Overview of automated deductions for checking satisfiability of propositional logic formulas

Lukas Vitroler

8. Juni 2012

Table of Contents

1 Introduction

Even the great Aristotle himself was bothered by the problem of satisfiability! This very short seminar paper written for the course Introduction to Scientific Working is about the problem of satisfiability and the various algorithms made throughout the years. It will show the reader a rough outline of the mentioned topic and since not a lot of space is given for this small overview, this paper will presume the reader already knows what the issue of satisfiability is. Due to the lack of space I will not describe how the different solvers work and only the most important aspects of the algorithms will be covered. The website SAT Live![1] has helped me a lot in finding the published literature that is available on the SAT subject. This paper is structured in a short description of the SAT problem, a general summary of SAT solvers and the explanation of how DPLL, CDCL and SLS work.

[1] http://www.satlive.org, June 3, 2012

2 Satisfiability (SAT) problem

The satisfiability problem is one of the oldest and most important problems known to mankind. The definition is as follows: You have a set of clauses[2] and have to look for an assignment of TRUE and FALSE values that makes every clause satisfying, or prove that such an assignment does not exist. It is NP complete which means that there currently exists no efficient algorithm that solves all instances of SAT and probably will never exist. However, since the contrary has not been proven yet, a lot of work and money is still put into the solution of this problem. Many areas of science reduce to SAT, such as logic synthesis, software verification, theorem proving, artificial intelligence and more. Therefore scientists have been creating and improving SAT solvers.

3 SAT Solvers

Throughout the years a lot of different ways to solve SAT have been created. Because SAT itself is already a NP complete problem and the existing solutions can be slow or very inefficient, it was decided to make the SAT problems smaller before letting them go through the algorithms. Thus you reduce them to a set of clauses, in this case CNFs. One of the solving algorithms created is resolution where you discard complementary literals and try to derive an empty clause.

With the emergence of computers the amount of automated deduction has increased enormously, however, the first creations were badly thought of and slow. Things started rolling when Davis and Putnam proposed applying resolution on CNFs. While a really efficient solver has not been found yet, the state-of-the-art solvers have become really fast and are now able to handle formulas with millions of variables and clauses. Credits towards that has to be given to the follow-ups of DPLL algorithms, CDCL or SLS.

3.1 Davis Putnam Logemann Loveland (DPLL) algorithm

DPP, the algorithm Davis and Putnam had created, had the problem that the algorithm required a lot of main memory and memory was very low back then so a new solution had to be found. DPLL is a stack-based divide-and-conquer implementation of the DPP created by Davis, Putnam, Loveland and Logemann and uses backtracking. But since it is similar to tree resolution it can become very inefficient because information acquired during the moving between branches is immediately thrown away. To resolve this there have been various improvements to the algorithm such as storing information in clauses or enabling backjumping.

[2] a finite disjunction of only AND, OR, NOT, variables and parentheses

3.2 Conflict Driven Clause Learning (CDCL) algorithm

CDCL is one of the algorithms resulting from the changes done to DPLL. Unlike its predecessor CDCL has many features DPLL didn't have: backjumping, clause learning, no recursion, conflict clause generation. These additions are pivotal for solving large SAT problems which exist in electronic design automation[1]. Since so many parts of sciences are depending on fast or efficient SAT solvers CDCL has enjoyed a lot of recognition. Modern CDCL solvers also possess lazy data structures.

3.3 Stochastic Local Search (SLS) algorithm

SLS is a completely different way to solve SAT problems as opposed to DPLL's resolution and CDCL's conflict driven approach. But this fact does not make the algorithm less effective. It has its uses outside of propositional logic too, the algorithm is well known for its methods in the creation of the Lin-Kernighan algorithm for the Traveling Salesman problem. Another two prominent SLS practices are ”Simulated Annealing and Evolutionary algorithms”.[2] Two instances of SAT solvers with SLS algorithms have been introduced at the same time in 1999 by Gu, Selman et al., and by Hansen, Jaumard and Minton, et al. Interestingly, neither group has known of the other's existence. The success of the solvers had sparked interest among the automated deduction community. Within a year many new versions of the solvers were created. SLS algorithms for SAT have helped in realising ”upper bounds on worst-case time complexity for solving SAT on k-CNF formulas”.[3]

4 Conclusion

In this paper I have outlined the problem of satisfiability and the different tools created to solve this problem, such as DPLL, CDCL and SLS. After all my investigating of the SAT solving problem I have found out that it really is a problem that has concerned humanity for centuries and will probably continue doing so as it is a NP complete problem. However, one thing that should be noted is that even though it might never be solved a lot of brave scientists have still tried and reduce the difficulties of the issue by creating not completely efficient algorithms to counter the challenge. It makes me glad to see that, although it is not a complete fix of the problem, solvers based on the algorithms of DPLL, CDCL and/or SLS have been built thus making the work of the scientists not for naught. The composition of this overview has motivated me to believe in the strength of science and humanity's thirst for knowledge.

[3] Holger Hoos. Handbook of Satisfiability. Chapter 1.23, 2009, page 42

Bibliography

[1] **Biere, Armin, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh**
Handbook of Satisfiability, volume 185 of Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009.

[2] **Hoos, Holger H. and Thomas Stützle**
Stochastic Local Search: Foundations and Applications (Morgan Kaufmann Series in Artificial Intelligence), Morgan Kaufmann, 1 edition, September 2004.

[3] **Marek, Victor W.**
Introduction to Mathematics of Satisfiability (Chapman and Hall/CRC Studies in Informatics Series), Chapman and Hall/CRC, 1 edition, September 2009.

[4] **Van Harmelen, Frank, Vladimir Lifschitz and Bruce Porter**
Handbook of Knowledge Representation (Foundations of Artificial Intelligence). Elsevier Science, 1 edition, January 2008.