Complete DRAT Proof Checking

Johannes Altmanninger

November 13, 2019

Roadmap

- Introduction
- Problems
- Remedies
  - Solvers
  - Checkers
- Conclusion

DRAT Proofs

- Used for verifying SAT solvers' unsatisfiability results
- Log of clause introductions and deletions
- Proof checker can reproduce unsatisfiability
- Introduced clauses are redundant (RAT)

Unit Propagation

- Central reasoning technique in SAT solvers and checkers

  Example: $x \wedge \overline{x}y \vdash_1 x \wedge y$

- *Unit clause*: one non-falsified literal

  $\supseteq$ *Reason clause*: reason for satisfying some literal

  $\supseteq$ *Unique reason clause*: only possible reason

Deletion of unique reason clauses shrinks the set of literals implied by $\vdash_1$

Generating Clausal Proofs in CDCL Solvers

$F = xy \wedge \overline{x}y \wedge x\overline{y} \wedge \overline{x}\,\overline{y}$

$F \nvdash_1 \bot$

SAT solver needs to make a *decision*, for example x:

$F \wedge x \vdash_1 \bot$

Clause $\overline{x}$ is learned

$F \wedge \overline{x} \vdash_1 \bot$

Resulting DRAT Proof: (**add** x)

Two Flavors of DRAT

- *Operational DRAT* ignores deletions of unit clauses

  Implemented by state-of-the-art checkers

- *Specified DRAT* honors all deletions

Operational DRAT is much easier to implement *efficiently*

Motivation for Using Specified DRAT

- Operational DRAT is insufficient to verify proofs with arbitrary deletions, which are used to support advanced inprocessing techniques
- Computing properties of a proof is much less costly
- Conceptually simpler
- Disadvantage: checker implementation is more difficult (but solvers could still produce produce proofs that are correct in either flavor)

Research Question

Is it possible to check specified DRAT as efficiently as operational DRAT?

Current status

- Many DRAT proofs incorrect under specified DRAT
- Legit proofs might be incorrect under operational DRAT (when relevant deletions are ignored)

Incorrectness Scenarios

- Specified DRAT:
  - Some unique reason clause is *deleted*
  - Proof fails due to **absence** of clause
- Operational DRAT:
  - Deletion of unique reason clause is *not deleted*
  - Proof fails due to **presence** of clause (c.f. non-monotonicity of RAT)

Making Proofs Conform to Specified DRAT

- Prerequisite for adoption of specified DRAT
- Allows to detect rejections of valid proofs under operational DRAT

Most solvers' proofs can be verified using operational DRAT just fine — their proofs are correct after dropping any unit deletions

Why are so many proofs incorrect under specified DRAT?

DRUPMinisat-based solvers delete unique reason clauses that are still used to show unsatisfiability

As an operational checker ignores those deletions, verification succeeds regardless

Why generate the deletions in the first place? We provide patches to avoid them!

DRUPMinisat Problematic Deletion Sketch

$F = xyz \wedge \overline{x}yz \wedge x\overline{y}z \wedge \overline{x}\,\overline{y}z \wedge \overline{z}$

Clause $\overline{z}$ is satisfied, so it is deleted during simplification

Resulting DRAT Proof: (**del** $\overline{z}$, **add** x)

Correct under operational DRAT, but incorrect under specified DRAT

Checking Specified DRAT

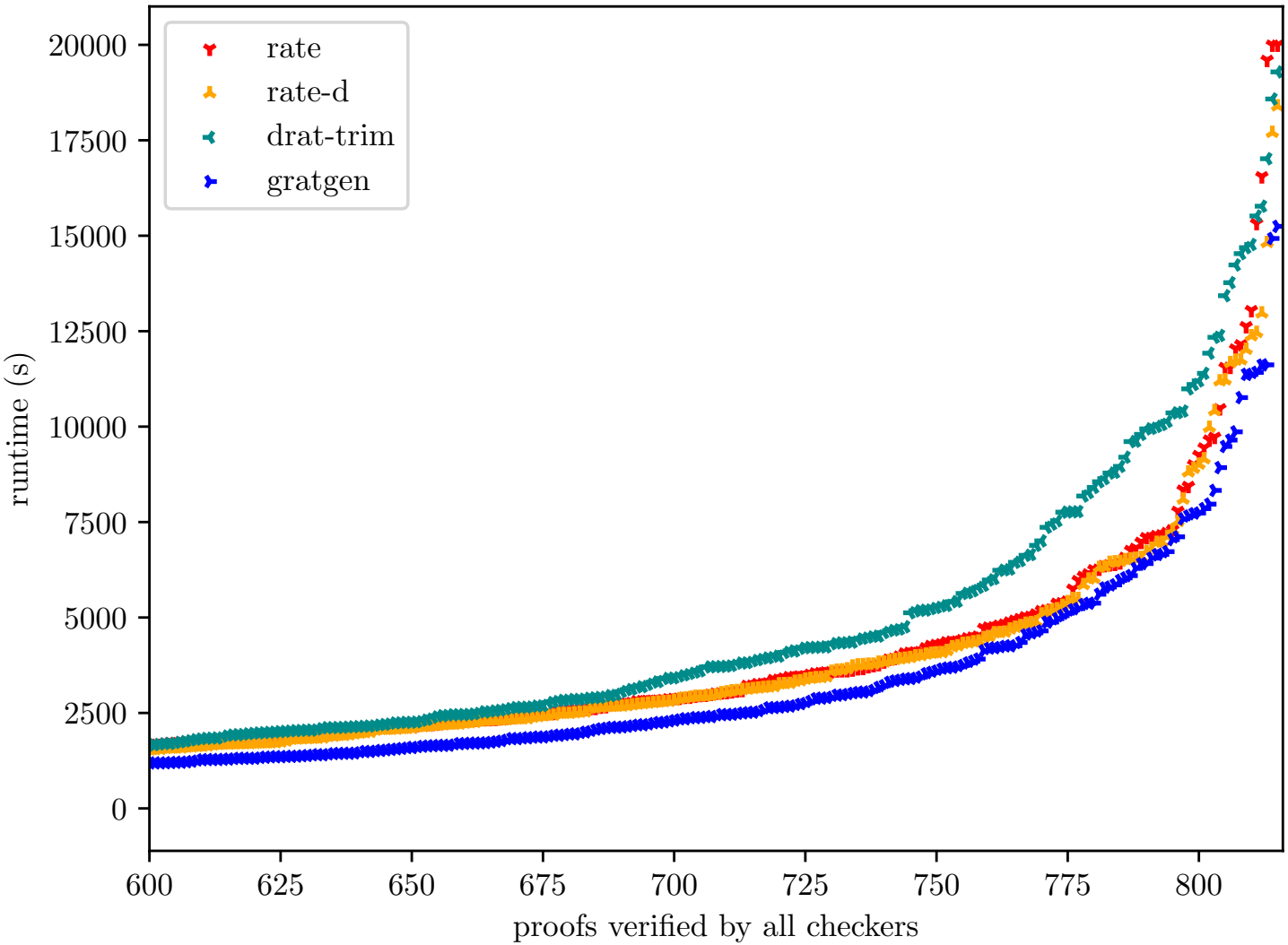
- Need to implement deletions that actually discard information
- Non-trivial when combined with other optimizations like backwards checking
- Efficient algorithm exists, but implementations were not competitive

Our checker: `rate`

aka "`rate` ain't trustworthy either"

- MIT licensed, openly developed DRAT proof checker https://github.com/krobelus/rate
- Supports specified and operational DRAT
- Detects deletions of unique reason clauses
- Competitive performance
- "seems like a nice piece of work, much-much nicer to read than drat-trim"

`rate` vs. other checkers

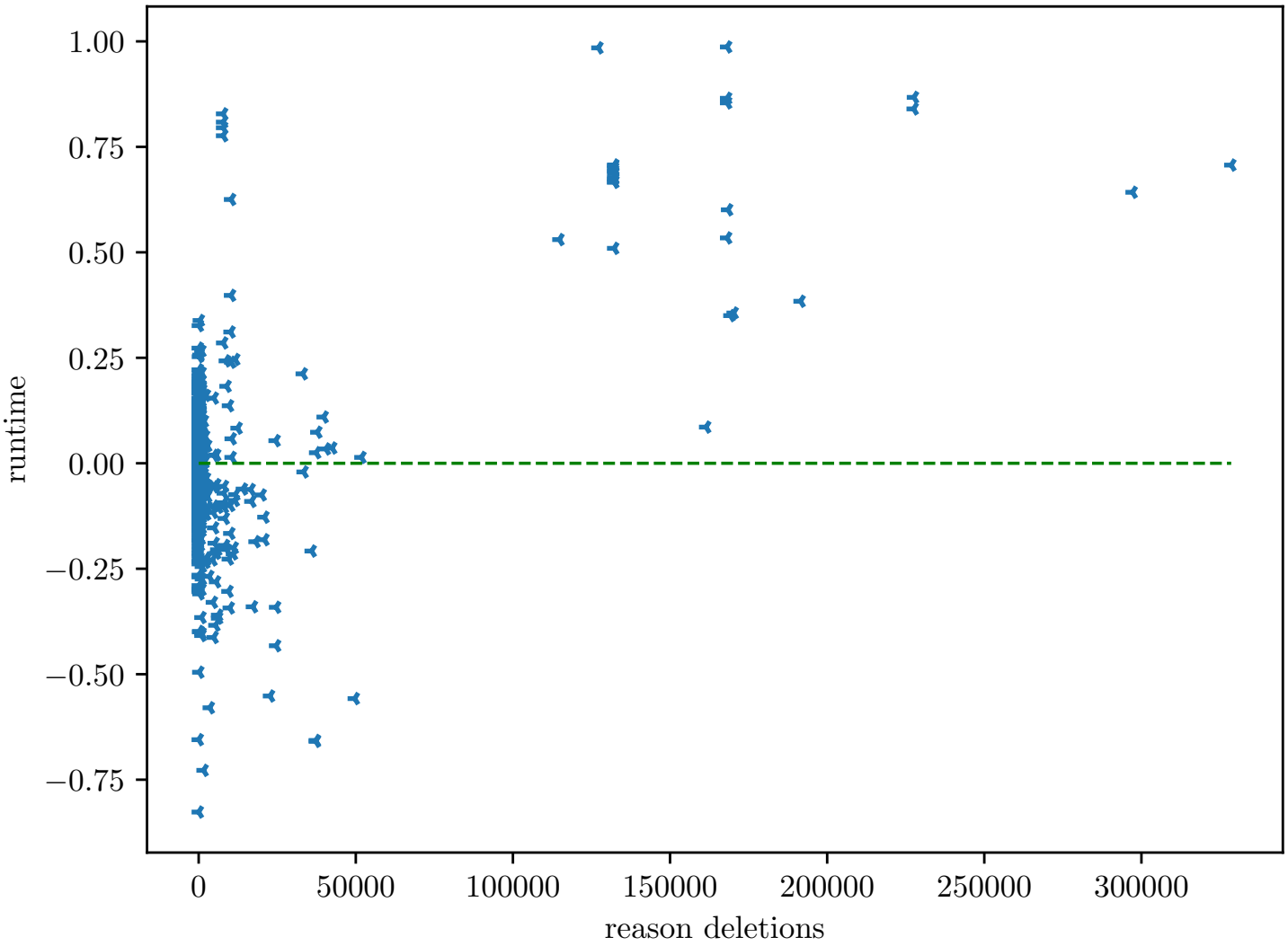


Answering the Research Question

Is it possible to check specified DRAT as efficiently as operational DRAT?

- On the average instance, the cost is the same
- Specified DRAT is usually more costly on proofs with many deletions

Overhead of Reason Deletions



Double Checking Incorrectness Results

- Many proofs are rejected by `rate` (*could that be a bug?*)
- SICK format specifies small, efficiently checkable counter-examples for clausal proofs
- Comprises the incorrect proof step & the partial assignment
- Checked with a simple independent tool (called `sick-check`)
- `sick-check` would be much more complex and less efficient with operational DRAT (essentially defeating its purpose)

Conclusion

- Specified DRAT is required for
  - verifying advanced inprocessing techniques
  - reasoning about a proof efficiently
- We pave the way for specified DRAT by
  - patching solvers to conform to it
  - providing an efficient checker
  - specifying SICK witnesses of proof incorrectness

Outlook

- Idea: operational DRAT, but fail on deletions of unique reason clauses
- DPR as more powerful proof format DRAT
  - `rate` already implements it
  - lack of large benchmarks
- LRAT as alternative to DRAT?
  - saves time & costs space
  - only one solver supports it