Boolean Satisfiability Solvers: Techniques and Extensions

Sharad Malik
Princeton University, USA

Boolean Satisfiability (SAT) is the problem of checking if a propositional logic formula can ever evaluate to true. This problem has long enjoyed a special status in computer science. On the theoretical side, it was the first problem to be classified as being NP-complete. NP-complete problems are notorious for being hard to solve; in particular, in the worst case, the computation time of any known solution for a problem in this class increases exponentially with the size of the problem instance. On the practical side, SAT manifests itself in several important application domains such as the design and verification of hardware and software systems, as well as applications in artificial intelligence. Thus, there is strong motivation to develop practically useful SAT solvers.

However, the NP-completeness is cause for pessimism, since it is unlikely that we will be able to scale the solutions to large practical instances. While attempts to develop practically useful SAT solvers have persisted for almost half a century, for the longest time it was a largely academic exercise with little hope of seeing practical use. Fortunately, several relatively recent research developments have enabled us to tackle instances with millions of variables and constraints – enabling SAT solvers to be effectively deployed in practical applications including in the analysis and verification of software systems.

In the first part of this series of lectures, I will cover the techniques used in modern SAT solvers. In the second part, I will consider extensions of these solvers that have proved to be useful in analysis and verification. For instances that are unsatisfiable, the proofs of unsatisfiability have been used to derive an unsatisfiable subset of constraints of the formula, referred to as the UNSAT core. The UNSAT core has seen successful applications in model checking. Related to the UNSAT core are the concepts of minimal correction sets and maximally satisfiable subsets. A maximally satisfiable subset of an unsatisfiable instance is a maximal subset of constraints that is satisfiable, and a minimal correction set is a minimal subset of constraints that needs to be dropped to make the formula satisfiable. I will show how these concepts are related, present algorithms to derive them and show their application in design debugging.

References

[1] N. Eén, A. Biere. *Effective Preprocessing in SAT through Variable and Clause Elimination.* Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2005.

[2] N. Eén, N. Sörensson. *An Extensible SAT?solver.* Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2003.

[3] M. Davis, G. Logemann, D. Loveland. *A Machine Program for Theorem Proving.* Communications of the ACM, Vol. 5, pp. 394-397, 1962.

[4] M. Davis, H. Putnam. *A Computing Procedure for Quantification Theory.* Journal of ACM, Vol. 7, pp. 201-215, 1960.

[5] M. H. Liffiton, K. A. Sakallah. *On Finding All Minimally Unsatisfiable Subformulas.* Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT-2005), LNCS 3569, pp. 173-186, 2005.

[6] M. H. Liffiton, K. A. Sakallah. *Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints.* Journal of Automated Reasoning, 40(1), Springer, 2008.

[7] M. W. Madigan, C. F. Madigan, Y. Zhao, L. Zhang, S. Malik. *Chaff: Engineering an Efficient SAT Solver.* Proceedings of the 38th Conference on Design Automation (DAC '01), New York, 2001.

[8] J. P. Marques-Silva, K. A. Sakallah. *GRASP: a Search Algorithm for Propositional Satisfiability.* IEEE Transactions on Computers, 48(5), pp.506-521, 1999.

[9] B. Selman, H. Levesque, D. Mitchell. *A New Method for Solving Hard Satisfiability Problems.* Proceedings of the 10th National Conference on Artificial Intelligence (AAAI), pp. 440-446, 1992.

[10] J. Marques-Silva, J. Planes. *Algorithms for Maximum Satisfiability using Unsatisfiable Cores.* Proceedings of the Conference on Design, Automation and Test in Europe (DATE '08), 2008.

[11] G. Tseitin. *On the Complexity of Derivation in Propositional Calculus.* In: Studies in Constructive Mathematics and Mathematical Logic, part 2, pp. 115-125, 1968. Reprinted in J. Siekmann, G. Wrightson (eds), Automation of Reasoning, Vol. 2, pp. 466-483, Springer, 1983.

[12] L. Zhang, S. Malik. *Validating SAT Solvers using an Independent Resolution-based Checker: Practical Implementations and other Applications.* Proceedings of Design, Automation and Test in Europe (Conference and Exhibition), 2003.