The CryptoMiniSat-4.4 set of solvers at the SAT Race 2015

Mate Soos, Marius Lindauer

I. Introduction

This paper presents the conflict-driven clause-learning SAT solver CryptoMiniSat v4.4 (*CMS4.4*) as submitted to SAT Race 15. CMS4.4 aims to be a modern, open-source SAT solver that allows for multi-threaded in-processing techniques while still retaining a strong CDCL component. In this description only the features relative to CMS4.4, the previous year's submission, are explained. Please refer to the previous years' description for details. In general, CMS4.4 is a in-processing SAT solver that usues optimized datastructures and finely-tuned timeouts to have good control over both memory and time usage of simplification steps.

A. Using watchlists as occurrence lists

As per lingeling [1], CMS4.4 now uses the watchlist to store occurrence lists (when they are needed) and related occurrence information such as data related to looking for XOR clauses or gates. This significantly reduces the memory overhead and, due to cache locality, increases speed.

B. Removal of uneeded code

Over the years, many lines of code has been added to CMS that in the end didn't help and often was detrimental to both maintinability and efficiency of the solver. Many such additions have now been removed. This simplifies understanding and developing the system. Further, it allows the system to be more lean especially in the tight loops such as propagation and conflict analysis where most of the time is spent.

C. Integration of ideas from SWDiA5BY A26

Some of the ideas from SWDiA5BY A26[2] have been included into CMS. In particular, the clause cleaning system employed and the switching restart have both made their way into CMS. Further, SWDiA5BY A26 was used as a testbed against CMS to clean up the codebase from unwated and unneeded elements.

D. Incremental solving

Incremental solving for a in-processing solver is not trivial and many bugs have been found in fuzzing the incremental solving interface. The fuzzer developed for this purpose contains more than 1000 lines of python and allows for testing both the incremental and the DRAT [3] interface of the solver.

E. Auto-tuning

The version 'autotune' reconfigures itself after about 160K conflicts. The configuration picked is one of 13 different setups that vary many different parameters of the solving such as learnt clause removal strategy, restart strategy, and in-processing strategies. CMS4.4 was run on all SAT Comp'09 + 11 + 13 problems with all configurations, extracting relevant information from the all problems after they have been solved and simplified for 160K conflicts. The information extracted and the top 5 best configurations were then given to a machine learning algorithm (C5.0[4]) which built a decision tree from this data. This decision tree was then translated into C++ and compiled into the CMS4.4 source code.

This work was carried out by the first author through a script for Amazon Web Services for reliably running any setup ($>$ 1500 lines of python), a script for extracting and sanitizing the parameters ($>$ 500 lines of python), and a script for translating the rule-based output of C5 into C++.

References

[1] Biere, A.: Yet another local search solver and lingeling and friends entering the sat competition 2014. In: SAT Competition 2014 Booklet. (2014)

[2] Oh, C.: MiniSat HACK 999ED, MiniSat HACK 1430ED and SWDiA5BY. In: SAT Competition 2014 Booklet. (201)

[3] Wetzler, N., Heule, M., Hunt, WarrenA., J.: Drat-trim: Efficient checking and trimming using expressive clausal proofs. In Sinz, C., Egly, U., eds.: Theory and Applications of Satisfiability Testing – SAT 2014. Volume 8561 of Lecture Notes in Computer Science. Springer International Publishing (2014) 422–429

[4] Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)