Sat7 – Engineering a Modular SAT-Solver

Christian Kern, Mohammad Khaleghi, Stefan Kugele, Christian Schallhart, Michael Tautschnig, and Andreas Weis

Institut für Informatik, Technische Universität München

As many other SAT-solvers are developed in an artful but monolithic style, we took interest in the question whether it is possible to design and implement a modular SAT-solver in a well-engineered and modular way. In particular, we are developing a framework which allows to exchange and adapt various components of the overall SAT-solver in order to match the requirements of particular problem instances.

The analysis and copy-implementation of a preexisting and successful SAT-solver was a natural starting point for our project. We chose MINISAT 1.14 [1] as guiding example, since MINISAT is an award-winning, yet compact solver.

The reengineering of MINISAT resulted in an algorithmically equivalent solver which is decomposed into a number of orthogonal and exchangeable components. In the process of analyzing MINISAT and developing its equivalent but modular sibling, MINISAT7, we found a number of approaches for potential improvements.

Thus, once MINISAT7 was running twice as long as MINISAT in the worst case, we started a branch from the faithful copy in order to develop our own solver SAT7. However, due to the lack of time, we were unable to pursue the implementation and testing of most of our ideas.

Consequently, we submit SAT7, a variant of MINISAT which is

- completely reengineered and reimplemented in C++ from scratch,
- decomposed into a number of exchangeable components, and features
- an alternative activity value initialization, akin to the approach of ACTIN [2].

The decomposition leads to the following components: `Analysis` (conflict-driven learning), `Assignment` (storage of the assumptions), `Decision` (choosing the next assignment), `Database` (stores the instance clauses as well as the learnt clauses), `Propagation` (propagates the enqueued facts), `Preprocessing` (performs an initial preprocessing and analysis of the instance), and `DPLL` (the overall algorithm)

During the initialization, we count the variable occurrences and then use these counts to seed the activity values, in contrast to the original MINISAT code, where they are initialized to 0. Thereby we follow an approach which is simpler than that of ACTIN, but nevertheless, we could achieve a proper speedup on our benchmark suites.

References

1. Eén, N., Sörensson, N.: An Extensible SAT-solver. Notes in Computer Science **2919** (2004) 502–518
2. Kibria, R.H.: Actin. Technical report, Darmstadt, Germany (2006)