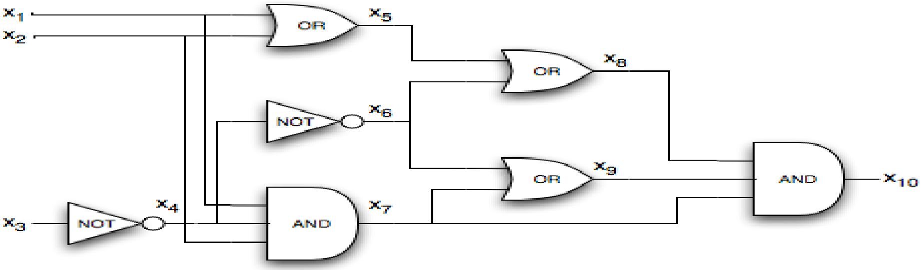
Practical SAT Solving

Lecture 10

Carsten Sinz, Tomáš Balyo | June 27, 2016

INSTITUTE FOR THEORETICAL COMPUTER SCIENCE



Lecture Outline

- Proof Generation

Proof Generation: Motivation

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

Proof Generation: Motivation

- For a satisfiable clause set, we can get an assignment α 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

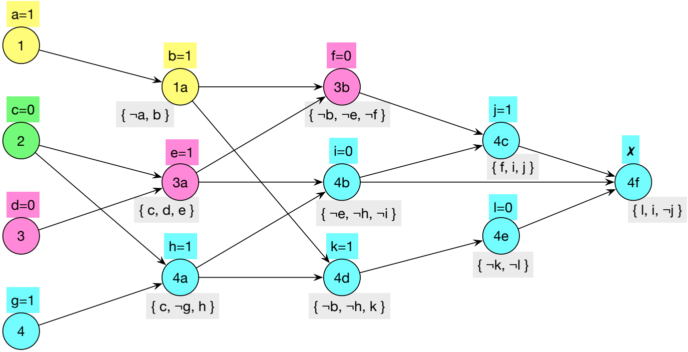
Proof Generation: Motivation

- For a satisfiable clause set, we can get an assignment α 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 α 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, ¬j }	3x	
4e:	{ ¬k, ¬l }	{ i, ¬j, ¬k }	3x	
4d:	{ ¬b, ¬h, k }	{ ¬b, ¬h, i, ¬j }	3x	
4c:	{ f, i, j }	{ ¬b, f, ¬h, i }	2x	
4b:	{ ¬e, ¬h, ¬i }	{ ¬b, ¬e, f, ¬h }	1x	1UIP clause: { ¬b, ¬e, f, ¬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

Simple Proof Generation Algorithm

Karlsruhe Institute of Technology

- 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`)

CNF formula

```
p cnf 4 8
 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 -2 -4 0
```

TraceCheck proof

```
1   1  2 -3 0                0
2  -1 -2  3 0                0
3   2  3 -4 0                0
4  -2 -3  4 0                0
5  -1 -3 -4 0                0
6   1  3  4 0                0
7  -1  2  4 0                0
8   1 -2 -4 0                0
9  -1  2    0 5 3 7          0
10 -1       0 5 4 2  9       0
11  2       0 3 6 1 10       0
12          0 4 6 8 11 10    0
```

- 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 $C = (x, y, z)$, $ALA(F, C) = (x, y, z, a)$.

Advanced Techniques: More Definitions

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.

Group Work

- Dissertation *Efficient, Mechanically-Verified Validation of Satisfiability Solvers* by N. D. Wetzler (UT Austin, 2015)
  - `http://www.cs.utexas.edu/users/nwetzler/publications/dissertation-preprint.pdf`
  - 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.