Practical SAT Solving
Lecture 10
Carsten Sinz, Tomáš Balyo | June 27, 2016
Lecture Outline

- Proof Generation
Proof Generation: Motivation

- For a satisfiable clause set, we can get an assignment $\alpha$ as a "certificate" on which we can check the correctness of the solution.
- What about unsatisfiable instances?

Resolution proof may serve as a witness/certificate, but the size of the resolution proof may be exponential in formula size. Efficient proof generation is important.

Additional uses include extraction of the unsatisfiable part of a formula ("unsatisfiable core"), interpolant generation, and computation of abstractions, e.g., in model checking (which will be covered later).
Proof Generation: Motivation

- For a satisfiable clause set, we can get an assignment $\alpha$ as a “certificate” on which we can check the correctness of the solution.
- What about unsatisfiable instances?
  - Resolution proof may serve as a witness / certificate.
Proof Generation: Motivation

- For a satisfiable clause set, we can get an assignment $\alpha$ as a "certificate" on which we can check the correctness of the solution.
- What about unsatisfiable instances?
  - Resolution proof may serve as a witness / certificate.
  - But size of resolution proof may be exponential in formula size.
  - Efficient proof generation important.
Proof Generation: Motivation

For a satisfiable clause set, we can get an assignment $\alpha$ as a “certificate” on which we can check the correctness of the solution.

What about unsatisfiable instances?
- Resolution proof may serve as a witness / certificate
- But size of resolution proof may be exponential in formula size
- Efficient proof generation important

Additional uses
- Extraction of unsatisfiable part of a formula (“unsatisfiable core”)
- Interpolant generation, computation of abstractions, e.g. in model checking (will be covered later)
Resolution Proofs for Learned Clauses

4f: \{ l, i, \lnot j \} 3x
4e: \{ \lnot k, \lnot l \} \{ i, \lnot j, \lnot k \} 3x
4d: \{ \lnot b, \lnot h, k \} \{ \lnot b, \lnot h, i, \lnot j \} 3x
4c: \{ f, i, j \} \{ \lnot b, f, \lnot h, i \} 2x
4b: \{ \lnot e, \lnot h, \lnot i \} \{ \lnot b, \lnot e, f, \lnot h \} 1x 1UIP clause: \{ \lnot b, \lnot e, f, \lnot h \}
Simple Proof Generation Algorithm

Approach

1. Run CDCL solver
2. For each learned clause store its resolution proof
3. Taking all partial resolution proofs (for each learned clause) together results in a proof for the original formula

Problems

- Not all learned clauses might be needed for the proof
- Care has to be taken about clause deletion
- Preprocessing techniques are not covered
- Proofs can become so large that they cannot be held in main memory (i.e. need to be dumped to disk)

Partial solution

- Mark each clause involved in generating a learned clause
- Only marked clauses are needed for the proof
Simple Proof Generation Algorithm

- **Approach**
  1. Run CDCL solver
  2. For each learned clause store its resolution proof
  3. Taking all partial resolution proofs (for each learned clause) together results in a proof for the original formula

- **Problems**
  - Not all learned clauses might be needed for the proof
  - Care has to be taken about clause deletion
  - Preprocessing techniques are not covered
  - Proofs can become so large that they cannot be held in main memory (i.e. need to be dumped to disk)
Simple Proof Generation Algorithm

Approach
1. Run CDCL solver
2. For each learned clause store its resolution proof
3. Taking all partial resolution proofs (for each learned clause) together results in a proof for the original formula

Problems
- Not all learned clauses might be needed for the proof
- Care has to be taken about clause deletion
- Preprocessing techniques are not covered
- Proofs can become so large that they cannot be held in main memory (i.e. need to be dumped to disk)

Partial solution
- Mark each clause involved in generating a learned clause
- Only marked clauses are needed for the proof
Proof File Formats

- TraceCheck ([http://fmv.jku.at/tracecheck](http://fmv.jku.at/tracecheck))

<table>
<thead>
<tr>
<th>CNF formula</th>
<th>TraceCheck proof</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p cnf 4 8</code></td>
<td>1 1 2 -3 0 0</td>
</tr>
<tr>
<td>1 2 -3 0</td>
<td>2 -1 -2 3 0 0</td>
</tr>
<tr>
<td>-1 -2 3 0</td>
<td>3 2 3 4 0 0</td>
</tr>
<tr>
<td>2 3 4 0</td>
<td>4 -2 -3 4 0 0</td>
</tr>
<tr>
<td>-2 -3 4 0</td>
<td>5 -1 3 -4 0 0</td>
</tr>
<tr>
<td>-1 -3 -4 0</td>
<td>6 1 3 4 0 0</td>
</tr>
<tr>
<td>1 3 4 0</td>
<td>7 -1 2 4 0 0</td>
</tr>
<tr>
<td>-1 2 4 0</td>
<td>8 1 -2 -4 0 0</td>
</tr>
<tr>
<td>1 -2 -4 0</td>
<td>9 -1 2 0 5 3 7 0</td>
</tr>
<tr>
<td></td>
<td>10 -1 0 5 4 2 9 0</td>
</tr>
<tr>
<td></td>
<td>11 2 0 3 6 1 10 0</td>
</tr>
<tr>
<td></td>
<td>12 0 4 6 8 11 10 0</td>
</tr>
</tbody>
</table>

- Lines:
  - `<clause id>`
  - `<learned or original clause> 0`
  - `<resolution derivation (by clause ids)> 0`
Advanced Techniques

- (D)RUP
- DRUP-trim
- (D)RAT
- DRAT-trim
Advanced Techniques: Basic Definitions

Reverse Unit Propagation (RUP)
Given a formula $F$ and a clause $C$, assign all literals in $C$ to false and perform unit propagation. If a conflict is generated (i.e. $F \cup \neg C \vdash_{UP} \bot$), then $C$ is a RUP clause (and redundant for $F$).

Asymmetrical Literal Addition (ALA)
Given a formula $F$ and a clause $C$, $ALA(F, C)$ is the unique clause obtained by repeatedly adding a literal $l$ to $C$ if there exists a clause $D \in F \setminus \{C\}$ with $D = D' \cup \{\neg l\}$ and $D' \subseteq C$.

Example: given $F = \{(x, \neg a)\}$ and $C = (x, y, z)$, $ALA(F, C) = (x, y, z, a)$. 
Resolution Asymmetric Tautology (RAT)

Given a formula $F$ and a clause $C$, $C$ has the RAT property on $l \in C$ with respect to $F$ if for all $D \in F$ with $\neg l \in D$, it holds that $F \cup \neg C \cup (\neg D \setminus \{l\}) \vdash_{UP} \bot$.

- If a clause $C$ has RAT on some $l \in C$ w.r.t $F$, then $F$ is satisfiability-equivalent to $F \cup \{C\}$.
- All current preprocessing and inprocessing techniques can be expressed in terms of addition and removal of RAT clauses.
Dissertation *Efficient, Mechanically-Verified Validation of Satisfiability Solvers* by N. D. Wetzler (UT Austin, 2015)

- Read pages 31–34 of the dissertation
- What is the RUP proof format?
- How can RUP proofs be verified?
- Discuss those questions in groups of two
Next Week...

- ... there will be no lecture.
- Instead, please
  - Read pages 44–49 (on DRUP) and pages 50–53 (on RAT) of Wetzler’s dissertation.
  - Answer the questions: What is DRUP? What is the RAT proof format? How can RAT proofs be verified? What are the benefits of DRUP and RAT over RUP?
  - We will discuss those questions at the beginning of the lecture on July 11.