Solving SAT by Counting the Subsumed Full Length Ordered Clauses

Gábor Kusper, Csaba Biró, Gergely Kovásznai

Eszterházy Károly College, Eger
Institute of Mathematics and Informatics

Email: see http://fmv.ektf.hu/

We introduce the Counting Subsumed Full Length Ordered Clauses algorithm, for short CSFLOC. It counts those full length clauses, which are subsumed by the input SAT problem. Full length clauses are n-clauses, where n is the number of variables in the input problem. A SAT problem is satisfiable if it does not subsume all n-clauses. We use ordered clauses where each variable has an index from 1 to n and the literals of a clause are ordered according to this indexing, where the first literal has the smallest index, and the last literal has the biggest one. The idea is that in an n-clause each of n variables is present either as a positive literal or as a negative one. So we can represent an ordered full length clause by an n bit integer. The algorithm works as follows:
(Input) S, a SAT problem given as a clause set; n, the number of variables of S.
(Output) M, a model for S if it satisfiable, or null, otherwise.
(Step 1.) It sets its counter to be 0.
(Step 2.) It converts the counter to an ordered full length clause, called C, where each bit is represented by a literal, the literal is positive if the corresponding bit is 1, negative otherwise.
(Step 3.) Let i be the index of the last positive literal in C, or 1 if there is no positive literal in C.
(Step 4.) Then it search for an ordered clause, say D, from S, which subsumes C, and the index of its last literal is greater than or equal to i, and the index of its last literal is the greatest among those clauses from S which subsume C.
(Step 5.) If no such clause is found, i.e., D is undefined, then a model is found for S, let M be the clause representation of negation of the counter. Goto (Stop).
(Step 6.) Otherwise, let k be the index of the last literal of D.
(Step 7.) Then it increases the counter by 2^{n-k}, which is the number of the consecutive ordered full length clauses which are subsumed by D.
(Step 8.) If the counter is 2^n, then S is unsatisfiable, let M be null. Goto (Stop).
(Step 9.) Goto (Step 2).
(Stop) Return M.
We show that this algorithm always stops and finds a model if there is one. We present a worst case time complexity analysis and lot of test results. CSFLOC is based on its earlier version, called CCC [1]. The new idea in CSFLOC is presented in (Step 3.), which states that in the best case the counter can be increased by 2^{n-i}, where i is the index of the last 1 bit in the counter. The test results show that this basic algorithm can outperform the well-known DPLL algorithm [2]. Our implementation can be downloaded and the reader is welcome to make a better solver out of it. We believe that this new algorithm could serve as a good basis for parallel algorithms, because its memory usage is constant and no communication is needed between the nodes.

References

[1] Gábor Kusper, Csaba Biró, Solving SAT by an Iterative Version of the Inclusion-Exclusion Principle. *SYNASC 2015*, DOI: 10.1109/SYNASC.2015.38, 189–190, 2015.

[2] M. Davis, G. Logemann, D. Loveland, A Machine Program for Theorem Proving. *Communications of the ACM*, vol. 5, 394–397, 1962.