Implementation and Evaluation of a Parallel Satisfiability Solver

For further details please contact supervisor Nils Timm

Department of Computer Science, University of Pretoria, South Africa
ntimm@cs.up.ac.za

Abstract. Satisfiability (SAT) solvers are tools for deciding whether there exists a satisfying assignment for an input propositional logic formula or not. Modern solvers are based on the search for a satisfying assignment by incrementally selecting unassigned variables, assigning them by either true or false, and propagating the resulting constraints to the clauses of the formula. In case the solver decisions lead to an unsatisfied clause, a so-called conflict clause is learned and added to the formula. Moreover, the solver tracks back by revising a former assignment decision and continuing the search from this point until a satisfying assignment is found or the search space is entirely explored. Learned conflict clauses enable to narrow the search space and are thus crucial for the runtime performance of SAT solving.

In parallel SAT solving multiple solvers run simultaneously and operate on different formulas. In this setting learned conflict clauses can also be shared. If a solver that operates on a formula A learns a conflict clause based on a subformula B of A, then all solvers that operate on formulas that also contain B can reuse the learned clause for pruning their search space. Thus, the solvers profits from each others learned clauses which accelerates the overall solving time.

The goal of this project is to implement such a parallel SAT solving tool. The implementation does not have to start from scratch. It shall make use of the existing Java-based SAT solver SAT4J. Given a set of propositional logic formulas as input, a set of solver threads shall be generated and executed in parallel such that each input formula is processed by one solver thread. Clause learning is a concept that is already included in SAT4J. In this project, a method for deciding whether a clause learned for a certain formula can also be shared with certain other formulas shall be additionally implemented. Finally, it shall be experimentally evaluated to which extent the runtime performance of SAT solving can profit from parallelisation and clause sharing.