FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

A Tiny Tweak to Proof Generation in MiniSat-based SAT Solvers & A Complete and Efficient DRAT Proof Checker

Masterstudium:

Computational Logic

Johannes Altmanninger, October 4, 2019

Technische Universität Wien
Institute of Logic and Computation
Arbeitsbereich: Formal Methods in Systems Engineering
Betreuer: Associate Prof. Dipl.-Ing. Georg Weissenbacher, D.Phil.

Motivation

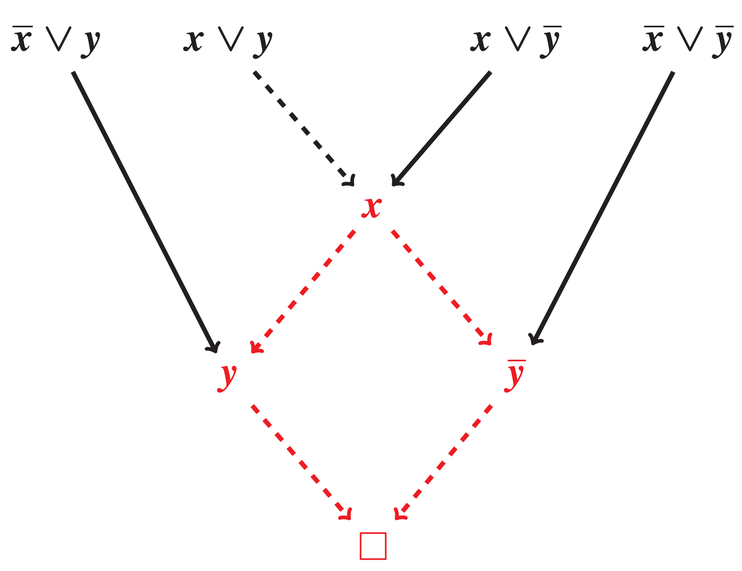
- SAT solvers produce DRAT proofs that are incorrect[a] due to spurious deletions of **unique reason clauses**
- proof checkers remedy this by ignoring unit deletions
- handling unique reason deletions requires a complicated algorithm

Problem

- unsatisfiable propositional formula $F = (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y})$
- redundant clauses (according to criteria RUP or RAT) can be added without affecting satisfiability of the formula
- proof of unsatisfiability adds and deletes clause until deriving the empty (unsatisfiable) clause $\square$

```
1:  add x        addition of redundant clause
2:  del x ∨ y    deletion of subsumed clause (always fine)
3:  del x        deletion of unique reason clause
4:  add □        addition of redundant conflict clause
```

- clause x is the **unique reason** for literal x
- deleting x removes derived clauses y, $\bar{y}$ and $\square$, making the proof incorrect
- many SAT solvers produce such proofs $\longrightarrow$ proof checkers ignore unit deletions



Contributions

- provide patches for SAT solvers to produce correct proofs
- extension of SICK incorrectness certificate, giving a counter-example for an incorrect proof
- implement efficient checker to measure performance impact of handling unique reason deletions

Avoiding Unique Reason Deletions in Solvers

- `DRUPMiniSat`-based solvers delete reason clauses but do not undo corresponding assignments
- remedy: emit a unit clause before deleting a reason clause — yields correct proofs that match the solver's behavior
- we implemented this in `DRUPMiniSat` patch shown below (reason clauses are called `locked`)
- similar patch for the 2018 SAT competition winner

```
@@ -632,9 +632,13 @@ void Solver::removeSatisfied(vec<CRef>& cs)
         Clause& c = ca[cs[i]];
+        if (output != NULL && locked(c)) {
+            Lit unit = c[0];
+            fprintf(output, "%i 0\n",
+                    (var(unit) + 1) * (-2 * sign(unit) + 1));
+        }
         removeClause(cs[i]);
```

SICK Format

- *small* artifact to *efficiently* certify incorrectness of a proof
- can be verified with our tool `sick-check`

incorrect DRAT proof for F:

```
1:  del x ∨ y
2:  add x
3:  add □
```

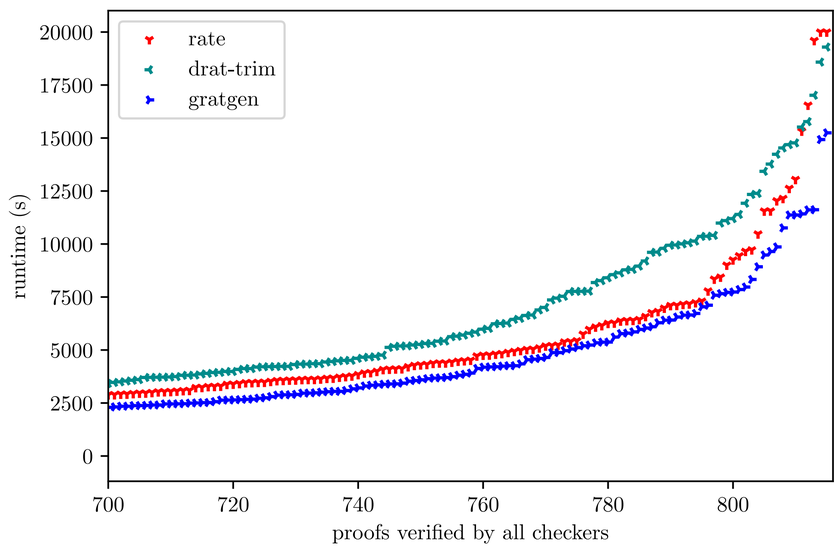
SICK certificate refuting RAT of clause x:

```
proof_format = DRAT-arbitrary-pivot
  proof_step = 2   (Failed line in the proof)
natural_model = {x̄, ȳ}
failing_clause = x̄ ∨ y
failing_model = {}
        pivot = x
```

Checker Implementation: `rate`

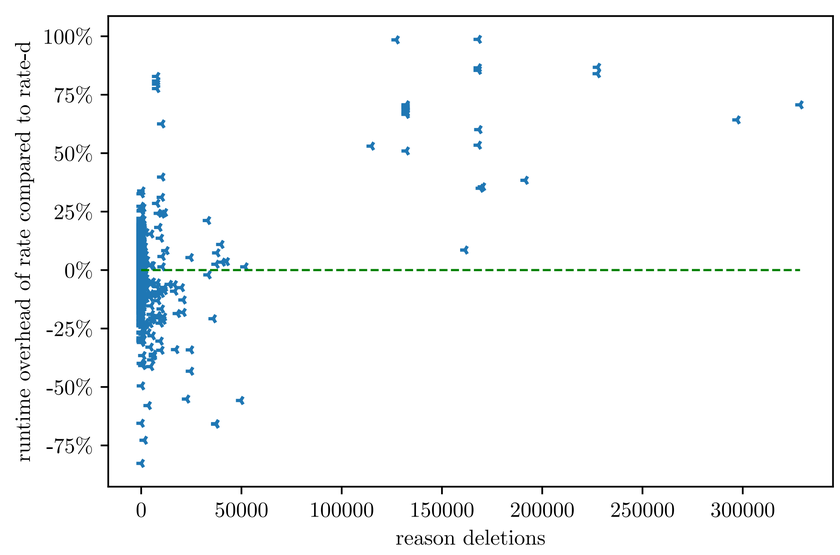
- efficiently handles reason deletions
- performance similar to best checkers (Figure 1)
- available at `https://github.com/krobelus/rate`
- written in Rust

Figure 1: Distribution of Proof Checkers' runtime



Insight: an excessive number of reason deletions may effect longer checking runtime (Figure 2). Find more details at `https://github.com/krobelus/rate-experiments`.

Figure 2: Overhead in seconds of handling reason deletions



[a] Due to space constraints, this poster assumes *"correctness"* of a DRAT proof to be defined in terms of *specified DRAT*, see *A. Rebola-Pardo and A. Biere, "Two Flavors of DRAT", Pragmatics of SAT, vol. 2018.*

Kontakt: aclopte@gmail.com