CirCUs 2.1 - SAT Race 2010 Edition*

Hyojung Han[1], HoonSang Jin[2], Hyondeuk Kim[1], and Fabio Somenzi[1]

[1] University of Colorado at Boulder
[2] Cadence Design Systems
{Hyojung.Han,Hyondeuk.Kim,Fabio}@colorado.edu
{hsjin}@cadence.com

1 Description

CirCUs is a SAT solver based on the DPLL procedure and conflict clause recording [7, 8, 2]. CirCUs includes most current popular techniques such as two-watched literals scheme for BCP, activity-based decision heuristics, clause deletion strategies, restarting heuristics, and first UIP-based clause learning. In particular, CirCUs adopts transformations of the CNF formula to be decided that allow them to do more through deduction and decrease their reliance on enumeration [4, 6].

Detecting whether the resolvent of two clauses subsumes either operand is easy and inexpensive. Therefore, checking *on-the-fly* for subsumption [5] can be added with almost no penalty to those operations of SAT solvers that are based on resolution. This detection is used to improve three stages of CirCUs: variable elimination, clause distillation, and conflict analysis.

Simplifying the CNF clauses leads to fast Boolean Constraint Propagation (BCP) and to earlier detection of conflicts in practice. CirCUs is incremented with preprocessing based on subsumption, variable elimination [1, 9], and distillation [4]. Resolution is the main operation in preprocessing. Therefore, on-the-fly susbsumption is also applied to the preprocessors for variable elimination and clause distillation. During eliminating variables, at each resolution operation, we can check if one of the operands is subsumed by the resolvent, like the on-the-fly subsumption check in conflict analysis. A clause can be simplified by on-the-fly subsumption, regardless of whether the variable is eventually eliminated. Conflict analysis in clause distillation also performs resolutions steps as conflict analysis in DPLL. Therefore we can increase efficiency in the distillation procedure by using on-the-fly simplification.

In addition to conflict learned clauses, this version of CirCUs utilizes the clauses learned by dominator analysis during the deduction procedure tend to produce smaller implication graphs and sometimes increase the deductive power of the input CNF formula. In our approach, immediate dominators are computed with an efficient self-subsumption check, and a learned clause based on dominators contains two literals. This approach can also be extended to clauses with more than two literals by computing multiple dominators. Our experiments showed that these learned clauses based on multiple dominators improve the performance of CirCUs.

Two variables p and q are equivalent in formula F if and only if $(p \rightarrow q)$ and $(q \rightarrow p)$ are implicates of F. Variables in equivalence relation belong to the same

* Supported by SRC contract 2009-TJ-1859.

equivalence class. In an equivalence class, a representative is selected and it substitutes for all other variables in the clause database. This yields fewer variables, and allows the SAT solver to explore a reduced search space.

A dominator clause tends to help in the identification of equivalent variables. As many dominator clauses with two-literals are generated, more chances to discover equivalent variables are produced. Thanks to a fact in [3], CirCUs may detect equivalence relations without explicitly checking two-literal clauses in formula F during implication. In addition, the two-literal clauses can be examined periodically to find equivalences that escape the on-the-fly detection.

CirCUs is written in C. An ANSI C compiler and GNU make are required to build it. It is supposed to be compiled for 32-bit machines.

References

[1] N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT 2005)*, pages 61–75, St. Andrews, UK, June 2005. Springer-Verlag. LNCS 3569.

[2] E. Goldberg and Y. Novikov. BerkMin: A fast and robust SAT-solver. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 142–149, Paris, France, Mar. 2002.

[3] H. Han, H. Jin, and F. Somenzi. Learning clauses based on multiple dominators to solve hard propositional SAT instances, 2010. submitted.

[4] H. Han and F. Somenzi. Alembic: An efficient algorithm for CNF preprocessing. In *Proceedings of the Design Automation Conference*, pages 582–587, San Diego, CA, June 2007.

[5] H. Han and F. Somenzi. On-the-fly clause improvement. In *Twelfth International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, pages 209–222, Swansea, UK, June 2009. Springer-Verlag. LNCS 5584.

[6] H. Han, F. Somenzi, and H. Jin. Making deduction more effective in SAT solvers. *IEEE Transactions on COMPUTER-AIDED DESIGN of Integrated Circuits and Systems*, 2010. to appear.

[7] J. P. Marques-Silva and K. A. Sakallah. Grasp—a new search algorithm for satisfiability. In *Proceedings of the International Conference on Computer-Aided Design*, pages 220–227, San Jose, CA, Nov. 1996.

[8] M. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the Design Automation Conference*, pages 530–535, Las Vegas, NV, June 2001.

[9] URL: http://http://minisat.se/MiniSat.html.