Riss 7

Norbert Manthey

***Abstract*—The sequential SAT solver RISS combines the Minisat-style solving engine of GLUCOSE 2.2 with a state-of-the-art preprocessor COPROCESSOR and adds many modifications to the search process. RISS allows to use inprocessing based on COPROCESSOR. Most recent changes focus mainly in incremental solving.**

I. INTRODUCTION

The CDCL solver RISS is a highly configurable SAT solver based on MINISAT [1] and GLUCOSE 2.2 [2], [3]. Besides many search algorithm extensions, RISS is equipped with the preprocessor COPROCESSOR [4]. The solvers supports, emitting DRAT proofs for many techniques, enumerating more than a single model, and incremental solving.

This document mentions only the differences to RISS 6 that has been submitted to SAT Competition 2016. Most differences are motivated by axiom pinpointing with SAT [5] that involves incremental SAT solving with many calls to the solver, where each single call has many assumption literals, and only a few conflicts.

II. MODIFICATIONS OF THE SEARCH

RISS 6 used automatic configuration based on formula features. This capability has been dropped.

III. INCREMENTAL SAT SOLVING WITH RISS

RISS 6 used simplification that have been applied lazily during calls to the solver. These simplifications have been disabled.

A. Persistent Incremental Solving

From a generic point of view, the idea of persistent incremental calls is to follow the following rules:

1) do not clear the trail after stopping search
2) in case new clauses are added, integrate them after making sure there are two literals that are not falsified
3) when being called with a new set of assumptions, extract the common prefix, and re-use this part of the trail

This way, parts of the search can be saved, because the trail has not to be revisited each time. Together with the IPASIR interface, benefits of this technique are not very strong, as new assumptions have to be passed for each call to the solver each time. When RISS is tightly integrated into an application, e.g. SATPIN [5], the effect is much more visible.

B. Applying Techniques Partially

If assumptions are given, RISS performs restarts only to the decision level where the last assumption was used as decision. This saves redundant work, as the first part of the trail in the solver stays fixed with assumptions (assuming the order of assumptions is not switched during search).

The original implementation of MINISAT 2.2 executed search under assumptions until the decision literal that should be assigned next is falsified already. In RISS we implement *Early Refined Cores* [6], which starts conflict analysis already if a conflict is found on a decision level that was reached based on assumptions only. This will trigger conflict analysis faster, but might produce longer conflict clauses.

C. Reverse Core Refinement

After a set of assumptions A has been found to be infeasible, a conflict clause C is produced. Reverse core refinement [6] tries to produce a smaller clause $D \subseteq C$ by re-running a call to incremental search and using $\neg C$ as assumptions – most importantly in the reverse order of the initial assumptions.

IV. SAT COMPETITION SPECIFICS

RISS and COPROCESSOR are implemented in C++. RISS is submitted to all sequential tracks, except the *random SAT* track.

V. AVAILABILITY

All tools in the solver collection are available for research. The source of RISS will be made publicly available under the LGPL v2 license at https://github.com/nmanthey/riss-solver.

ACKNOWLEDGMENT

The author would like to thank the developers of GLUCOSE 2.2 and MINISAT 2.2.

REFERENCES

[1] N. Eén and N. Sörensson, "An extensible SAT-solver," in *SAT 2003*, ser. LNCS, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Heidelberg: Springer, 2004, pp. 502–518.

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

[3] ——, "Refining restarts strategies for sat and unsat," in *CP'12*, 2012, pp. 118–126.

[4] N. Manthey, "Coprocessor 2.0 – a flexible CNF simplifier," in *SAT 2012*, ser. LNCS, A. Cimatti and R. Sebastiani, Eds., vol. 7317. Heidelberg: Springer, 2012, pp. 436–441.

[5] N. Manthey, R. Peñaloza, and S. Rudolph, "Efficient axiom pinpointing in EL using SAT technology," in *Proceedings of the 29th International Workshop on Description Logics, (DL 2016)*, ser. CEUR Workshop Proceedings, M. Lenzerini and R. Peñaloza, Eds., vol. 1577. CEUR-WS.org, 2016. [Online]. Available: http://ceur-ws.org/Vol-1577/paper_33.pdf

[6] N. Manthey, "Refining unsatisfiable cores in incremental SAT solving," TU Dresden, Tech. Rep., September 2015.