ACM Communications in Computer Algebra, Vol. 45, No. 2, Issue 176, June 2011

Boolean Gröbner Bases in SAT Solving

Christoph Zengler and Wolfgang Küchlin
Wilhelm-Schickard-Institute for Informatics
University of Tübingen
Tübingen, Germany, 72076
`{zengler,kuechlin}@informatik.uni-tuebingen.de`

Abstract

We want to incorporate the reasoning power of Boolean Gröbner bases into modern SAT solvers. There are many starting points where to plug in the Gröbner bases engine in the SAT solving process. As a first step we chose the learning part where new consequences (lemmas) of the original formula are deduced. This paper shows first promising results, also published at the CASC 2010 in Armenia [1].

1 Introduction

In the last years SAT solvers became a vital tool in computing solutions for problems e.g. in computational biology or especially in formal verification. The vast majority of SAT solvers successfully applied to real-world problems uses the DPLL approach and operates on CNF formulas. An important break-through was the development of clause learning SAT solvers (see [2, chapter 4]), which use an optimized form of resolution dynamically in conflict situations to deduce ("learn") a limited number of lemmas, so called *conflict clauses*.

Current SAT solvers learn exactly one lemma per conflict and try to minimize this clause. On the other hand it is known that binary clauses are very helpful to prune the search space early. So the idea is to deduce new binary clauses from each conflict with the help of Boolean Gröbner bases.

2 The Incorporation of Boolean Gröbner Bases

Following the DPLL approach, the SAT solver assigns truth values to variables until either the formula is satisfied or there is a conflict, i.e. the formula is unsatisfiable under the current variable assignment. In the latter case, we take all clauses involved in this conflict and compute a new clause via resolution. This clause then forces backtracking and a new part of the search tree is examined.

For our approach we choose the Boolean ring $\mathbb{B} = \mathcal{B}^{\leftrightarrow}[x_1, \ldots, x_n]/\operatorname{Id}(x_1^2 + x_1, \ldots, x_n^2 + x_n)$ with the coefficient ring $\mathcal{B}^{\leftrightarrow} = (\leftrightarrow, \vee, \mathbf{T}, \mathbf{F})$ We collect all clauses c involved in the conflict at hand and store their polynomial representation $p(c)$ w. r. t. the Boolean ring $\mathbb{B}$ in a set R. We compute a Gröbner basis $G = \operatorname{gb}(R \cup \mathcal{F})$, where $\mathcal{F}$ is the set of idempotency polynomials $x^2 + x$ for all variables x in $\bigcup_{p \in R} \operatorname{vars}(p)$. We collect all polynomials $p \in G \setminus R$ with $|\operatorname{vars}(p)| = 2$ and add their corresponding clause representation $\operatorname{clause}(p)$ to a set L. The clauses of L are then added to the original set of clauses at an appropriate time in the solving process.

Since current Gröbner basis packages cannot cope with large polynomial systems, we have to restrict the set R in number and length of polynomials. In our current experiments we compute only Gröbner bases of subsets $R' \subseteq R$, where there are between 4 and 6 underlying clauses with 2 to 8 literals. We found these numbers by extensive testing. Choosing more or larger reason clauses often leads to long BGB computations and therefore slows down the overall solving process. Taking fewer or shorter reason clauses does often not produce new binary clauses and therefore does not speed up the solving process.

3 Results

We implemented our approach on top of the publically available 2007 version of MiniSat. For the Gröbner bases computations we used the package `cgb` with lexicographical term ordering for all computations. `cgb` is implemented in the open-source Computer Algebra system Reduce and it is used within the logic package Redlog for various quantifier elimination procedures and for a simplifier based on Gröbner bases. So far we used `cgb` as a black box and therefore other (more efficient Boolean) Gröbner basis implementations could easily be substituted. However, with the current heuristics, we spend only about 1/1000 of the overall time in the Gröbner basis computations.

As benchmark set we chose all 84 instances of the SAT 2009 competition, which could be solved by MiniSat in less than 10,000 s. Our approach clearly outperforms the original MiniSat. This is mainly because our implementation performs especially well on the large and hard benchmark instances of the benchmark set. Accumulating all instances, we produce 13.8% fewer conflicts and therefore save 23.5% of solving time. In the best cases, we could achieve speedup factors up to 3.3 (621.2 s vs. 2195.3 s).

4 Ongoing Research

There are still many open questions. Are there better heuristics when to compute a Gröbner basis, based not only on the number and length of clauses? Can we profit from results about the impact of term orderings? Can we learn slightly longer clauses that are still useful? How much improvement is possible by going to a dedicated implementation of Boolean Gröbner bases? We want to compute the Gröbner bases not only of the clauses of one conflict but also of the collected clauses of different conflicts. These Gröbner bases could then be used to perform simplifications of the problem. This seems especially interesting since with a first implementation of our simple approach in the new Version of MiniSat, we could not yield results as good as with the 2007 version.

References

[1] Zengler, C., Küchlin, W.: Extending clause learning of sat solvers with boolean gröbner bases. In: Computer Algebra in Scientific Computing. Volume 6244 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Germany (2010)

[2] Biere, A., Heule, M., van Maaren, H., Walsh, T., eds.: Handbook of Satisfiability. Volume 185 of Frontiers in Artificial Intelligence and Applications. IOS Press (2009)

The abstracts of the ISSAC 2011 software demonstrations will appear in the next issue.