BreakIDGlucose 2

Jo Devriendt
University of Leuven
Leuven, Belgium

Bart Bogaerts
University of Leuven
Leuven, Belgium

***Abstract*—The defining characteristic of the BreakIDGlucose 2 SAT solving system is the addition of a preprocessing step in which symmetry breaking clauses are added to the CNF theory. It is meant to be an improvement on the 12 years old symmetry breaking preprocessor Shatter.**

I. Introduction

Many real-world problems exhibit symmetry, but the SAT competition and SAT challenge seldomly feature solvers who are able to exploit symmetry properties. This discrepancy can be explained by the assumption that for most of the problems in these competitions, symmetry exploitation is not worth the incurred overhead.

Two years ago we tested this hypothesis by submitting two symmetry breaking SAT solvers to the SAT competition. The first was ShatterGlucose, a coupling of the symmetry breaking preprocessor Shatter [1] with Glucose 2.2 [2] (which belongs to the family of MiniSat-based solvers [3]). The other was BreakIDGlucose, which instead of Shatter used our own symmetry breaking preprocessor BreakID [4] specialized in detecting and breaking row interchangeability symmetry subgroups from the symmetry group inferred by Saucy [5].

The result was surprising: BreakIDGlucose took the gold medal in the SAT+UNSAT track, showing that symmetry breaking is still relevant in today's SAT competition. However, BreakID was only a proof of concept, with some unelegant implementation choices and suboptimal symmetry detection algorithms. To improve on this, we developed BreakID2, use version 4.0 of Glucose as backend solver, and will participate in the 2015 SAT-Race.

II. Main Techniques

The workflow of BreakIDGlucose is straightforward:

1) BreakID2 uses Saucy to detect symmetry on a CNF file, and adds symmetry breaking clauses to the CNF theory.
2) Glucose 3.0 solves the resulting CNF.

III. Main Parameters

The main user-provided parameters control:

- How much time should be allocated to symmetry detection. For a 3600 second time limit per instance Saucy gets 300 seconds for symmetry detection. The symmetry breaking routine then continues with any symmetry detected by Saucy to construct symmetry breaking clauses.
- How large the symmetry breaking sentences are allowed to grow, measured in the number of auxiliary variables introduced by a symmetry breaking formula. We limit this to 100 variables.

IV. Special Algorithms, Data Structures, and Other Features

The symmetry breaking preprocessor BreakID2 differs from the original BreakID (dubbed BreakID1 for the remainder of this text) in three regards:

- A more tightly coupled integration of Saucy into the symmetry breaking routine. Saucy source code is integrated in BreakID2, resulting in a more simple tool chain.
- Row interchangeability symmetry detection is done by repeated calls to Saucy to construct row-swapping symmetries. This is an improvement to the naive symmetry enumeration approach used by BreakID1.
- BreakID2 constructs special binary symmetry breaking clauses, which potentially break much symmetry at little cost.

These improvements come on top of the improvements made by BreakID1 compared to Shatter:

- Completely breaking any detected row interchangeability symmetry group.
- Constructing a more suitable variable ordering with which to break symmetry.
- Using a smaller encoding for the symmetry breaking clauses.

Lastly, Saucy requires a slightly cleaned CNF as input, so the BreakID preprocessor also employs a small preprocessing step:

- Removing duplicate and tautological clauses from the CNF theory.

V. Implementation details

BreakID2 was written from scratch in C++. We refer to the webpages of the other programs for their implementation details.

VI. Acknowledgements

We would like to thank

1) Paul T. Darga, Mark Liffiton and Hadi Katebi for providing the source code of the symmetry detection tool Saucy.
2) Laurent Simon and Gilles Audemard for making their SAT solver Glucose available for use in the 2015 SAT-Race.

REFERENCES

[1] F. A. Aloul, K. A. Sakallah, and I. L. Markov, "Efficient symmetry breaking for Boolean satisfiability," *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 549–558, 2006.

[2] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *IJCAI*, C. Boutilier, Ed., 2009, pp. 399–404.

[3] N. Eén and N. Sörensson, "An extensible SAT-solver," in *SAT*, ser. LNCS, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Springer, 2003, pp. 502–518.

[4] J. Devriendt, B. Bogaerts, and M. Bruynooghe, "BreakIDGlucose: On the importance of row symmetry," in *Proceedings of the Fourth International Workshop on the Cross-Fertilization Between CSP and SAT (CSPSAT)*, 2014.

[5] H. Katebi, K. A. Sakallah, and I. L. Markov, "Symmetry and satisfiability: An update," in *SAT*, ser. LNCS, O. Strichman and S. Szeider, Eds., vol. 6175. Springer, 2010, pp. 113–127.