SWORD – Module-based SAT Solving

Robert Wille, Jean Christoph Jung, André Sülflow and Rolf Drechsler

Institute of Computer Science
University of Bremen
28359 Bremen, Germany
{rwille,jeanjung,suelflow,drechsle}@informatik.uni-bremen.de

Introduction

In this paper, we describe *SWORD* – a decision procedure for bit-vector logic that uses SAT techniques and exploits word level information [6]. The main idea of SWORD is based on the following observation: While current SAT solvers perform very well on instances with a large number of logic operations, their performance on arithmetic operations degrades with increasing data-path width. In contrast, pure word-level approaches are able to handle arithmetic operations very fast, but suffer from irregularities in the word-level structure (e.g. bit slicing).

SWORD tries to combine the best of both worlds: On the one hand, it includes fast propagation, sophisticated data structures, as well as advanced techniques like non-chronological backtracking and learning from modern SAT solvers. On the other hand word-level information is exploited in the decision heuristic and during propagation. Additionally, rewriting on word-level and bit-level is performed before starting the search process. In this sense, SWORD has certain similarities with SMT solvers (e.g. [2]).

The Solver

The overall architecture of SWORD is shown in Figure 1. At first, the instance is rewritten on word-level, i.e., rules for distributivity and commutativity are applied. Afterwards, the resulting (already simplified) instance is translated to an AIG-like data structure [4]. More precisely, a data structure not only supporting *and*-nodes but also *iff*-nodes is used for rewriting. The rules for *and*-nodes are adapted from [5] and have been extended for *iff*-nodes.

After rewriting, the bit-level data structure is converted into *Conjunctive Normal Form* (CNF) and given to the solve engine (see right side of Figure 1). Basically, the solve engine of SWORD is a DPLL style decision procedure as deployed in many state-of-the-art SAT solvers [3]: While free variables remain, a free variable is assigned, and its implications are propagated. If a conflict occurs, it is analyzed and a conflict clause is learnt.

SWORD extends the basic algorithm by so called "modules" (for more details see [6]). Modules can be instantiated for, principally, any sub-unit of the formula

Dagstuhl Seminar Proceedings 09461
Algorithms and Applications for Next Generation SAT Solvers
http://drops.dagstuhl.de/opus/volltexte/2010/2506

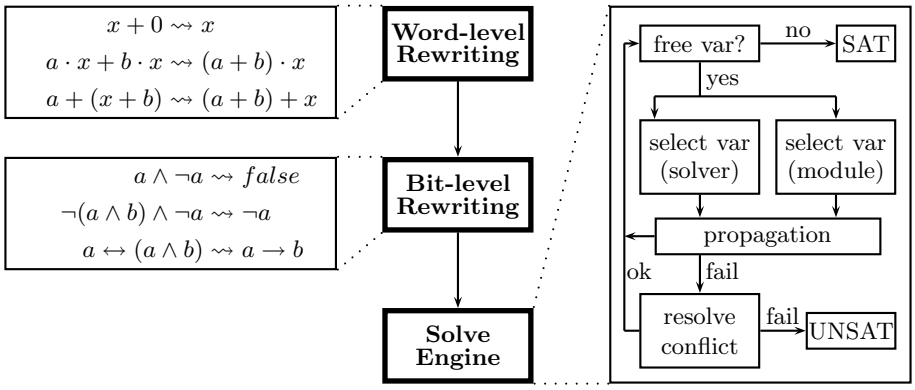

Fig. 1. The overall architecture of SWORD

under consideration. However, currently supported modules are multiplication and addition. The motivation of modules is twofold: First, they are used for propagation where a translation to CNF would be too expensive. Second, high level information is exploited inside modules, e.g., for making decisions.

Applying modules yields several new options. As an example, there is a choice between using the SAT solver's decision heuristic and the decision heuristic of a module (as depicted on the right side of Figure 1). The decision heuristic of a module depends on the type of the module and uses module specific high level information. For example, the heuristic of the addition module differs from that of the multiplication module.

SWORD supports the QF_BV logic defined in [7] and is implemented in C++ on top of the SAT solver MiniSat [3]. The parse routine of the solver is based on the grammar of *Smt2Sf* [1].

References

1. D. Babic. Smt2Sf. http://www.cs.ubc.ca/~babic/index_tools.htm.
2. B. Dutertre and L. Moura. A fast linear-arithmetic solver for DPLL(T). In *Computer Aided Verification*, volume 4114 of *LNCS*, pages 81–94, 2006.
3. N. Eén and N. Sörensson. An extensible SAT solver. In *Theory and Applications of Satisfiability Testing 2003*, volume 2919, pages 502–518, 2004.
4. A. Kuehlmann, V. Paruthi, F. Krohm, M. K. Ganai:. Robust Boolean reasoning for equivalence checking and functional property verification. In *Trans. on CAD of Integrated Circuits and Systems*, volume 21(12), pages 1377–1394, 2002.
5. R. Brummayer and A. Biere. Local Two-Level And-Inverter Graph Minimization without Blowup. In *Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, 2006
6. R. Wille, G. Fey, D. Große, S. Eggersglüß, and R. Drechsler. SWORD: A SAT like Prover Using Word Level Information. In *Int'l Conference on Very Large Scale Integration*, pages 88–93, 2007. http://www.informatik.uni-bremen.de/agra/eng/sword.php.
7. S. Ranise and C. Tinelli. The SMT-LIB Standard: Version 1.2. http://combination.cs.uiowa.edu/smtlib/