The SAT Solver KW

BY JOHAN ALFREDSSON, OEPIR CONSULTING

johan@oepir.com

Introduction

KW is a propositional satisfiability solver. The aim when creating it has been to build a flexible SAT framework that can easily be extended in several directions. Therefore, the architecture is extremely modular and in the trade-off between speed and generality, the latter has almost always been favoured. At the heart of the KW solution process is a strategy selection mechanism [1] that time-slices different strategies to simplify and solve the problem instances. KW supports incremental problem solving and proof generation.

Strategies

KW is parametrized over which strategies to use to try to solve a SAT instance. Currently, the following strategies are available:

DPLL. The implementation features most modern extensions like 1-UIP conflict generation [2], conflict clause minimization [3], watched literals [4], efficient VSIDS heuristics and rapid restarts [5].

Intersection. By branching on both a literal and its complement and recording the similarities between these propagation results, information about literal values can be propagated which simplifies the problem instance. Further details exist in [6].

Strongly connected components. Finding the strongly connected components of the binary implication graph, equivalent literal information is extracted. By selecting representatives for equivalence classes of literals, the clause database may be simplified after a representative literals normalization. For further information, see [7].

Variable elimination. By performing all possible resolutions and adding those to the clause database, a variable can be eliminated from the problem instance. To avoid extensive problem growth, only variables which do not generate a big increase in memory usage are eliminated [8].

Subsumption. A clause for which there is another shorter clause in the clause database containing only literals from the first clause is redundant. This strategy finds and removes such redundant clauses [9]. It also finds similar redundancies after one resolution step.

Automatic. This is a meta-strategy that select the next strategy to run from a set of allowed strategies.

Circuit support

KW supports solving circuit SAT problems formulated in the AIGER [10] format. Currently, only a simple translation from AIGER to CNF is done using a naive Tseitin transformation [11] after which the regular CNF instance solving takes over.

Future work

KW still lacks many bells and whistles. The primary road ahead consists of adding more strategies to complement the existing ones. Two more strategies are in development and yet three more are planned. Furthermore, the idea is to improve the capabilities of solving circuit problems. There is also on-going work to streamline the KW API to suit particular higher-level user applications.

References

[1] Johan Alfredsson – The SAT Solver Oepir, in *The SAT Competition 2004: Solver descriptions*

[2] Joao Marques-Silva and Karem Sakallah – Conflict Analysis in Search Algorithms for Propositional Satisfiability, in *Proceedings of the International Conference on Tools with Artificial Intelligence, 1996*

[3] Niklas Eén and Niklas Sörensson – MiniSat – A SAT Solver with Conflict-Clause Minimization, in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2005*

[4] Matthew Moskewicz, Conor Madigan, Lintao Zhang and Sharad Malik – Chaff: Engineering an Efficient SAT Solver, in *Proceedings of the Design Automation Conference, 2001*

[5] Jinbo Huang – The effects of restarts on the efficiency of clause learning, in *Proceedings of the International Joint Conference on Artificial Intelligence, 2007*

[6] Daniel Le Berre – Exploiting the real power of unit propagation lookahead, in *Proceedings of the Workshop on Theory and Applications of Satisfiability Testing, 2001*

[7] Ronen Brafman – A Simplifier for Propositional Formulas with many Binary Clauses, in *Proceedings of the International Joint Conferences on Artificial Intelligence, 2000*

[8] Sathiamoorthy Subbarayan and Dhiraj K Pradhan – NiVER: Non Increasing Variable Elimination Resolution for preprocessing SAT instances, in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2004*

[9] Niklas Eén and Armin Biere – Effective Preprocessing in SAT through Variable and Clause Elimination, in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2005*

[10] Armin Biere – The AIGER And-Inverter Graph (AIG) Format Version 20070427, 2007

[11] G. S. Tseitin – Om the complexity of derivation in propositional calculus, in *Constructive Mathematics and Mathematical Logic, Part II, 1968*