SWORD v0.2 – Module-based SAT Solving

Robert Wille André Sülflow Rolf Drechsler

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

In this paper, we present *SWORD* – a SAT like solver that facilitates word level information[1]. The main idea behind SWORD is based on the following observation: Current SAT solvers perform very well on instances with a large number of logic operations. But when more complex functions like arithmetic units are considered, the performance degrades with increasing data-path width. In contrast, pure word level approaches handle e.g. arithmetic operations very fast but suffer from complexity problems when irregularities in the word level structure (e.g. bit slicing) occur.

SWORD tries to combine the best of both worlds: Logic operations like *(bv)and*, *(bv)or*, and *(bv)xor* are represented in terms of clauses while more complex functions like arithmetic operations or shifts are represented by so called *modules*. These modules inherit a problem specific decision as well as a problem specific propagation strategy, which is exploited during the search. Thus, SWORD combines the advantages of a Boolean proof procedure with the power of word level knowledge. Moreover, SWORD is not limited to pre-defined encodings as CNF or QF_BV logic. Problem specific modules for respective domains can be developed.

Algorithm

The overall algorithm of SWORD is shown in Fig. 1. The flow is similar to the DPLL procedure as applied in standard SAT solvers: While free variables remain (a) a decision is made (c). Implications resulting from this decision are carried out (d). If a conflict occurs, it is analyzed (f). The important difference is that SWORD has two operation levels: the *global* algorithm controls the overall search process, handles all clauses, and calls the *local* procedures of the modules for decision and implication. Thus, decision making and implication engine can be adjusted by the modules.

In more detail, the solver first chooses a particular module based on a *global decision heuristic* (c.1). Here a module is selected that assigns a value to one of its connected variables. Therefore, a (global) heuristic is employed to decide which modules are "more important" than others. To determine the importance of a particular module, semantical information such as the type of the operation are available.

After global decision, the selected module chooses a value for one of its variables according to a *local decision heuristic* (c.2). Therefore, different strategies are applied for different types of modules (which concrete decision is made depends on the type of a module). For example, a module representing a multiplier uses a different heuristic than a module representing a shift operation.

[1] A more detailed description of the initial version of SWORD can be found in [3].

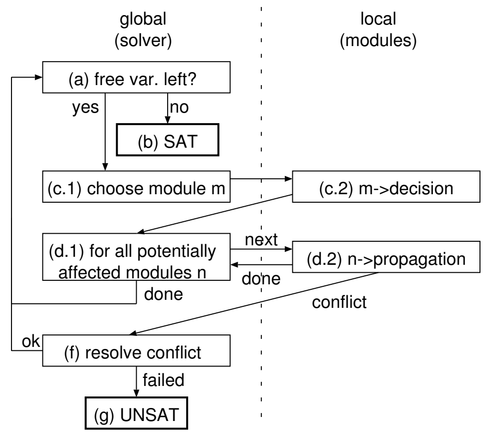

Fig. 1. General Idea

Afterwards the solver calls the *local implication procedures* (d.2) of all modules that are potentially affected (d.1) by the previous decision or implication. The chosen modules imply further assignments and detect conflicts. Again, the concrete strategy depends on the type of a module.

Implementation Details

SWORD in its current version has been (re)implemented on the top of the SAT solver *MiniSat* [2]. Furthermore, the readin routine of the QF_BV input language is based on the grammar of *Smt2Sf* [1]. Addition, multiplication, shifts, and ITE-operations are handled in terms of modules. All remaining QF_BV-operations are reformulated to these operations or represented by clauses, respectively. Moreover, further (problem specific) modules for respective domains can be developed. One example showing how a problem specific module will improve the solving time has been presented in [4] for logic synthesis of reversible circuits.

SWORD will participate in the QF_BV division at SMT Comp using the random seed 823.

Acknowledgements

We like to thank Görschwin Fey, Daniel Große, and Stephan Eggersglüß for helpful discussions and their contribution to the initial implementation of the solver presented in [3].

References

1. D. Babic. Smt2Sf. `http://www.cs.ubc.ca/~babic/index_tools.htm`.
2. N. Eén and N. Sörensson. An extensible SAT solver. In *Theory and Applications of Satisfiability Testing 2003*, volume 2919, pages 502–518, 2004.
3. 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.
4. R. Wille and D. Große. Fast exact Toffoli network synthesis of reversible logic. In *Int'l Conf. on CAD*, pages 60–64, 2007.