Mucsat

Nicolas Rachinsky and Klaus Aehlig

Ludwig-Maximilians-Universität München

Abstract. Mucsat is a DLL-based solver for the satisfiability problem. It uses coroutines in order explore different parts of the search tree in parallel. In this way clauses learned in one part of the search tree can be used in the other, and vice versa.

Mucsat grew out of an extended homework of a lecture on SAT algorithms. It implements the DLL search procedure and uses the following well-known algorithms.

- Unit propagations (based on watched literals),
- special handling of two-clauses via lookup lists per variable,
- conflict analysis and learning of clauses (the first-UIP strategy is implemented),
- a forget strategy of learned clauses, based on the number of unset literals,
- restarts, and
- a variable state independent choice heuristic.

One aim of mucsat was to investigate the use of "cooperative multithreading", that is, coroutines, in SAT solvers. Based on a common clause database, several copies of lists of variable settings and watched literals are kept, one for each "thread". Each thread individually tries to solve the problem, but learned clauses are inserted into a common database. The other threads are informed of newly learned clauses via a queue.

The "incorporation process" is performed whenever a free choice of a variable would otherwise be required, but not yet incorporated clauses exist. Then one such clause is taken and watched literals are set appropriately; if this clause is identified as a unit clause, or if the incorporation causes a conflict, then this event is handled in the usual way. Otherwise the next clause is incorporated.

The switching of "threads" in our single process is not preemptive, but cooperative, that is, every thread by itself passes the control to the next thread. Therefore no additional synchronisation overhead is needed (except for the cost that the processor cache is used less efficiently at the times of thread changes).