Mechanical Verification of SAT Solvers

Nathan Wetzler

The University of Texas at Austin

Project Proposal
CS 389R - Recursion and Induction
April 1, 2015

Motivation

Satisfiability (SAT) solvers are used in amazing ways...

- Hardware verification: Centaur x86 verification
- Combinatorial problems:
    - van der Waerden numbers
      [Dransfield, Marek, and Truszczynski, 2004]
    - Gardens of Eden in Conway's Game of Life
      [Hartman, Heule, Kwekkeboom, and Noels, 2013; Kouril and Paul, 2008]
- Unsatisfiability is often more important

Motivation

Satisfiability (SAT) solvers are used in amazing ways...

- Hardware verification: Centaur x86 verification
- Combinatorial problems:
  - van der Waerden numbers
    [Dransfield, Marek, and Truszczynski, 2004]
  - Gardens of Eden in Conway's Game of Life
    [Hartman, Heule, Kwekkeboom, and Noels, 2013; Kouril and Paul, 2008]
- Unsatisfiability is often more important

..., but satisfiability solvers have errors.

- Documented bugs in SAT, SMT, and QBF solvers
  [Brummayer and Biere, 2009; Brummayer et al., 2010]
- Competition winners have contradictory results
  (HWMCC winners from 2011 and 2012)
- Implementation errors often imply conceptual errors

Proposal

- Develop a model of a basic SAT solver
- Prove soundness of solver (SAT result)
- Prove completeness of solver (UNSAT result)

Satisfiability

Is there an assignment of values to variables such that a given Boolean formula evaluates to TRUE?

Formulas are in conjunctive-normal form (CNF).

$$
\begin{aligned}
&(x_1 \lor x_2 \lor \neg x_3) \land \\
&(\neg x_1 \lor \neg x_2 \lor x_3) \land \\
&(x_2 \lor x_3 \lor \neg x_4) \land \\
&(\neg x_2 \lor \neg x_3 \lor x_4) \land \\
&(x_1 \lor x_3 \lor x_4) \land \\
&(\neg x_1 \lor \neg x_3 \lor \neg x_4) \land \\
&(x_1 \lor \neg x_2 \lor \neg x_4) \land \\
&(\neg x_1 \lor x_2 \lor x_4)
\end{aligned}
$$

Satisfiability

Is there an assignment of values to variables such that a given Boolean formula evaluates to TRUE?

Formulas are in conjunctive-normal form (CNF).

```
CNF
 1  2 -3 0
-1 -2  3 0
 2  3 -4 0
-2 -3  4 0
 1  3  4 0
-1 -3 -4 0
 1 -2 -4 0
-1  3  4 0
```

Basic SAT Solver

```
Solve (f, a, h) =
  if eval(f, a) = true
    return (SAT, a)
  if empty(h)
    return (UNSAT, {})
  (s, m) = Solve(f, a U {top(h)}, pop(h))
  if (s == SAT)
    return (SAT, m)
  else
    return Solve(f, a U {¬top(h)}, pop(h))
```

Theorems

- Soundness

$$Solve(f, a, h) = \text{SAT}$$
$$\rightarrow \exists\, s : eval(f, s) = \text{TRUE}$$

- Completeness

$$\exists\, s : eval(f, s) = \text{TRUE}$$
$$\rightarrow Solve(f, a, h) = \text{SAT}$$

$$Solve(f, a, h) = \text{UNSAT}$$
$$\rightarrow \neg \exists\, s : eval(f, s) = \text{TRUE}$$

Timeline

- Week 1 - Model SAT problem and executable solver
  - Literals, Negation, Clauses, Formulas
  - Assignments, Evaluation, Heuristics
  - Basic solver algorithm
  - Satisfiability, Solutions
- Week 2 - Write main theorems, begin work on soundness
  - Theory of heuristics: subset, union, disjointedness
- Week 3 - Complete soundness proof
  - Proof by quantification
- Week 4 - Complete completeness proof
  - Proof by enumeration

Additional Work

- Model DP solver
- Model DPLL solver (unit propagation, pure literal elimination
- Prove DPLL solver sound and complete
- Model CDCL solver (learned clauses, conflict analysis, backtracking)