(Q)CompSAT and (Q)PicoSAT at the SAT'06 Race

Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University, Linz, Austria

Abstract

This report describes the new features of our four SAT solvers that entered the SAT'06 Race affiliated to the SAT'06 conference in Seattle. Two core solvers were submitted, CompSAT and PicoSAT. We also integrated these two solvers as back end into our QBF solver Quantor [1]. Quantor can be used for purely propositional problems. Then it is used as preprocessor very similar to SATeLite [6]. The resulting solvers are called QCompSAT and QPicoSAT.

CompSAT Version 1.5

The implementation of CompSAT followed ZChaff [5]. Since our original implementation as described in [2] we worked on decomposing SAT problems into disconnected components [3]. We also integrated two features from MiniSAT Version 1.14. One is conflict clause shrinking and the other is to use a priority queue implemented as heap for ranking decision variables. As decision heuristics we followed the approach taken by BerkMin [4]. However, CompSAT still does not have a sophisticated reduction strategy. Learned clauses in essence can only be discarded if they become top level satisfied. Often, CompSAT needs too much memory.

PicoSAT Version 0.1

PicoSAT follows the main design decisions of MiniSAT Version 1.14. It implements clause shrinking, simplification of clauses by top level assigned literals and an activity based reduction of learned clauses, which is memory aware. It has support for generating compressed proof traces and cores for unsatisfiable instances *in memory*. This feature can be enabled at run time. Furthermore, PicoSAT learns failed literals and binary clauses on-the-fly. In contrast to MiniSAT, binary clauses are stored as ordinary clauses, but on the other hand prioritized in BCP.

PicoSAT uses an efficient representation of occurrence lists. Instead of organizing the lists as stacks of pointers to clauses in which a literal occurs, we added two link fields to each clause, one for each watched literal. When we switched to this representation we experienced speed-ups of up to 30%. As in CompSAT and MiniSAT we use Van Gelder's idea of keeping the two watched literals of a clause at its front.

QCompSAT and QPicoSAT

Quantor as opposed to SATeLite does not support self subsuming resolution and in addition has a slightly different optimization criteria. It tries to minimize the number of literals and not the number of clauses. SATeLite ignores trivial clauses when calculating the benefit of eliminating a variable, while Quantor for efficiency reasons of the elimination procedure does not. Our experience is that Quantor is not as efficient as SAT preprocessor as SATeLite.

References

[1] A. Biere. Resolve and expand. In *Proc. SAT'04*, volume 3542 of *LNCS*. Springer.

[2] A. Biere. The evolution from Limmat to Nanosat. Technical Report 444, Dept. of Computer Science, ETH Zürich, 2004.

[3] A. Biere and C. Sinz. Decomposing SAT problems into connected components. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:191–198, 2006.

[4] E. Goldberg and Y. Novikov. BerkMin: a fast and robust SAT-solver. In *Proc. Design and Test Conf. Europe (DATE'02)*.

[5] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. 38th Design Automation Conference (DAC'01)*.

[6] N. Één and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proc. SAT'05*, volume 3569 of *LNCS*.