LySAT: solver description

Youssef Hamadi[1,2], Said Jabbour[3], and Lakhdar Sais[4]

[1] Microsoft Research, 7 J J Thomson Avenue, Cambridge, United Kingdom
youssefh@microsoft.com
[2] LIX École Polytechnique, F-91128 Palaiseau, France
[3] Microsoft-INRIA joint-lab, 28, rue Jean Rostand, 91893 Orsay, France
said.jabbour@inria.fr
[4] CRIL-CNRS, Université Lille Nord de France
Rue Jean Souvraz SP18, F-62307 Lens Cedex France
sais@cril.fr

Overview

LySAT is a DPLL-based satisfiability solver which includes all the classical features like lazy data-structures and activity-based decision heuristics. It differs from well known satisfiability solvers such as Rsat [6] and MiniSAT [3] on many important components such as restart strategies and clause learning. In addition to the classical first-UIP scheme, it incorporates a new technique which extends the classical implication graph used during conflict-analysis to exploit the satisfied clauses of a formula [1]. It also includes a dynamic restart strategy, where the cut-off value of the next restart is computed using information gathered in the two previous runs. Finally, a new phase-learning [4, 6] policy based on the computed occurrences of literals in the learnt clauses is used.

Additionally, LySAT exploits a new dynamic subsumption technique for Boolean CNF formulae[5]. It detects during conflict analysis, clauses that can be reduced by subsumption. During the learnt clause derivation, and at each step of the resolution process, the solver checks for backward subsumption between the current resolvent and clauses represented in the implication graph. This dynamic subsumption approach gives rise to a strong and dynamic simplification technique which exploits learning to eliminate literals from the original clauses.

Code

The system is written in C++ and has about 2000 lines of code. It is written on top of minisat 2.02 [3]. SatElite is systematically applied as a pre-processor [2].

References

1. Gilles Audemard, Lucas Bordeaux, Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. A generalized framework for conflict analysis. In *11th International Conference on Theory and Applications of Satisfiability Testing - SAT'2008*, volume 4996 of *Lecture Notes in Computer Science*, pages 21–27. Springer, 2008.

2. Niklas Eén and Armin Biere. Effective preprocessing in sat through variable and clause elimination. In Fahiem Bacchus and Toby Walsh, editors, *SAT*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.
3. Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.
4. Daniel Frost and Rina Dechter. In search of the best constraint satisfaction search. In *AAAI*, pages 301–306, 1994.
5. Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. Learning for dynamic subsumption. In *ICTAI*, pages 328–335. IEEE Computer Society, 2009.
6. Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In João Marques-Silva and Karem A. Sakallah, editors, *SAT*, volume 4501 of *Lecture Notes in Computer Science*, pages 294–299. Springer, 2007.