Short Proofs Without New Variables

Marijn J.H. Heule, Benjamin Kiesl, and Armin Biere

UT Austin, Vienna University of Technology, and JKU Linz

CADE-26 in Gothenburg, Sweden

August 8, 2017

Proofs of Unsatisfiability

Certifying Satisfiability and Unsatisfiability

- Certifying satisfiability of a formula is easy:

$$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$$

Certifying Satisfiability and Unsatisfiability

- Certifying satisfiability of a formula is easy:
  - Just consider a satisfying assignment: $x\bar{y}z$

$$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$$

  - We can easily check that the assignment is satisfying:
    Just check for every clause if it has a satisfied literal!

Certifying Satisfiability and Unsatisfiability

- Certifying satisfiability of a formula is easy:
  - Just consider a satisfying assignment: $x\bar{y}z$

$$(x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})$$

  - We can easily check that the assignment is satisfying:
    Just check for every clause if it has a satisfied literal!

- Certifying unsatisfiability is not so easy:
  - If a formula has n variables, there are 2^n possible assignments.
  - ➥ Checking whether every assignment falsifies the formula is costly.
  - More compact certificates of unsatisfiability are desirable.
    - ➥ Proofs

What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable...

What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable...
    ... but can be of exponential size with respect to a formula.

What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable...
    ... but can be of exponential size with respect to a formula.

- Example: Resolution proofs
  - A resolution proof is a sequence $C_1, \ldots, C_m$ of clauses.
  - Every clause is either contained in the formula or derived from two earlier clauses via the resolution rule:

$$\frac{C \vee x \qquad \bar{x} \vee D}{C \vee D}$$

  - C_m is the empty clause (containing no literals).
  - There exists a resolution proof for every unsatisfiable formula.

Resolution Proofs

- Example: $F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u)$
- Resolution proof:
  $(\bar{x} \lor \bar{y} \lor z), (\bar{z}), (\bar{x} \lor \bar{y}), (x \lor \bar{y}), (\bar{y}), (\bar{u} \lor y), (\bar{u}), (u), \bot$

Resolution Proofs

- Example: $F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{z}) \wedge (x \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (u)$
- Resolution proof:
  $(\bar{x} \vee \bar{y} \vee z), (\bar{z}), (\bar{x} \vee \bar{y}), (x \vee \bar{y}), (\bar{y}), (\bar{u} \vee y), (\bar{u}), (u), \perp$

$$\dfrac{\dfrac{\bar{u} \vee y \qquad \dfrac{\dfrac{\bar{x} \vee \bar{y} \vee z \qquad \bar{z}}{\bar{x} \vee \bar{y}} \qquad x \vee \bar{y}}{\bar{y}}}{\bar{u}} \qquad u}{\perp}$$

Resolution Proofs

- Example: $F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u)$
- Resolution proof:
  $(\bar{x} \lor \bar{y} \lor z), (\bar{z}), (\bar{x} \lor \bar{y}), (x \lor \bar{y}), (\bar{y}), (\bar{u} \lor y), (\bar{u}), (u), \bot$

$$\dfrac{\dfrac{\bar{u} \lor y \qquad \dfrac{\dfrac{\bar{x} \lor \bar{y} \lor z \qquad \bar{z}}{\bar{x} \lor \bar{y}} \qquad x \lor \bar{y}}{\bar{y}}}{\bar{u}} \qquad u}{\bot}$$

- Drawbacks of resolution:
  - For many seemingly simple formulas, there are only resolution proofs of exponential size.
  - State-of-the-art solving techniques are not succinctly expressible.

Interference-Based Proofs

Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is logically implied by the premises.

$$\frac{C \lor x \qquad \bar{x} \lor D}{C \lor D}\,(\text{res}) \qquad\qquad \frac{A \qquad A \rightarrow B}{B}\,(\text{mp})$$

Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is logically implied by the premises.

$$\frac{C \lor x \qquad \bar{x} \lor D}{C \lor D}\,(\text{res}) \qquad\qquad \frac{A \qquad A \rightarrow B}{B}\,(\text{mp})$$

➥ Inference rules reason about the presence of facts.

- If certain premises are present, infer the conclusion.

Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is logically implied by the premises.

$$\frac{C \lor x \qquad \bar{x} \lor D}{C \lor D}\,(\text{res}) \qquad\qquad \frac{A \qquad A \rightarrow B}{B}\,(\text{mp})$$

➥ Inference rules reason about the presence of facts.

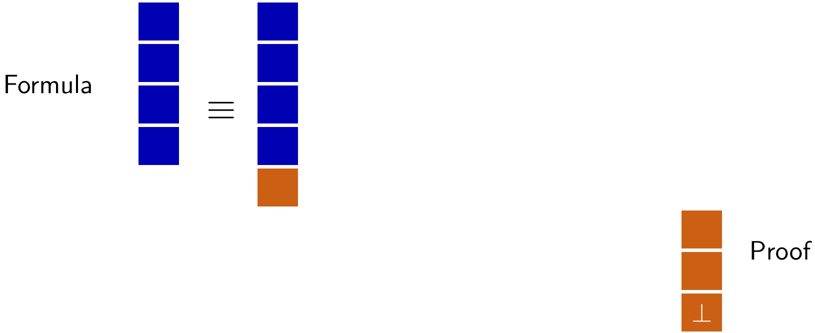
- If certain premises are present, infer the conclusion.

- Different approach: Allow not only implied conclusions.
  - Require only that the addition of facts preserves satisfiability.
  - Reason also about the absence of facts.

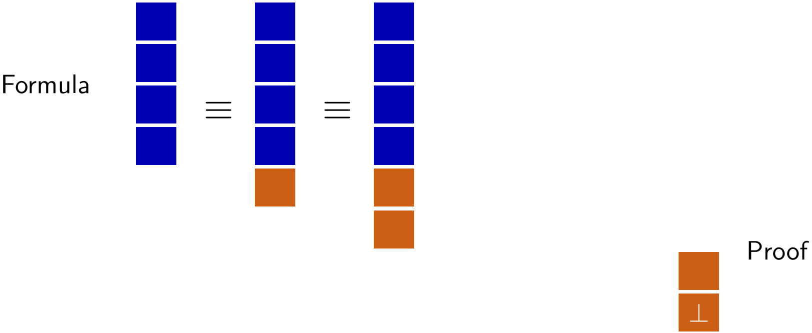
  ➥ This leads to interference-based proof systems.

Interference-Based Proofs

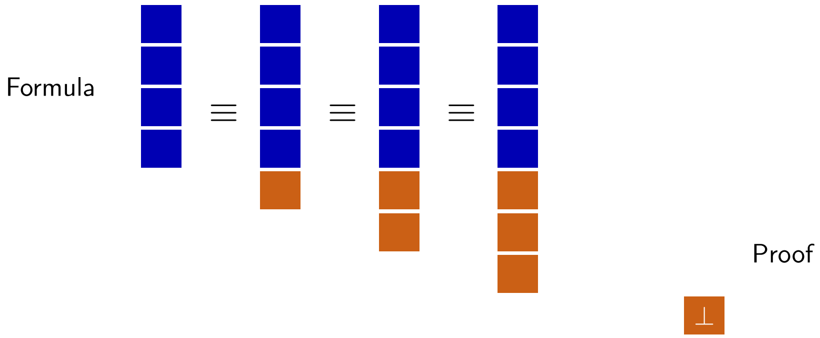
Interference-Based Proofs



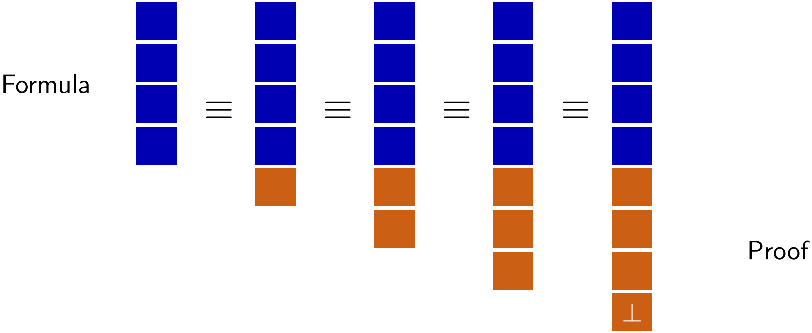
Interference-Based Proofs



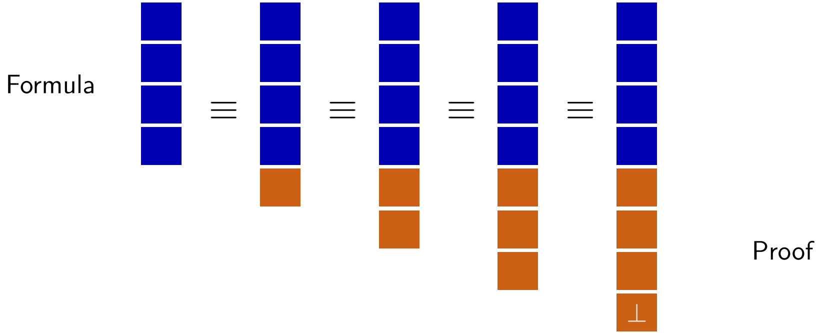
Interference-Based Proofs



Interference-Based Proofs

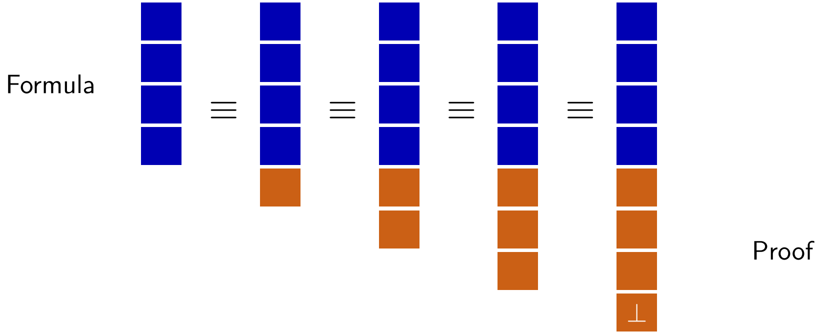


Interference-Based Proofs



- Checking whether additions preserve satisfiability should be efficient.
- Clauses whose addition preserves satisfiability are called redundant.

Interference-Based Proofs



- Checking whether additions preserve satisfiability should be efficient.
- Clauses whose addition preserves satisfiability are called redundant.

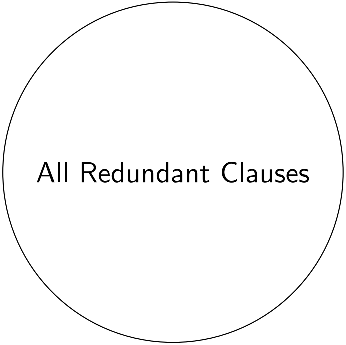
➥ Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

DRAT: An Interference-Based Proof System

- Popular example of an interference-based proof system: DRAT
- DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).
  - It can be efficiently checked if a clause is a RAT.
  - RATs are not necessarily implied by the formula.
  - But RATs are redundant: their addition preserves satisfiability.
  - A RAT check involves reasoning about the absence of facts.
    - A clause is a RAT w.r.t. a formula if the formula contains no clause such that ...
- Are there more general types of redundant clauses than RATs?

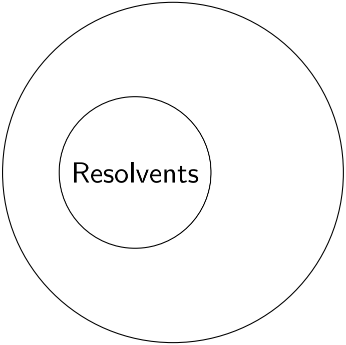
Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



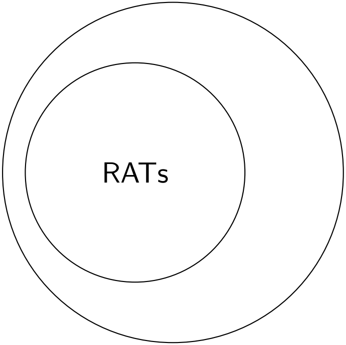
Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



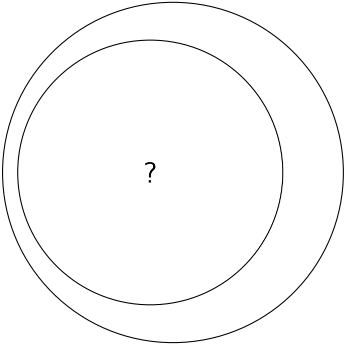
Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



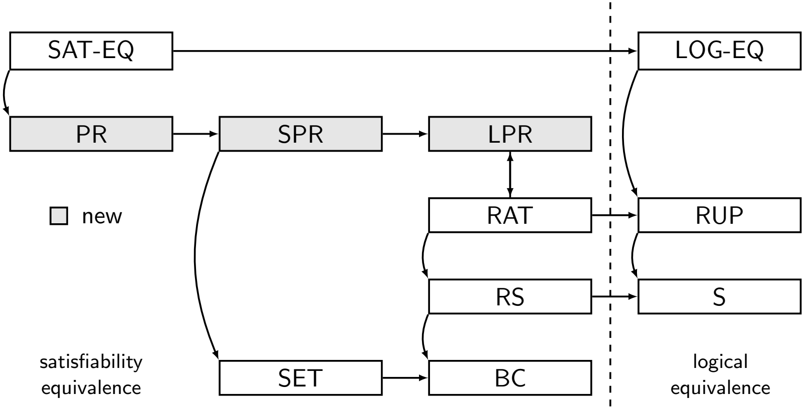
- Are stronger redundancy notions still efficiently checkable?

Propagation Redundancy

Main Contributions

- We introduced new clause-redundancy notions:
  - Propagation-redundant (PR) clauses
  - Set-propagation-redundant (SPR) clauses
  - Literal-propagation-redundant (LPR) clauses
- LPR clauses coincide with RAT.
- SPR clauses strictly generalize RATs.
- PR clauses strictly generalize SPR clauses.
- The redundancy notions provide the basis for new proof systems.

New Landscape of Redundancy Notions



Stronger Proof Systems: What Are They Good For?

- The new proof systems can give short proofs of formulas that are considered hard.

- We have short SPR and PR proofs for the well-known pigeon hole formulas (linear in the size of the input).
  - Pigeon hole formulas have only exponential-size resolution proofs.
  - If the addition of new variables via definitions is allowed, there are polynomial-size proofs.
    - So-called extended resolution proofs.

- Our proofs do not require new variables.

➥ Search space of possible clauses is finite.

➥ Makes search for such clauses easier.

Redundancy as an Implication

A formula G is at least as satisfiable as a formula F if $F \vDash G$.

Given a formula F and assignment α, we denote with $F|_\alpha$ the reduced formula after removing from F all clauses satisfied by α and all literals falsified by α.

Theorem

Let F be a formula, C a clause, and α the smallest assignment that falsifies C. Then, C is redundant w.r.t. F iff there exists an assignment ω such that 1) ω satisfies C; and 2) $F|_\alpha \vDash F|_\omega$.

This is the strongest notion of redundancy. However, it cannot be checked in polynomial time (assuming $\mathrm{P} \neq \mathrm{NP}$), unless bounded.

Checking Redundancy Using Unit Propagation

- Unit propagation (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula, C a clause, and α the smallest assignment that falsifies C. C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F|_\alpha$ results in a conflict.
- Implied by UP is used in SAT solvers to determine redundancy of learned clauses and therefore $\vdash_1$ is a natural restriction of $\models$.
- We bound $F|_\alpha \models F|_\omega$ by $F|_\alpha \vdash_1 F|_\omega$.
- Example: $F = (x \vee y \vee z) \wedge (\overline{x} \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{y} \vee z)$ and $G = (z)$. Observe that $F \models G$, but that $F \nvdash_1 G$.

Evaluation

Hand-crafted PR Proofs of Pigeon Hole Formulas

We manually constructed PR proofs of the famous pigeon hole formulas and the two-pigeons-per-hole family.

- The proofs consist only of binary and unit clauses.
- Only original variables appear in the proof.
- All proofs are linear in the size of the formula.

➥ Our proofs are smaller than Cook's extended resolution proofs.

- All resolution proofs of these formulas are exponential in size.

Pigeon Hole Formulas

Can $n{+}1$ pigeons be placed in n holes (at most one pigeon per hole)?

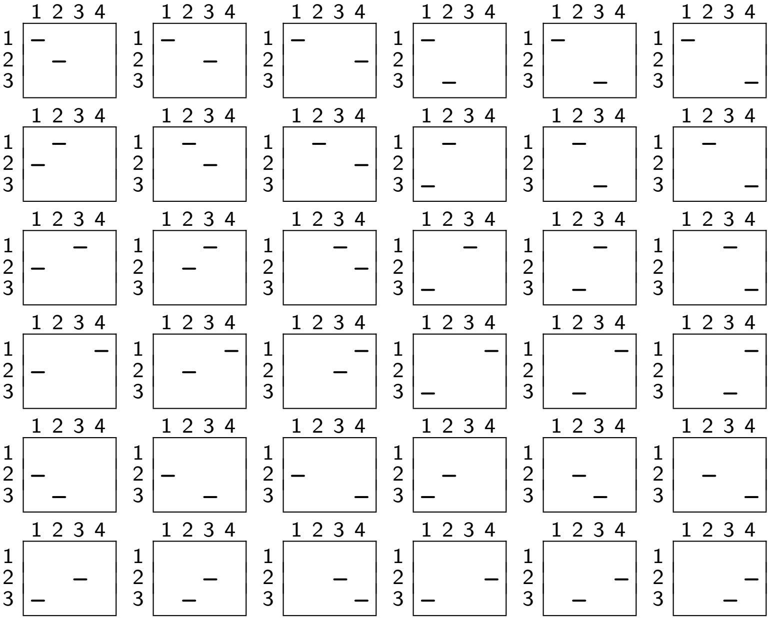
$$PHP_n := \bigwedge_{1 \le j \le n+1} (x_{1,j} \vee \cdots \vee x_{n,j}) \wedge \bigwedge_{1 \le i \le n} \; \bigwedge_{1 \le j < k \le n+1} (\overline{x}_{i,j} \vee \overline{x}_{i,k})$$

Or in array notation for PHP_3 (inspired by Haken):

$$\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & + & & & \\ 2 & + & & & \\ 3 & + & & & \\ \hline \end{array} \qquad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & + & & \\ 2 & & + & & \\ 3 & & + & & \\ \hline \end{array} \qquad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & + & \\ 2 & & & + & \\ 3 & & & + & \\ \hline \end{array} \qquad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & + \\ 2 & & & & + \\ 3 & & & & + \\ \hline \end{array}$$

$$\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & - & - & & \\ 2 & & & & \\ 3 & & & & \\ \hline \end{array} \quad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & - & & - & \\ 2 & & & & \\ 3 & & & & \\ \hline \end{array} \quad \cdots \quad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & - & - & \\ 3 & & & & \\ \hline \end{array} \quad \cdots \quad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & - & & - \\ \hline \end{array} \quad \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & & - & - \\ \hline \end{array}$$

All Binary PR Clauses for PHP_3



PR Clauses for Pigeon Hole Formulas

Array notation for PHP_3 (inspired by Haken):

$$
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & + & & & \\ 2 & + & & & \\ 3 & + & & & \\ \hline \end{array}
\quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & + & & \\ 2 & & + & & \\ 3 & & + & & \\ \hline \end{array}
\quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & + & \\ 2 & & & + & \\ 3 & & & + & \\ \hline \end{array}
\quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & + \\ 2 & & & & + \\ 3 & & & & + \\ \hline \end{array}
$$

$$
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & - & - & & \\ 2 & & & & \\ 3 & & & & \\ \hline \end{array}
\quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & - & & - & \\ 2 & & & & \\ 3 & & & & \\ \hline \end{array}
\quad \cdots \quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & - & - & \\ 3 & & & & \\ \hline \end{array}
\quad \cdots \quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & - & & - \\ \hline \end{array}
\quad
\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & & - & - \\ \hline \end{array}
$$

Key observation: each clause $\bar{x}_{i,j} \vee \bar{x}_{l,k}$ with $i \neq l$, $j \neq k$ is a PR clause.

$$
C : \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & - & & \\ 2 & & & & \\ 3 & & & & - \\ \hline \end{array}
\qquad
\alpha : \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & + & & \\ 2 & & & & \\ 3 & & & & + \\ \hline \end{array}
\qquad
\omega : \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & & - & & + \\ 2 & & & & \\ 3 & & + & & - \\ \hline \end{array}
$$

One can learn a unit clause after learning n such binary clauses.

One can reduce PHP_n to PHP_{n-1} by learning n such unit clauses.

Efficient PR Proof Checker

We implemented an efficient PR proof checker on top of the `DRAT-trim` checker (used to validate SAT competition results).

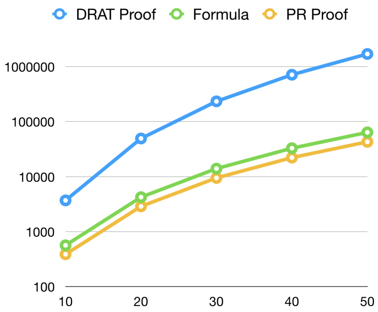
- Complexity is $\mathcal{O}(m^3)$ with m being the number of proof steps.
- However the worst-case is similar to DRAT proof checking...

➥ ..., and DRAT proof checking is in practice almost linear in the size of the formula and proof, by aggressively deleting clauses to limit the size of F.

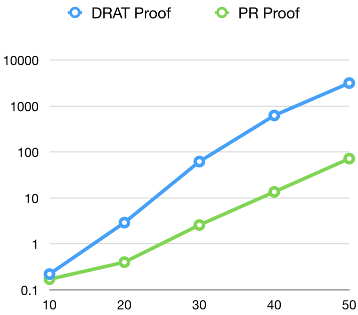
PRcheck (CNF formula F; PR proof $(C_1, \omega_1), \ldots, (C_m, \omega_m)$)

for $i \in \{i, \ldots, m\}$ **do**
  for $D \in F$ **do**
    if $D|\omega_i \neq \top$ **and** $(D|\alpha_i = \top$ **or** $D|\omega_i \subset D|\alpha_i)$ **then**
      if $F|\alpha_i \nvdash_1 D|\omega_i$ **then return** *failure*
  $F := F \cup \{C_i\}$
return *success*

Comparison of Proof Size and Validation Times



size in the number of clauses



validation time in seconds

Conclusions

Conclusions

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.
- Proof systems based on these redundancy notions are strong.
  - They allow for short proofs without new variables.

Conclusions

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.
- Proof systems based on these redundancy notions are strong.
  - They allow for short proofs without new variables.
- Proofs for the pigeon hole formulas are hand-crafted.
  ➥ Open problem: Automatically generate such short proofs.
  - A first approach "Satisfaction-Driven Clause Learning" under submission.

Short Proofs Without New Variables

Marijn J.H. Heule, Benjamin Kiesl, and Armin Biere

UT Austin, Vienna University of Technology, and JKU Linz

CADE-26 in Gothenburg, Sweden August 8, 2017