

Size-Space Bounds and Trade-offs for CDCL Proofs

Jan Johannsen¹, Massimo Lauria², Jakob Nordström², and Marc Vinyals²

¹ Ludwig-Maximilians-Universität München

² KTH Royal Institute of Technology

Abstract. A long line of research has shown that conflict-driven clause learning viewed as a proof system can polynomially simulate resolution. We give a more explicit description of this CDCL proof system, which allows us to define proof size and space in a natural way and also capture aspects such as clause learning schemes and restart policies. This in turn makes it possible to study which upper and lower bounds on size, space and size-space trade-offs in general resolution carry over to proofs actually realizable by CDCL solvers.

State of the art SAT solvers implement the *Conflict Driven Clause Learning* (CDCL) algorithm. If we regard the execution trace of a solver on an unsatisfiable instance as a proof, the ensuing proof system is a subset of resolution.

An important question is whether CDCL can efficiently search for resolution proofs. Since it is unlikely that resolution is automatizable [1], we would still like to know how strong CDCL is if it does optimal choices. Equivalently, if we view CDCL with non-deterministic choices as a proof system the question is whether this proof system polynomially simulates resolution.

The answer is affirmative under almost reasonable assumptions [2, 5] — for technical reasons the solver needs to keep in memory all the clauses it learns. Furthermore, CDCL can solve formulas of width w in time $n^{O(w)}$ with high probability (over the choices of the solver).

We use the same model as [2, 5] but we give a more detailed description of the proof. While their proof system is based on the behaviour of the solver, we emphasize the resolution steps, making the properties of the proof explicit. This approach draws on ideas from [4].

This makes it possible to give clear and simple definitions of proof size and space that are close parallels of those in general resolution. It also allows us to describe clause learning schemes and restarts in an explicit way, and reason about the power of different heuristics.

We extract the resolution proof the solver generates and we annotate the proof with enough details of the internal state of the solver to ensure that each derivation step can actually be the outcome of a CDCL algorithm. The length of the proof now comes naturally as the length of the resolution proof, while the space corresponds to the number of clauses the solver had to keep in memory.

Previous approaches do not attempt to measure space, and in particular one of the key assumptions in the proofs that CDCL p-simulates resolution is that

the solver never forgets a clause that it learned. However, in practice learned clauses are pruned aggressively and some proofs can be done in less space if some clauses are learned repeatedly.

A second assumption is the ability of the solver to discard its position in the search space and *restart*. So a follow-up question is whether CDCL without restarts is as strong as CDCL. The reason why restarts are important in practice is not clear, which makes this question of additional interest.

For the following results, note that we would like lower bounds to apply to the most general setting and upper bounds to be feasible with heuristics used in practice.

Since we are still modelling the same algorithm, if we allow unlimited space and unlimited restarts then it is straightforward to verify that previous results about length for general resolution automatically translate into the CDCL proof system. Space lower bounds also follow immediately from general resolution.

We show that the trivial resolution upper bounds of exponential length and linear space respect to the number of variables holds for CDCL as well, even with any choice of asserting learning scheme and without restarts.

It is also immediate that lower bounds on size-space trade-offs carry over. For complete trade-offs we need upper bounds on each of the measures, but these are not so clear.

We have some preliminary results in this direction where we show upper bounds for both the size and space of pebbling formulas [3] even when we fix the learning scheme of the solver to 1UIP and we do not allow restarts.

References

1. Michael Alekhovich and Alexander A. Razborov. Resolution is not automatizable unless $W[P]$ is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, October 2008. Preliminary version appeared in *FOCS '01*.
2. Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, January 2011. Preliminary version appeared in *SAT '09*.
3. Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011. Full-length version available at <http://eccc.hpi-web.de/report/2010/125/>.
4. Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science*, 4(4:13), December 2008.
5. Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175:512–525, February 2011. Preliminary version appeared in *CP '09*.