Nathan Wetzler
The University of Texas at Austin
Project Proposal
CS 389R - Recursion and Induction
April 1, 2015
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)
Is there an assignment of values to variables such that a given Boolean formula evaluates to TRUE?

Formulas are in conjunctive-normal form (CNF).
Is there an assignment of values to variables such that a given Boolean formula evaluates to TRUE?

Formulas are in conjunctive-normal form (CNF).

<table>
<thead>
<tr>
<th>CNF</th>
<th>1</th>
<th>2</th>
<th>-3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-2</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>-3</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-3</td>
<td>-4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-2</td>
<td>-4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Basic SAT Solver

\[
\text{Solve} \ (f, \ a, \ h) = \\
\quad \text{if} \ \text{eval}(f, \ a) = \text{true} \\
\quad \quad \text{return} \ (\text{SAT}, \ a) \\
\quad \text{if} \ \text{empty}(h) \\
\quad \quad \text{return} \ (\text{UNSAT}, \ \{\}) \\
(s, \ m) = \ \text{Solve}(f, \ a \ U \ \{\text{top}(h)\}, \ \text{pop}(h)) \\
\text{if} \ (s == \text{SAT}) \\
\quad \text{return} \ (\text{SAT}, \ m) \\
\text{else} \\
\quad \text{return} \ \text{Solve}(f, \ a \ U \ \{\neg \text{top}(h)\}, \ \text{pop}(h))
\]
Theorems

• Soundness

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

• Completeness

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