Trimming while Checking Clausal Proofs

Marijn J.H. Heule
Warren A. Hunt, Jr.
Nathan Wetzler

The University of Texas at Austin

Formal Methods in Computer-Aided Design (FMCAD)
Portland, Oregon
October 23, 2013
Outline

- Motivation and Contributions
- Resolution versus Clausal Proofs
- Checking Clausal Proofs Efficiently
- Experimental Evaluation
- Conclusion
SAT solvers are used in many tools and applications.
- Counter-examples (satisfiable) using symbolic simulation;
- Equivalence-checking (unsatisfiable) using miters;
- Small explanations (unsatisfiable core) for diagnosis;
- Small (trimmed) proofs to validate with a verified checker.
SAT solvers are used in many tools and applications.
- Counter-examples (satisfiable) using symbolic simulation;
- Equivalence-checking (unsatisfiable) using miters;
- Small explanations (unsatisfiable core) for diagnosis;
- Small (trimmed) proofs to validate with a verified checker.

However,
- Documented bugs in SAT, SMT, and QBF solvers
  [Brummayer and Biere, 2009; Brummayer et al., 2010];
- Solvers that emit additional information use lots of memory.
Motivation

SAT solvers are used in many tools and applications.
- Counter-examples (satisfiable) using symbolic simulation;
- Equivalence-checking (unsatisfiable) using miters;
- Small explanations (unsatisfiable core) for diagnosis;
- Small (trimmed) proofs to validate with a verified checker.

However,
- Documented bugs in SAT, SMT, and QBF solvers [Brummayer and Biere, 2009; Brummayer et al., 2010];
- Solvers that emit additional information use lots of memory.

We developed a tool that can efficiently validate the results of SAT solvers and produce trimmed formulas and trimmed proofs.
Contributions and Related Work

- **Easy to Emit**
- **Compact**
- **Checked Efficiently**

**Resolution Proofs**
- Zhang and Malik, 2003
- Van Gelder, 2008; Biere, 2008

**Clausal Proofs**
- Goldberg and Novikov, 2003
- Van Gelder, 2008

**Clausal proofs + clause deletion**
- Heule, Hunt, Jr., and Wetzler [STVR 201X]

**A fast clausal proof checker, called DRUP-trim**
- Heule, Hunt, Jr., and Wetzler [FMCAD 2013]
Contributions and Related Work

Easy to Emit

Compact

Checked Efficiently

Resolution Proofs
Zheng and Malik, 2003
Van Gelder, 2008; Biere, 2008

Clausal Proofs
Goldberg and Novikov, 2003
Van Gelder, 2008

Clausal proofs + clause deletion
Heule, Hunt, Jr., and Wetzler [STVR 201X]

A fast clausal proof checker, called DRUP-trim
Heule, Hunt, Jr., and Wetzler [FMCAD 2013]

All approaches can be used for applications such as minimal unsatisfiable core extraction, computing interpolants, reduce proofs
Given a Boolean formula $F$, is there an assignment to variables in $F$ such that the formula evaluates to TRUE?

\[
\begin{array}{c}
\overline{bc} \\
ac \\
\overline{ab} \\
\overline{a}b \\
ab
\end{array}
\]
Given a Boolean formula $F$, is there an assignment to variables in $F$ such that the formula evaluates to TRUE?

\[
\overline{bc} \quad ac \quad \overline{ab} \quad \overline{ab} \quad ab
\]

Checking a solution, such as assignment $\overline{a} \overline{b} c$, is easy.
Given a Boolean formula $F$, is there an assignment to variables in $F$ such that the formula evaluates to TRUE?

$\overline{bc} \ \overline{ac} \ \overline{âb} \ \overline{ab} \ ab$

Checking a solution, such as assignment $\overline{a} \overline{b} \ c$, is easy.

Unsatisfiability proofs use lemmas (resolvents):

$\overline{ab} \ \overline{bc}$

$\overline{ac} \ \ ac$

$\overline{bc} \ \ ac \ \ \overline{ab}$
Resolution Graph / Proof and Core

Resolution graph

bc ac ab ab ab bc

Trimming while Checking Clausal Proofs

Marijn J.H. Heule

Wednesday, October 30, 13
Resolution Graph / Proof and Core

Resolution graph

Resolution proof

Trimming while Checking Clausal Proofs

Marijn J.H. Heule

Wednesday, October 30, 13
Resolution Graph / Proof and Core

Resolution proof

Resolution graph
Resolution Graph / Proof and Core

resolution proofs are HUGE
A clause is **unit** with respect to an assignment if all literals in the clause are falsified except for one literal, which is unassigned.

Unit propagation:
- If a unit clause is found, extend the assignment and repeat.
- Else, return the assignment.
A clause is **unit** with respect to an assignment if all literals in the clause are falsified except for one literal, which is unassigned.

Unit propagation:
- If a unit clause is found, extend the assignment and repeat.
- Else, return the assignment.
A clause is **unit** with respect to an assignment if all literals in the clause are falsified except for one literal, which is unassigned.

Unit propagation:
- If a unit clause is found, extend the assignment and repeat.
- Else, return the assignment.

$$\overline{b}c \ \overline{a}c \ \overline{a}b \ \overline{a}\overline{b} \ ab$$

**assignment:** $\overline{c} \ \overline{b}$
A clause is **unit** with respect to an assignment if all literals in the clause are falsified except for one literal, which is unassigned.

**Unit propagation:**
- If a unit clause is found, extend the assignment and repeat.
- Else, return the assignment.

![Assignment: \( \tilde{c} \ \tilde{b} \ a \)]
A clause is **unit** with respect to an assignment if all literals in the clause are falsified except for one literal, which is unassigned.

Unit propagation:
- If a unit clause is found, extend the assignment and repeat.
- Else, return the assignment.

```
bc ac ab ab ab
```

Assignment: \( \bar{c} \bar{b} a \)

Reverse Unit Propagation (RUP) of a lemma:
- Assign all literals in the lemma to false and apply unit propagation.
- If another clause / lemma becomes falsified, then the lemma is valid.
Trimming while Checking Clausal Proofs

Marijn J.H. Heule

Wednesday, October 30, 13
Clausal Proof: Check using Unit Propagation

Trimming while Checking Clausal Proofs

Wednesday, October 30, 13
Clausal Proof: Check using Unit Propagation

Trimming while Checking Clausal Proofs

Wednesday, October 30, 13
Clausal Proof: Check using Unit Propagation

Trimming while Checking Clausal Proofs
Clausal Proof: Check using Unit Propagation

Trimming while Checking Clausal Proofs

Marijn J.H. Heule

Wednesday, October 30, 13
Clausal proofs are expensive to validate
Goldberg and Novikov proposed checking the refutation backwards [DATE 2003]:
- start by validating the empty clause;
- mark all lemmas using conflict analysis;
- only validate marked lemmas.

Advantage: validate fewer lemmas.

Disadvantage: more complex.

We provide a fast open source implementation of this procedure.
We proposed to extend clausal proofs with deletion information [STVR 201X]:
- clause deletion is crucial for efficient solving;
- emit learning and deletion information;
- proof size might double;
- checking speed can be reduced significantly.

Clause deletion can be combined with backwards checking [FMCAD 2013]:
- ignore deleted clauses earlier in the proof;
- optimize clause deletion for trimmed proofs.
We propose a new unit propagation variant:
1) propagate using clauses already in the core;
2) examine non-core clauses only at fixpoint;
3) if a non-core unit clause is found, goto 1);
4) otherwise terminate.

Our variant, called Core-first Unit Propagation, can reduce checking costs considerably.

Fast propagation in a checker is different than fast propagation in a SAT solver.

Also, the resulting core and proof are smaller.
Trimming while Checking Clausal Proofs

Checking: Backwards + Core-first + Deletion
Checking: Backwards + Core-first + Deletion

Trimming while Checking Clausal Proofs

Marijn J.H. Heule
Checking: Backwards + Core-first + Deletion

Trimming while Checking Clausal Proofs

Marijn J.H. Heule

Wednesday, October 30, 13
Trimming while Checking Clausal Proofs

Wednesday, October 30, 13
Trimming while Checking Clausal Proofs

Checking: Backwards + Core-first + Deletion
Trimming while Checking Clausal Proofs

Checking: Backwards + Core-first + Deletion
Checking: Backwards + Core-first + Deletion

Trimming while Checking Clausal Proofs

Marijn J.H. Heule
Checking: Backwards + Core-first + Deletion

Trimming while Checking Clausal Proofs

Marijn J.H. Heule

Wednesday, October 30, 13
Checking: Backwards + Core-first + Deletion

Core-first unit propagation results in smaller cores and proofs
We implemented DRUP logging into Glucose using only 40 LoC.

Glucose (DRUP) solves about twice as many benchmarks as compared to Picosat (resolution).

Resolution proof logging increased memory usage up to a factor 100.

DRUPtrim validated clausal proofs in a time similar to the solving time.
Unsatisfiable tracks required certificates. Allowed formats:
- TraceCheck (resolution);
- DRUP, Delete Reverse Unit Propagation.

Timeout: 5,000 seconds for solving and 20,000 seconds for checking

Submissions with proof logging:
- 11 application solvers (9 DRUP, 2 RUP);
- 9 hard-combinatorial solvers (7 DRUP, 2 RUP);
- Most submissions were certified unsatisfiable versions of top-tier solvers.

Statistics:
- 98% of DRUP proofs of top-tier solvers were checked within the time limit;
- Checking most RUP proofs (i.e., no clause deletion) results in a timeout.
Our **DRUPtrim** tool:

- makes it feasible to check the results of state-of-the-art solvers efficiently (demonstrated at SAT Competition 2013);
- validates learning, preprocessing, and inprocessing techniques;
- and produces trimmed proofs and trimmed formulas.

**Our next goal** is to increase confidence in all SAT solvers by efficiently checking proofs with a mechanically-verified proof checker.

**Discussion:** should UNSAT proof logging be mandatory for tools participating in competitive events (e.g., SAT Competition, HWMCC)?
Recent Work

Bridging the Gap Between Easy Generation and Efficient Verification of Unsatisfiability Proofs
Marijn J.H. Heule, Warren A. Hunt, Jr., and Nathan Wetzler
Accepted: Software Testing, Verification, and Reliability (STVR 201X)

Verifying Refutations with Extended Resolution
Marijn J.H. Heule, Warren A. Hunt, Jr., and Nathan Wetzler
Published: Conference on Automated Deduction (CADE 2013)

Mechanical Verification of SAT Refutations with Extended Resolution
Nathan Wetzler, Marijn J.H. Heule, and Warren A. Hunt, Jr.
Published: Interactive Theorem Proving (ITP 2013)

Trimming while Checking Clausal Proofs
Marijn J.H. Heule, Warren A. Hunt, Jr., and Nathan Wetzler
Accepted: Formal Methods in Computer-Aided Design (FMCAD 2013)

Thank you for your attention! Questions?