The SAT Solver MXC, v.1

David R. Bregman and David G. Mitchell
Simon Fraser University
drb@sfu.ca, mitchell@cs.sfu.ca

July 2, 2006

Wherefor

MXC, in a sense, reflects an alternate interpretation of "SAT Race". The submitted solver was designed and implemented, completely from scratch, in 45 days. Implementation was done by the first author, currently a second-year undergraduate student, working part-time.[1]. The second author devoted about 1 day per week. The primary goal was a solver to support research based on [1], where problems are modelled using first- and second-order logic with inductive definitions, and solved by reduction to (extended) SAT. The first author registered for the SAT-Race because it sounded like a good challenge, but the work could not begin until the summer term. It still seemed a nice – but not obviously realistic – challenge to produce a first-rate solver in time for the SAT-Race. A conventional design was clearly called for, but we chose not to simply re-code an existing solver: Each design decision was weighed in turn, and often we tested multiple solutions. We did not quite manage a first-rate solver, but we did produce a respectable one which will serve our needs.

Solver Details

MXC is a fairly conventional solver in the Chaff/Berkmin/siege/MiniSat family. The solver does its' own memory management, has a separate binary clause database [3], and stores watched and binary-clause literals in linked lists with 31 literals per node. A bit-vector scheme is used for membership checks during conflict clause derivation. The version submitted for the second qualification round used the VMTF heuristic [4], and geometric restarts, but no clause deletion. For the final submission, we added a decision strategy similar to the MiniSat version of VSIDS [2], conflict clause minimisation as in MiniSAT [5], and clause deletion. There is no pre-processing.

References

[1] D. Mitchell and E. Ternovska. A framework for representing and solving NP search problems. In *Proc., AAAI-05*, pages 430–435, 2005.

[2] M. Moskewicz, C. Madigan, Y. Zhao, and L. Zhang. Chaff: Engineering an efficient sat solver. In *Proc., DAC2001*, June 2001.

[3] Slawomir Pilarski and Gracia Hu. Speeding up sat for eda. In *Proc., DATE 2002*, 2002.

[4] Lawrence O. Ryan. Efficient algorithms for clause learning sat solvers. Master's thesis, Simon Fraser University, Burnaby, Canada, 2004.

[5] N. Sorenssen and N. Een. MiniSat v1.13 - A SAT solver with conflict-clause minimization. 2005. SAT-2005 Poster.

[1]Perhaps under a generous notion of "part-time", but still concurrently taking a statistics course and leading a normal life.