Learning polynomials over GF(2) in a SAT solver

Georgios Katsirelos, Laurent Simon

Learning Polynomials over GF(2) in a SAT Solver (Poster Presentation)

George Katsirelos[1] and Laurent Simon[2]

[1] INRA, Toulouse, email: george.katsirelos@toulouse.inra.fr
[2] LRI, Univ Paris 11, email: simon@lri.fr

1 Introduction

One potential direction for improving the performance of SAT solvers is by using a stronger underlying proof system, e.g., [1]. We propose a step in improving the learning architecture of SAT solvers and describe a learning scheme in the polynomial calculus with resolution (PCR), a proof system that generalizes both resolution and Gaussian elimination. The scheme fits the general structure of CDCL solvers, so many of the other techniques of CDCL solvers should be reusable.

The PCR proof system was introduced in [2]. In it, lines of a proof are polynomials, which are derived by summing two previous polynomials or multiplying a previous polynomial by a variable. The system also includes the axioms $x^2 - x = 0$, $\neg x^2 - \neg x = 0$ and $x + \neg x = 1$ for all variables x. In our approach, we use only polynomials over $GF(2)$. In this system, a clause $(a \vee b \vee \neg c)$ is expressed as the polynomial $\neg a \neg b c = 0$. A xor-clause $(a \oplus b \oplus \neg c)$ is also naturally expressed, as the polynomial $a + b + \neg c = 0$. However, neither a clause nor a xor clause can capture a general polynomial such as $xy + zw + pq + 1 = 0$. Note that the variables $\neg x$ are not necessary, as they can be replaced by $(1 + x)$ but using them can drastically reduce the number of monomials. When written as a sum of monomials, a global order on variables allows a canonical representation, unique for all equal polynomials.

There is significant previous work that addresses the efficient integration of XOR (or equivalence) reasoning techniques in SAT solvers, e.g. [3, 4]. However, in these approaches, interaction between the CNF and XOR subproblems is limited to passing unit clauses from the CNF part to the XOR part and implied clauses from the XOR part to the CNF part.

2 Structure of the solver

The structure and main loop of the proposed solver is identical to that of CDCL algorithms (not recalled here). However, the basic constraint stored in this scheme is a polynomial. Therefore, the operations we need to specify are propagating the implications of polynomials as we make decisions, and learning new polynomials when we encounter conflicts. The natural way to propagate polynomials is to decompose it into a set of clauses and one xor-clause. $c + \sum_{i=1}^{k} m_i = 0$ where $c \in \{0, 1\}$ and $m_i = \prod_{j=1}^{d} x_{ij}$ can be decomposed with one new variable y_{m_i} for each term m_i by the clauses that encode

$y_{m_i} \iff \bigwedge_{j=1}^{d} x_{ij}$ and the xor-clause $c+\sum_{i=1}^{k} y_{m_i} = 0$. Unfortunately, unit propagation on this decomposition is not complete. Consider the polynomial $ad + bd + 1 = 0$. Unit propagation on the corresponding CNF $x \iff a \wedge d$, $y \iff b \wedge d$, $x \oplus y$ does nothing, but all solutions have d set to true. We can improve this decomposition by factoring common subexpressions, but propagation remains incomplete. However, achieving complete propagation is too expensive and unnecessary.

In order to perform conflict analysis, we define a *polynomial resolution step*:

$$\frac{\begin{array}{c} yp_1 + p_2 \\ yq_1 + q_2 \end{array}}{\begin{cases} p_1q_2 + p_2q_1 & \text{if } p_1 \neq q_1 \\ p_2 + q_2 & \text{if } p_1 = q_1 \end{cases}}$$

This allows us to use polynomial resolution in much the same way as resolution: we keep track of the polynomial that forced each literal. On conflict, we iteratively resolve away the deepest variable until we get a polynomial that satisfies a stopping condition, such as having a single variable at the decision level. However, there are cases where polynomial resolution is less well behaved than resolution. First, during conflict analysis we may get a polynomial which contains no variable from the last decision level. Second, even if there exists a 1-UIP polynomial, it is not necessarily asserting. Third, a polynomial may contain variables which are assigned but which do not affect the satisfiability of that polynomial under the current assignment. Resolving on these variables may result in a tautology, so these variables have to be ignored. Additionally, the size of polynomials may grow quadratically with every polynomial resolution step. To keep their size in check, we propose several simplification procedures that can reduce their size, as well as a weakening procedure that, given a polynomial p gives a smaller and weaker polynomial p' such that $p' = 1 \implies p = 1$ and $p = 0 \implies p' = 0$.

In terms of its theoretical power, this solver is not any more powerful than a CDCL solver if the input is given in CNF and therefore also strictly less powerful than the unrestricted polynomial calculus. Thus, we apply a preprocessing step in which we detect XOR clauses and AND gates to extract implied polynomials, as in [5]. Given this preprocessing step, the solver is strictly more powerful than CDCL, as it clearly p-simulates resolution and additionally p-simulates Gaussian elimination but also more powerful than SMT with XOR reasoning [4].

References

1. Dixon, H.E., Ginsberg, M.L.: Inference methods for a pseudo-boolean satisfiability solver. In: AAAI 2002. (2002) 635–640
2. Alekhnovich, M., Ben-Sasson, E., Razborov, A.A., Wigderson, A.: Space complexity in propositional calculus. SIAM J. Comput **31**(4) (2002) 1184–1211
3. Soos, M., Nohl, K., Castelluccia, C.: Extending sat solvers to cryptographic problems. In: SAT 2009. (2009) 244–257
4. Laitinen, T., Junttila, T., Niemel, I.: Extending clause learning DPLL with parity reasoning. In: ECAI 2010. (2010) 21–26
5. Ostrowski, R., Grégoire, É., Mazure, B., Sais, L.: Recovering and exploiting structural knowledge from cnf formulas. In: Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002. (2002) 185–199