Eureka-2008 SAT Solver

Alexander Nadel[1], Vadim Ryvchin[1]

[1] Design Technology Solutions Group
Intel Corporation
Haifa, Israel
{alexander.nadel; vadim.ryvchin}@intel.com

Abstract. We describe the SAT solver Eureka. Eureka is a state-of-the-art SAT solver, used in various Formal Verification flows at Intel. Eureka is based upon backtrack search DLL algorithm, enhanced by failure-driven assertion loop; non-standard conflict analyses; restart and clause deletion strategies; CBH decision heuristic and decision stack shrinking.

1 Basic Algorithm

Eureka makes usage of the following well known algorithms:

- Subsumption and resolution-based preprocessing [9]
- DLL algorithm, enhanced by Boolean Constraints Propagation (BCP)
- Failure-driven assertion loop [1]
- Conflict clause recording: Eureka records the 1IUP conflict clause [2], enhanced by conflict clause minimization [3]. Eureka records another conflict clause, called local conflict clause. More details are provided in the next subsection
- Local restart strategy [10]
- Conflict clause deletion strategy, based on the age and the length of the clauses [4, 5]
- Decision stack shrinking [5, 6]
- CBH decision heuristic [7] is used starting from the 2nd restart. Berkmin's decision heuristic [4] is invoked for the first 2 restarts

2 Local Conflict Clause recording

On conflict occasion, Eureka records a *local conflict clause* in addition to the minimized 1UIP conflict clause. The local conflict clause is created as follows. Suppose that the decision level on conflict occasion is *dl*. In addition to the decision variable *A*, the decision level *dl* may contain some *flipped* variables, that is, variables whose value was flipped as a result of a failure-driven assertion. Each flipped variable is implied in some conflict clause.

Eureka treats the last flipped variable F as if it was a decision variable. It increments the decision level of F and all the implied literals, assigned after F and marks F a non-implied decision variable. Then, it creates a minimized 1UIP conflict clause *w.r.t to the new decision level*. This conflict clause is referred to as a local conflict clause.

Local conflict clause is used for decision stack shrinking whenever possible.

References

[1] J.P. Marques-Silva and K.A. Sakallah. GRASP: A search algorithm for propositional satisfiability. IEEE Transactions on Computers, (48):506-521, 1999.

[2] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik. Chaff: Engineering an efficient SAT solver. In Proceedings of the Design Automation Conference, 2001.

[3] N. Eén, N. Sörensson. MiniSat — A SAT Solver with conflict-clause minimization, poster for SAT 2005.

[4] E. Goldberg and Y. Novikov. BerkMin: A fast and robust SAT-solver. In Design, Automation, and Test in Europe (DATE '02), p. 142-149, March 2002.

[5] Y.S. Mahajan, Z. Fu, S. Malik. ZChaff2004: an efficient SAT solver. In Preliminary Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT 2004), 2004.

[6] A. Nadel. Backtrack Search Algorithms for Propositional Logic Satisfiability: Review and Innovations. Master's thesis, the Hebrew University, 2002.

[7] N. Dershowitz, Z. Hanna, and A. Nadel. A Clause-Based Heuristic for SAT Solvers. In Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT 2005), 2005.

[8] N. Dershowitz, Z. Hanna, and A. Nadel. Towards a better understanding of the functionality of a conflict-driven SAT solver. In J. Marques-Silva and K. A. Sakallah,editors, SAT, volume 4501 of Lecture Notes in Computer Science, pages 287–293. Springer, 2007.

[9] N. Eén and A. Biere. Effective Preprocessing in SAT through Variable and Clause Elimination, In Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT 2005), 2005.

[10] V. Ryvchin and O. Strichman. Paper in preparation.