P{re,i}coSAT@SC'09

Armin Biere

Johannes Kepler University, Linz, Austria

Abstract. In this note we describe the new features of PicoSAT version 913 as it was submitted to the SAT competition 2009. It also contains a description of our new solver PrecoSAT version 236, which tightly integrates various preprocessing techniques into a PicoSAT like core engine.

PicoSAT 193

The results of the SAT Race 2008 [15] showed that the old version of PicoSAT which is mostly covered in [7] was actually doing very well up to a certain point where its ability to solve more instances stagnated. This is particularly apparent in the cactus plots [15]. Our analysis revealed that the garbage collection schedule to reduce the number of learned clauses was much more aggressive than in earlier versions which did not use rapid restarts. In essence, PicoSAT in the last SAT Race did not keep enough learned clauses around. In order to use the original reduce policy of PicoSAT, which is similar to the one in MiniSAT [9], we separated the reduce scheduler from the restart scheduler.

To simplify comparison with other solvers, we also use Luby [13] style restart scheduling [11] instead of our inner/outer scheme [7]. Beside this clean-up work, we added two new features, which we have not seen described in the literature before. First, we employed a new literal watching scheme, that uses a literal move-to-front strategy for the literals in visited clauses instead of just swapping the new watched literal with the head respectively tail literal. In our experiments this reduces the average number of traversed literals in visited clauses considerably.

While minimizing learned clauses [17], it seems to be counter-productive, as also explained in [17], to resolve with binary clauses extensively. Learned clauses can be shortened this way, even without decreasing backjumping (backward pruning). But using these learned clauses shortened by extensive binary clause reasoning in a conflict driven assignment loop [14] results in less propagation (forward pruning). This argument can be turned around as follows. Maybe it is better to continue propagating along binary clauses and not stop at the first conflict, but at the last. We experimented with some variations of this idea. It turns out that a conflict that occurs while visiting a longer clause should stop BCP immediately. But for binary clause we run propagation until completion and only record the last conflict that occurred, which is then subsequently used for conflict analysis.

PrecoSAT 236

In the last SAT Race it became apparent, that in order to be successful in these competitions, the integration of a preprocessor, such as SATeLite [8] is mandatory. Last year we experimented, with an external simplifier, called PicoPrep [4], which shares many ideas with SATeLite [8] and Quantor [6]. As in Quantor we used signatures heavily and also functional substitution whenever possible instead of clause distribution. A new feature was to use signatures in forward subsumption as well. Our new solver PrecoSAT is a prototype that allows us to experiment with tight integration of these ideas into a more or less standard PicoSAT/MiniSAT like core engine. This year in PrecoSAT with respect to SATeLite-like preprocessing, we additionally support, functional substitution of XOR and ITE gates. XOR gates with an arbitrary number of inputs are extracted. Gate extraction uses signatures as in Quantor.

We also implemented all the old and new features of PicoSAT discussed before and in addition revived and old idea from PicoSAT 2006 [5], which learns binary clauses during BCP, whenever a forcing clause can be replaced by a new learned binary clause. This can be checked and implemented with negligible overhead in the procedure that assigns a forced variable, by maintaining and computing a dominator tree for the binary part of the implication graph. As a new feature of PrecoSAT during simplification of the clause data base, we decompose the binary clause graph into strongly connected components and merge equivalent literals. We conjecture that the combination of these two techniques allows to simulate equivalence reasoning with hyper-binary resolution [2] and structural hashing.

Most of the binary clauses in PrecoSAT are learned during failed literal preprocessing, which is the only preprocessing technique currently available in plain PicoSAT. Equivalent literals are also detected during failed literal preprocessing and in addition with the help of a hash table. The hash table also allows fast self-subsuming resolution for binary clauses.

The reduce scheduler was simplified and in addition to enlarge the reduce limit on learned clauses in a geometric way, as in PicoSAT/MiniSAT, we also shrink it proportionally to the number of removed original clauses eliminated during simplification and preprocessing phases. We maintain a doubly linked list of all learned clauses, which together with a move-to-front policy [10] allows us to remove the least active learned clause during conflict analysis. Full reduction as in PicoSAT/MiniSAT is only needed if too many inactive clauses are used as reasons.

For the decision heuristic we use a low-pass filter on the number of times a variable is involved in producing a conflict, implemented as an infinite impulse response filter of order 3. This order is configurable at run-time. An order of 1 gives similar characteristics as the exponential VSIDS scheme described in [3].

The most important aspect of PrecoSAT is however, that all three preprocessing techniques, using strongly connected components, failed literals, and SATeLite style preprocessing are tightly integrated in the main loop of the solver, and can be run after new top level units or new binary clauses are derived. The scheduling of these preprocessors during the search is rather complex and leaves place for further optimizations.

We also integrated blocking literals [16] to reduce the number of visited clauses during BCP and also experimented with more general implication graph analysis [1]. Finally, we flush the phase-saving-cache in regular intervals, also controlled by a Luby strategy, and "rebias" the search by recomputing new phase scores from scratch taking also learned clauses into account. This in contrast to PicoSAT, where we compute a static two-sided Jeroslow-Wang [12] score as phase bias once using the original clauses only.

The tight integration of all these optimizations was very difficult to implement. We spent considerable time in debugging very subtle bugs, also because PrecoSAT can not produce proof traces yet. Accordingly, PrecoSAT is still considered to be in an early stage of development. Moreover, there is only a partial understanding how these optimizations interact.

References

1. G. Audemard, L. Bordeaux, Y. Hamadi, S. Jabbour, and L. Sais. A generalized framework for conflict analysis. In *Proc. SAT'08*.
2. F. Bacchus and J. Winter. Effective preprocessing with hyper-resolution and equality reduction. In *Proc. SAT'03*.
3. A. Biere. Adaptive restart control for conflict driven SAT solvers. In *Proc. SAT'08*.
4. A. Biere. PicoSAT and PicoAigerSAT entering the SAT-Race 2008.
5. A. Biere. (Q)CompSAT and (Q)PicoSAT at the SAT'06 Race.
6. A. Biere. Resolve and expand. In *Proc. SAT'04*, 2004.
7. A. Biere. PicoSAT essentials. *JSAT*, 4, 2008.
8. N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proc. SAT'05*, 2005.
9. N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. SAT'03*, 2003.
10. R. Gershman and O. Strichman. HaifaSat: A new robust SAT solver. In *Proc. Haifa Verification Conference*, 2005.
11. J. Huang. The effect of restarts on the eff. of clause learning. In *Proc. IJCAI'07*.
12. R. Jeroslow and J. Wang. Solving propositional satisability problems. *Annals of mathematics and AI*, 1, 1990.
13. M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47, 1993.
14. J. Marques-Silva, I. Lynce, and S. Malik. Conflict-driven clause learning SAT solvers. In *Handbook of Satisfiability*. IOS Press, 2009.
15. C. Sinz. SAT-Race'08. http://baldur.iti.uka.de/sat-race-2008.
16. N. Sörensson. MS 2.1 and MS++ 1.0 SAT Race 2008 editions.
17. N. Sörensson and A. Biere. Minimizing learned clauses. 2009. Submitted.