applied in the form of a preprocessing engine. The next natural step is to incorporate the techniques in specific search algorithms. However, for backtrack search algorithms that implement some form of learning, special details must be considered.

The remainder of this paper is organized as follows. Section 2 gives more insight into simplification techniques. Afterwards, in Section 3 we describe the integration of algebraic simplification techniques in a SAT solver that implements intelligent backtrack search. Preliminary experimental results are presented and analyzed in Section 4, and the paper concludes in Section 5.

2 Simplification Techniques

Resolution is an inference rule for propositional logic and can be regarded as a general technique for deriving new clauses. However, when applying general resolution, we generate a very large number of redundant and often irrelevant conclusions in the worst case. As a consequence of general resolution being computationally too expensive, some strategies have been developed. These strategies indirectly identify which resolution operations to apply. In other words, they consist in particular ways of applying *selective* resolution, and therefore only simplified formulas can be derived.

Two-variable equivalence is another well-known formula simplification procedure. Let us consider an example, using the pair of clauses $w_a = (\neg x \vee y)$ and $w_b = (x \vee \neg y)$. For any assignment that satisfies the two clauses, the truth values of x and y must be the same. For this reason, variables x and y are said to be equivalent. Therefore one can replace y by x on all occurrences of y. As a consequence, the number of variables in the formula decreases by 1.

Observe that the inference of binary clauses, using selective resolution, can contribute to finding more equivalent variables. Moreover, the inference of clauses can always contribute to add more information to the problem specification, and therefore to simplify the search.

In [7] are proposed subsets of clauses for inferring binary clauses. The objective of studying sets of clauses is to indirectly select to which sets of clauses the resolution operation should be applied to.

3 Integrating Simplifications

From the simplification techniques described above, we implemented two-variable equivalence and inference of binary clauses (using 2B/1T and 0B/4T reasoning [1] described in [7] — see Table 1). Both clause patterns 2B/1T and 0B/4T can be explained resorting to resolution. Additional simplifications were also implemented, to prevent tautologous clauses or repeated literals within a clause, as a consequence of replacing variables.

We start from an existing SAT solver — GRASP [8], an algorithm with intelligent backtrack search. We

[1] Each clause pattern is categorized in terms of how many binary and ternary clauses are involved. In other words, each clause pattern is categorized as being of the form mB/nT, meaning that m binary clauses and n ternary clauses are involved.

Table 1: Rules for Inferring Binary Clauses

Clause Pattern		Inferred Clause
2B/1T	$(l_1 \vee \neg l_2) \wedge (l_1 \vee \neg l_3) \wedge (l_2 \vee l_3 \vee l_4)$	$(l_1 \vee l_4)$
0B/4T	$(l_1 \vee l_2 \vee l_3) \wedge (\neg l_1 \vee \neg l_2 \vee l_3) \wedge (l_1 \vee \neg l_2 \vee l_4) \wedge (\neg l_1 \vee l_2 \vee l_4)$	$(l_3 \vee l_4)$

Table 2: Results for the c1908_bug instance

Simplifications	Nodes	Time
none	22442	787.7
all	16490	494.6

should note that equivalency reasoning in EQSATZ [6] already incorporates a subset of the proposed techniques. However, integrating them in a non-chronological algorithm brings additional technical challenges.

When considered in the context of backtrack SAT algorithms, there are two main concerns for integrating the described simplification techniques:

1. For a backtrack search algorithm, it is necessary to know when to replace or undo previous replacements of variables. (This is also true for SATZ.)
2. For a non-chronological backtrack search algorithm (e.g. GRASP), one has to consider special cases of clause recording, whenever conflicts involve clauses with variables replaced by equivalences.

The solution to guarantee the correctness of the creation of conflict clauses, consists in adding a set of literals to the simplified clauses. Whenever we identify two equivalent variables x and y, due to the existence of binary clauses ω_i and ω_j, we select literals assigned value 0 from ω_i and ω_j. These literals will be added to all clauses where y is replaced by x, assuming variable y is the one that is eliminated.

4 Experimental Results

This section presents and analyzes preliminary experimental results of the techniques proposed in this paper. With the objective of evaluating the formula simplification ability of the simplification techniques, several representative problem instances from different real-world applications were used. We selected benchmarks that are known to have a significant number of equivalency clauses [6]. As counterpart to having such characteristics, these instances are large, hard and structured, and most state-of-the-art SAT solvers are currently unable to solve. For the experimental results presented in this section a PIII @733MHz Linux machine with 256MByte of RAM was used.

We selected two problem instances, distinguished by originating from different application domains: *c1908_bug* from Combinational Equivalence Ckecking [5] and *cnf-r1-b3-k1.2* from Data Encription Standard [9].

From the results presented in Table 2, we conclude that applying simplification techniques reduces the search (number of decision *nodes*) and the time required for solving the *c1908_bug* instance.

Table 3: Results for the cnf-r1-b3-k1.2 instance

Simplifications	Nodes	Time
none	139	23.57
Equivalence	134	21.32
Equiv + 0B/4T	118	20.21
Equiv + 2B/1T	95	16.61
Equiv + 2B/1T +0B/4T (*all*)	84	14.43

Moreover, from the presented results in Table 3 for solving the *cnf-r1-b3-k1.2* instance is possible to evaluate the contribution of clause inference for identifying equivalences and therefore for reducing the search space.

5 Conclusions

In this paper we propose the integration of algebraic simplification techniques within intelligent backtrack search algorithms. The described techniques had already been used in intelligent backtrack search algorithms as a preprocessing engine, but never during the search phase. With the obtained experimental results, we have shown the importance of the proposed techniques, demonstrating that large simplifications can be observed during the search.

References

[1] L. Baptista and J. Marques-Silva. Using randomization and learning to solve hard real-world instances of satisfiability. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2000.

[2] 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, 1997.

[3] C. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *Proceedings of the National Conference on Artificial Intelligence*, pages 431–437, 1998.

[4] J. Groote and J. Warners. The propositional formula checker HeerHugo. *Journal of Automated Reasoning*, 24(1-2):101–125, 2000.

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

[6] C. M. Li. Integrating equivalency reasoning into Davis-Putnam procedure. In *Proceedings of the National Conference on Artificial Intelligence*, August 2000.

[7] J. Marques-Silva. Algebraic simplification techniques for propositional satisfiability. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2000.

[8] J. P. Marques-Silva and K.A. Sakallah. GRASP-A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999.

[9] F. Massacci. Using Walk-SAT and Rel-SAT for cryptographic key search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 290–295, 1999. (DES instances available from URL http://www.uni-koblenz.de/~massacci.

[10] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 290–295, 1993.

[11] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 440–446, 1992.