SAT Solving

Jan Johannsen

Practical course, winter semester 2026/27

Overview

Introduction

Preliminaries

DPLL algorithms

CDCL solvers

Lookahead-based solvers

Certification

Overview

Introduction
Organization
Introduction and Motivation

Lecture and Tutorial

Lecture: Tuesdays 14:15 - 15:45 weekly in room 101 (Amalienstr. 17)

Exception: second lecture on Friday, October 16

Tutorial Fridays, 14:15 - 15:45 in room 101, on demand.

Exception: first tutorial on Tuesday, October 27

See detailed schedule on the homepage.

Projects

Three implementation projects to be completed, in groups of three students.

Project 1: Solving a puzzle game with a SAT solver.
Handed out: October 27, due date: November 27

Project 2: Implementing a simple DPLL solver.
Handed out: November 27, due date: January 15

Project 3: Implementing a CDCL solver.
Handed out: January 15, due date: TBD (mid March)

After each due date: oral presentation and code review.

The problem SAT

Problem k-SAT:

Given F in k-CNF, in n variables
find $\alpha \models F$, or report UNSAT

SAT is NP-complete $\rightsquigarrow$ (probably) no efficient (polynomial time) algorithm.

Trivial algorithm: exhaustive search in time $O(2^n \cdot |F|)$.

SETH: no algorithm in time $2^{o(n)} \cdot |F|$

Algorithms for SAT

Research since mid 1990s:

relatively efficient subexponential time algorithms for k-SAT.

Deterministic or probablistic: outputs $\alpha \models F$ w.h.p. if F is satisfiable.

Current best:

- time $O(1.307^n \cdot |F|)$ for 3-SAT
- time $O(1.469^n \cdot |F|)$ for 4-SAT

Current best deterministic alsorithms:

- time $O(1.331^n \cdot |F|)$ for 3-SAT
- time $O(1.5^n \cdot |F|)$ for 4-SAT

SAT Solver

Also since mid 1990s:
very good implementations and heuristics for backtracking-based algorithms (DPLL).

These SAT solvers

- solve instances with 100.000s of variables and millions of clauses
- are industry standard in hardware verification
- used in many other application domains
  - software verification
  - planning
  - combinatorics

SAT Solving

The topic of SAT solving has become a flourishing field of research, with

- annual international conference
  International Conference on Theory and Applications of Satisfiability Testing
- including annual competitions
- a scientific journal
  Journal on Satisfiability, Boolean Modeling and Computation
- run by an association

Certification

How can the output of a SAT solver be trusted?

In the positive case:
easily verified that the output assignment $\alpha \models F$

In the negative case?
solver outputs independently verifiable proof of unsatisfiability
required in competitions since 2013

Originally: resolution proof of unsatisfiability

Today: more concise proof format DRAT

Pythagorean triples

Pythagorean triple: $(a, b, c) \in \mathbb{N}^3$ with $a^2 + b^2 = c^2$

Pythagorean Triples Conjecture

Every 2-coloring of $\mathbb{N}$ contains a mono-chromatic pythagorean triple.

I.e., for all $\chi : \mathbb{N} \to \{r, b\}$ there exists a p.t. (a, b, c) with $\chi(a) = \chi(b) = \chi(c)$.

Formula PT_n in variables $x_1, \ldots, x_n$:

clauses $x_a \vee x_b \vee x_c$ and $\neg x_a \vee \neg x_b \vee \neg x_c$ for all p.t. $(a, b, c) \in \{1, \ldots, n\}^3$

$PT_{7.825}$ was shown unsatifiable by a special-purpose parallel SAT solver with 800 cores in 35.000 hours CPU time.

The generated DRAT proof has 200 TB.

Schur number 5

The n^{th} Schur number $s(n)$:

the maximal m such that $\{1, \ldots, m\}$ can be colored with n colors
without a mono-chromatic triple (a, b, c) with $a + b = c$

- $s(2) = 4$ is easily seen.
- $s(3) = 13$ can be found by hand with some effort.
- $s(4) = 44$ was shown in 1965 by Golomb and Baumert.

Heule (2018) showed $s(5) = 160$ with a special purpose parallel SAT solver.

The generated DRAT proof has 2 PB.