What a Difference a Variable Makes

Marijn J.H. Heule and Armin Biere

UT Austin and JKU Linz

TACAS 2018 April 18, 2018

Proofs of Unsatisfiability

Interference-Based Proofs

Conversion Algorithms

Evaluation

Conclusions

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: $x\bar{y}z$

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

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

- Certifying 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?

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

What Is a Proof in SAT?

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

What Is a Proof in SAT?

- A proof is a string that certifies the unsatisfiability of a formula (in general)
  - 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), denoted by $\bot$.
  - There exists a resolution proof for every unsatisfiable formula.

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:
  $(\bar{x} \vee \bar{y} \vee z), (\bar{z}), (\bar{x} \vee \bar{y}), (x \vee \bar{y}), (\bar{y}), (\bar{u} \vee y), (\bar{u}), (u), \bot$

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:
  $(\bar{x} \vee \bar{y} \vee z), (\bar{z}), (\bar{x} \vee \bar{y}), (x \vee \bar{y}), (\bar{y}), (\bar{u} \vee y), (\bar{u}), (u), \bot$

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

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:
  $(\bar{x} \vee \bar{y} \vee z), (\bar{z}), (\bar{x} \vee \bar{y}), (x \vee \bar{y}), (\bar{y}), (\bar{u} \vee y), (\bar{u}), (u), \bot$

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

- 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.

$$\frac{C \vee x \qquad \bar{x} \vee D}{C \vee D}\ (\text{res}) \qquad \frac{A \qquad A \to B}{B}\ (\text{mp})$$

Traditional Proofs vs. Interference-Based Proofs

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

$$\frac{C \vee x \qquad \bar{x} \vee D}{C \vee D}\ (\text{res}) \qquad \frac{A \qquad A \to B}{B}\ (\text{mp})$$

➥ 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.

$$\frac{C \vee x \qquad \bar{x} \vee D}{C \vee D}\ (\text{res}) \qquad \frac{A \qquad A \to B}{B}\ (\text{mp})$$

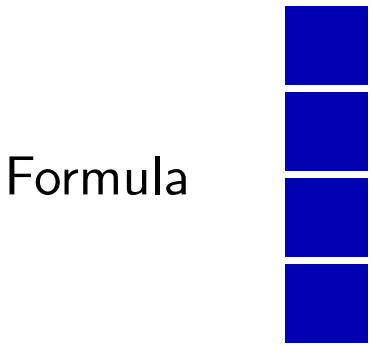
↪ Inference rules reason about the presence of facts.

- 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.

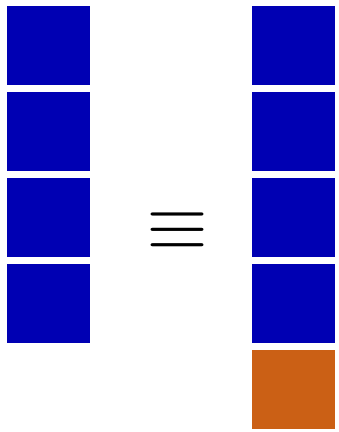
  ↪ This leads to interference-based proof systems.

Interference-Based Proofs



Interference-Based Proofs

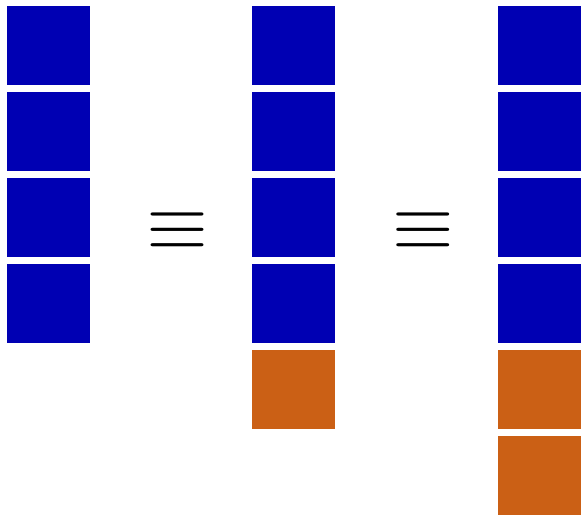
Formula



Proof

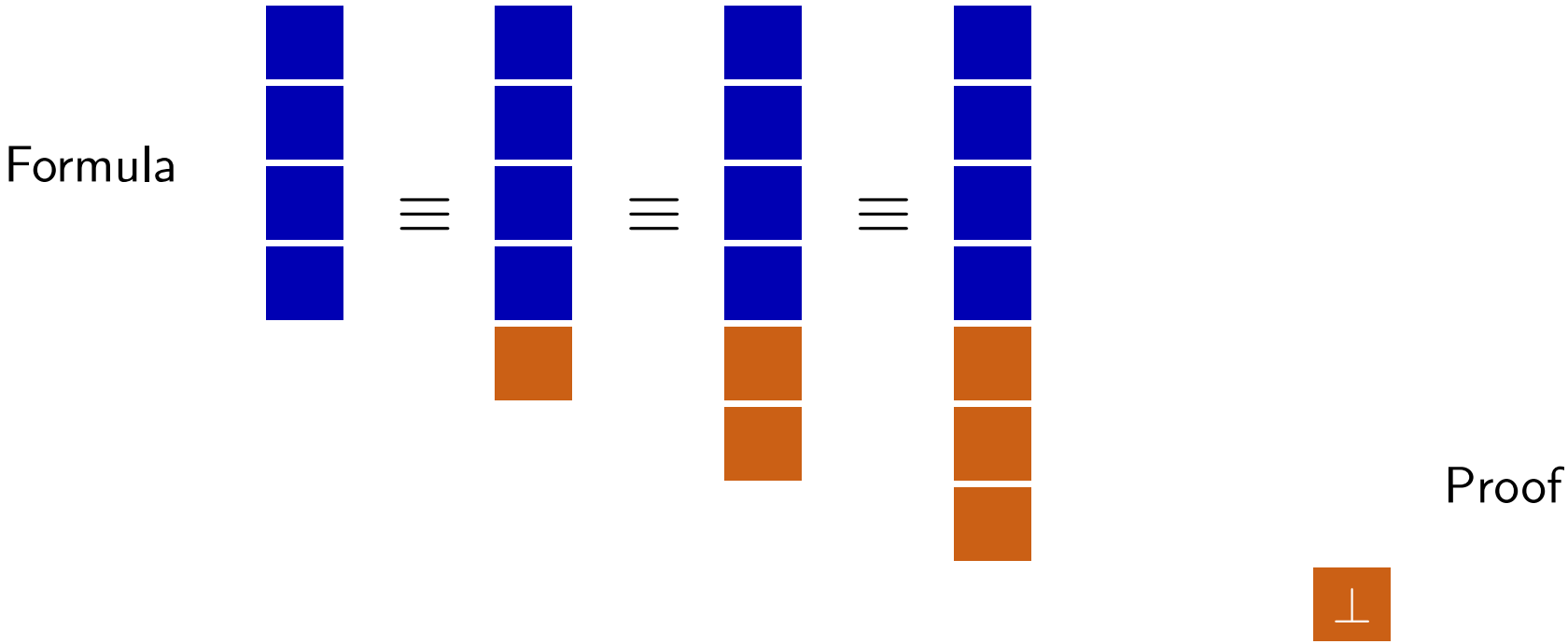
Interference-Based Proofs

Formula

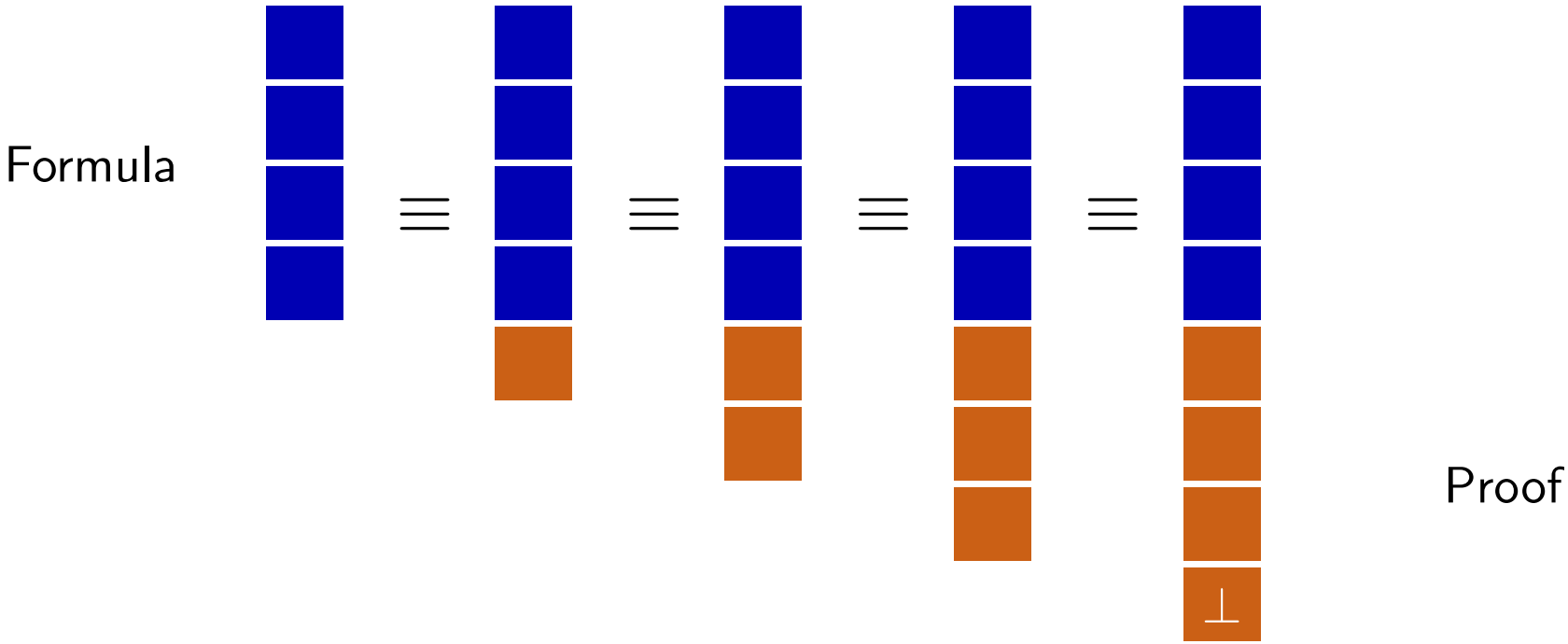


Proof

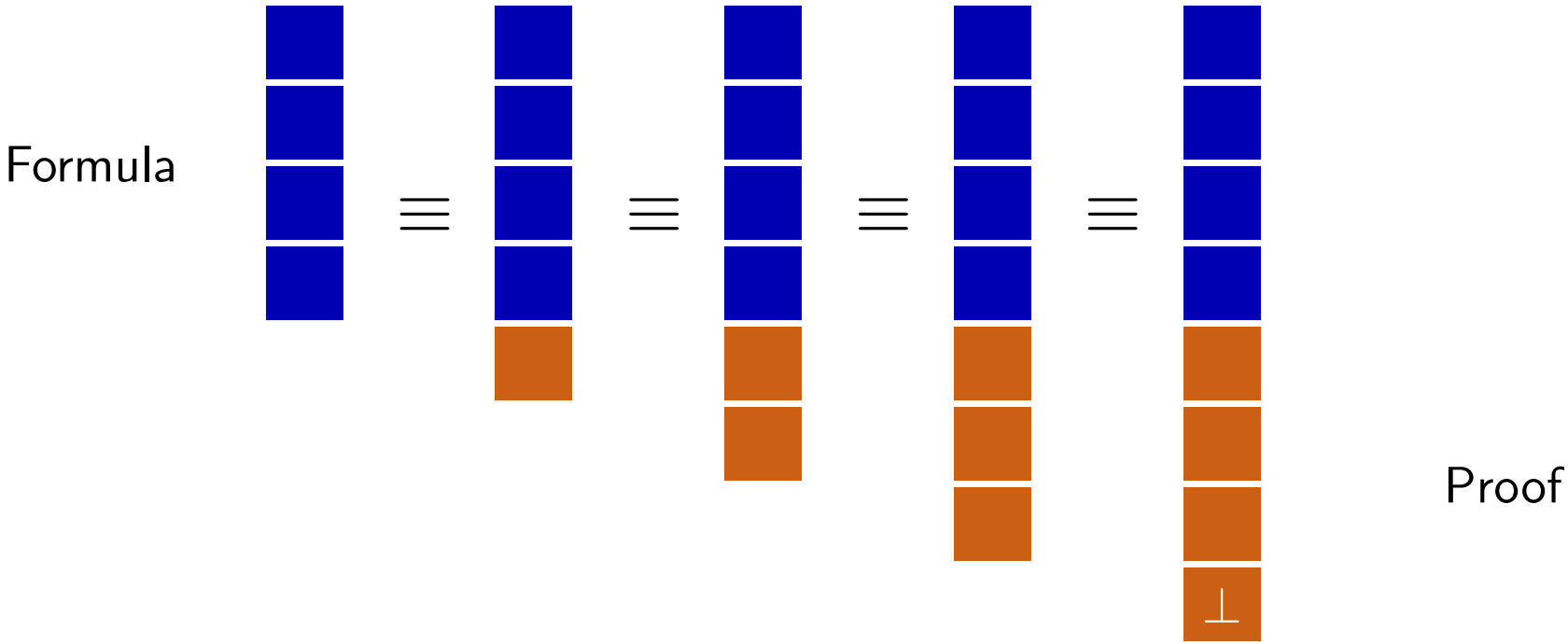
Interference-Based Proofs



Interference-Based Proofs

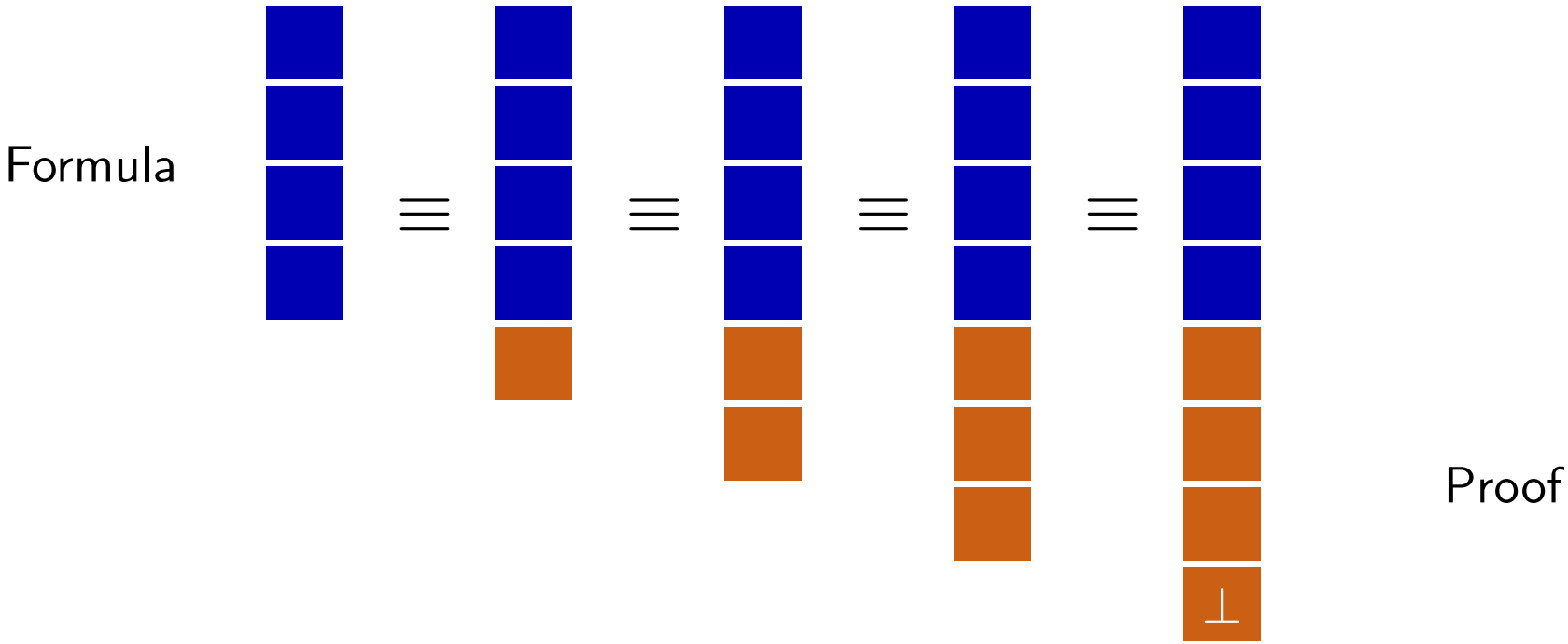


Interference-Based Proofs



- 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?
  - Yes! PR (short for Propagation Redundant) clauses.

Bocked Clauses

clauses with x	clauses with $\bar{x}$	other clauses (without x nor $\bar{x}$)
$\vdots$	$(\bar{x} \vee \bar{a})$	$\vdots$
$\boxed{(a \vee b \vee c \vee x)}$	$(\bar{x} \vee \bar{b})$	$\vdots$
$\vdots$	$(\bar{x} \vee \bar{c})$	$\vdots$

all resolvents of $\boxed{(a \vee b \vee c \vee x)}$ on x are tautological

Resolution Asymmetric Tautological (RAT) Clauses

clauses with x	clauses with $\bar{x}$	other clauses (without x nor $\bar{x}$)
$\vdots$	$(\bar{x} \vee \bar{a})$	$(a \vee y)$
$\boxed{(a \vee b \vee x)}$	$(\bar{x} \vee \bar{b})$	$(b \vee z)$
$\vdots$	$(\bar{x} \vee \bar{c})$	$(\bar{c} \vee \bar{y} \vee \bar{x})$

all resolvents of $\boxed{(a \vee b \vee x)}$ on x are unit implied (written $\vdash_1$)

in particular the resolvent $(a \vee b \vee \bar{c})$

Propagation Redundant (PR) Clauses

Definition

Let F be a formula, C a non-empty clause, and α the smallest assignment that falsifies C. Then, C is propagation redundant (PR) with respect to F if there exists an assignment ω which satisfies C, such that $F|_\alpha \vdash_1 F|_\omega$.

The clause C "prunes" all assignments that extend α.

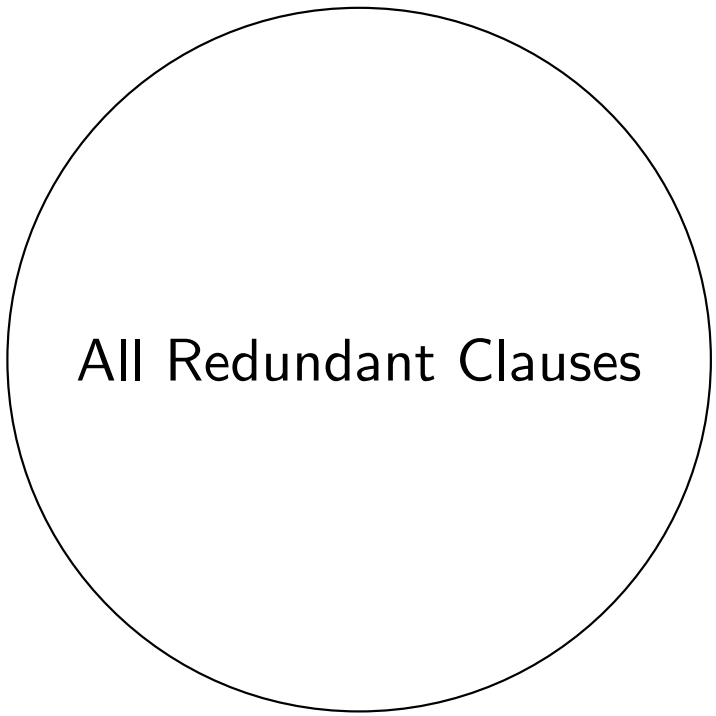
The witness ω provides an alternative at least as satisfiable (partial) assignment as α.

Example

Let $F = (x \vee y) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee z)$, $C = (x)$, and let $\omega = x\,z$ be an assignment. Then, $\alpha = \overline{x}$ is the smallest assignment that falsifies C. Now, consider $F|_\alpha = (y)$ and $F|_\omega = (y)$. Clearly, unit propagation on $F|_\alpha \wedge (\overline{y})$ derives a conflict. Thus, $F|_\alpha \vdash_1 F|_\omega$ and C is propagation redundant w.r.t. F. Notice that C is not RAT w.r.t F as $(y) = F|_\alpha \nvdash_1 F|_{\alpha_x} = (y)(z)$.

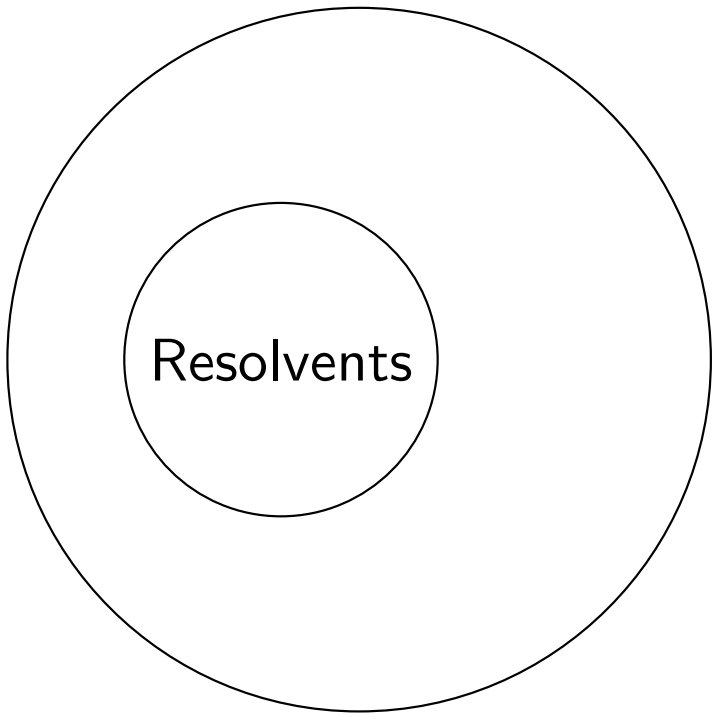
Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



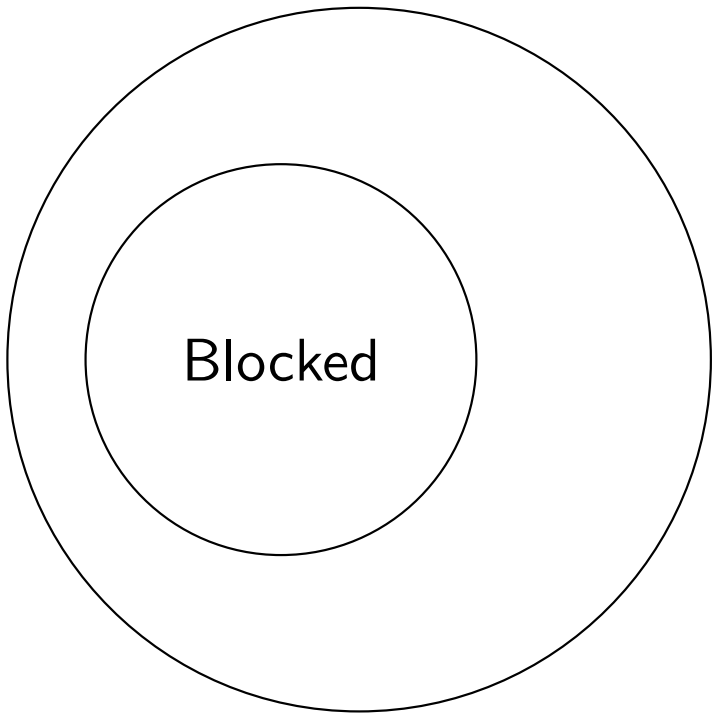
Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



Redundant Clauses

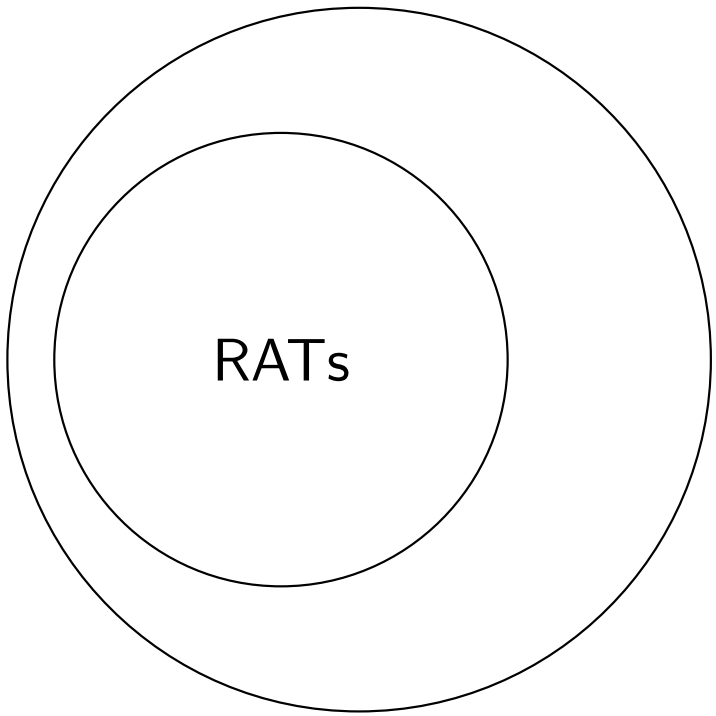
- Strong proof systems allow addition of many redundant clauses.



- Adding Blocked Clauses allows exponentially smaller proofs

Redundant Clauses

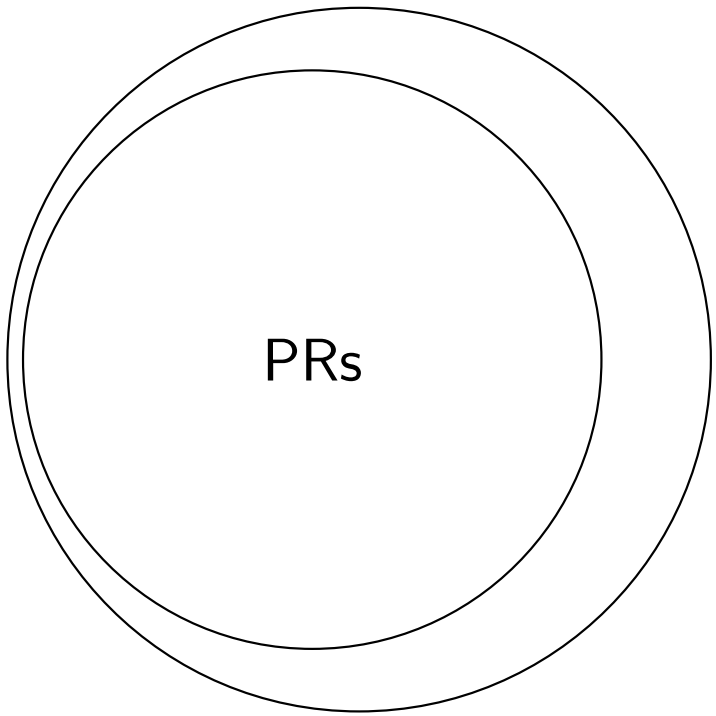
- Strong proof systems allow addition of many redundant clauses.



- Adding Blocked Clauses allows exponentially smaller proofs
- Same applies to adding RAT Clauses

Redundant Clauses

- Strong proof systems allow addition of many redundant clauses.



- Adding Blocked Clauses allows exponentially smaller proofs
- Same applies to adding RAT Clauses
- Can PRs result in exponentially smaller proofs (w.r.t. RATs)?

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 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.

Conversion Algorithms

The Plain Algorithm

We present a 5-phase algorithm to convert proofs in PR into DRAT

- The algorithm is quite technical and omitted from this talk;
- Please check the paper for the details.

The algorithm introduces only a single new Boolean variable

The worst-case blow-up in size is quadratic

This algorithm shows that the proofs systems are equally strong

The tool `PR2DRAT` implements the algorithm

Optimizations

Focus on refutations

- The algorithm weakens many clauses;
- To prove unsatisfiability these steps are trivial.

Minimize the witness

- The size of the blow-up depends on the witness of a PR clause;
- Hence, minimizing the witness reduces the size of DRAT proof.

From quadratic to linear blow-up in practice

- Under some conditions, the conversion is linear;
- These conditions appeared to occur quite frequently in practice.

Alternative Conversions

Limiting the number of expensive steps

- The plain algorithm introduces several expensive (RAT) checks;
- All but one can be replaced by multiple basic checks.

Converting DPR proofs into DRAT proofs

- The algorithm focusses on conversion of proofs of unsatisfiability;
- A modified algorithm converts all derivations can be converted into DRAT.

Converting PR refutations into RAT refutations

- Clause deletion in DRAT is required to use only one new variable;
- Conversion into RAT requires introducing many new variables.

Evaluation

Experimental Setup

Benchmark suite of famous hard problems for resolution:

- Pigeon hole formulas (holeXX);
- Two-pigeons-per-hole formulas (tphXX);
- Tseitin formulas by Urquhart (Urq-S5-bX).

Experiment I — Size comparison of DRAT proofs from PR proofs:

- Used the smallest known DRAT proofs in the literature;
- Compared the plain and optimized conversions.

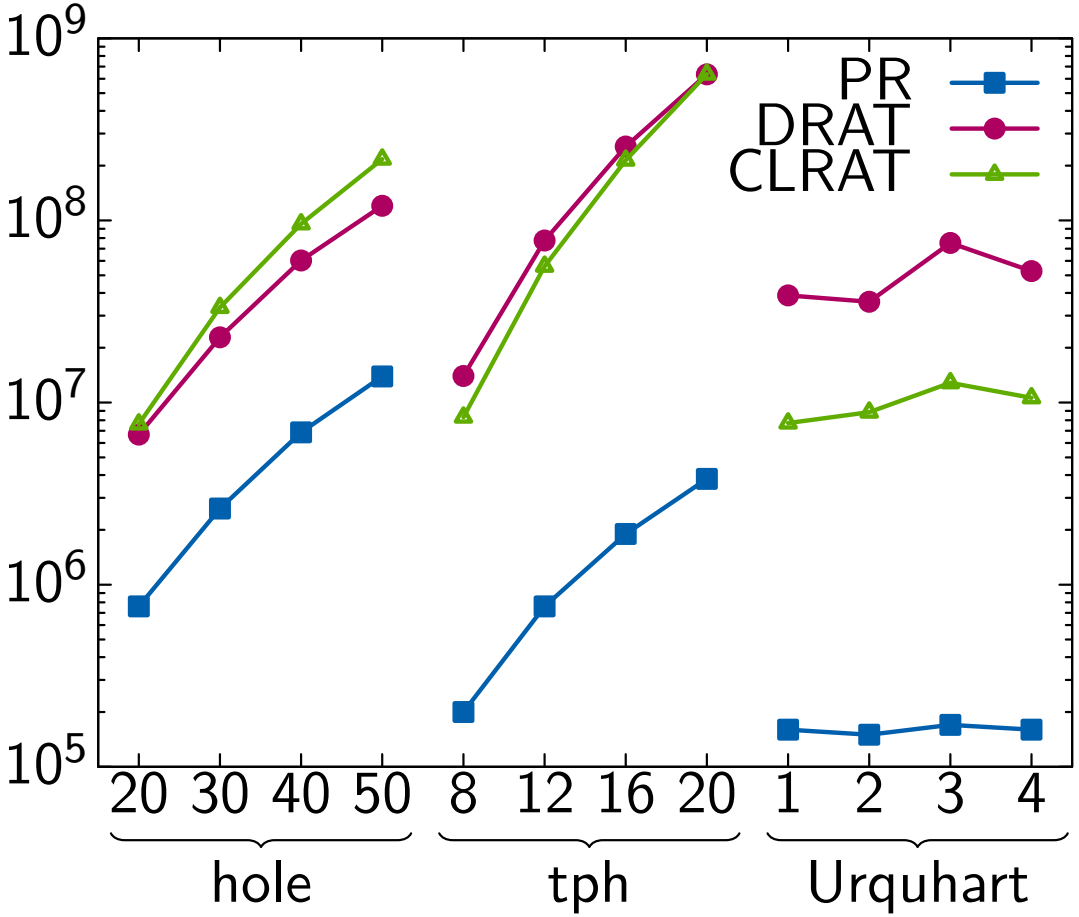
Experiment II — Validate existing PR proofs using the conversion:

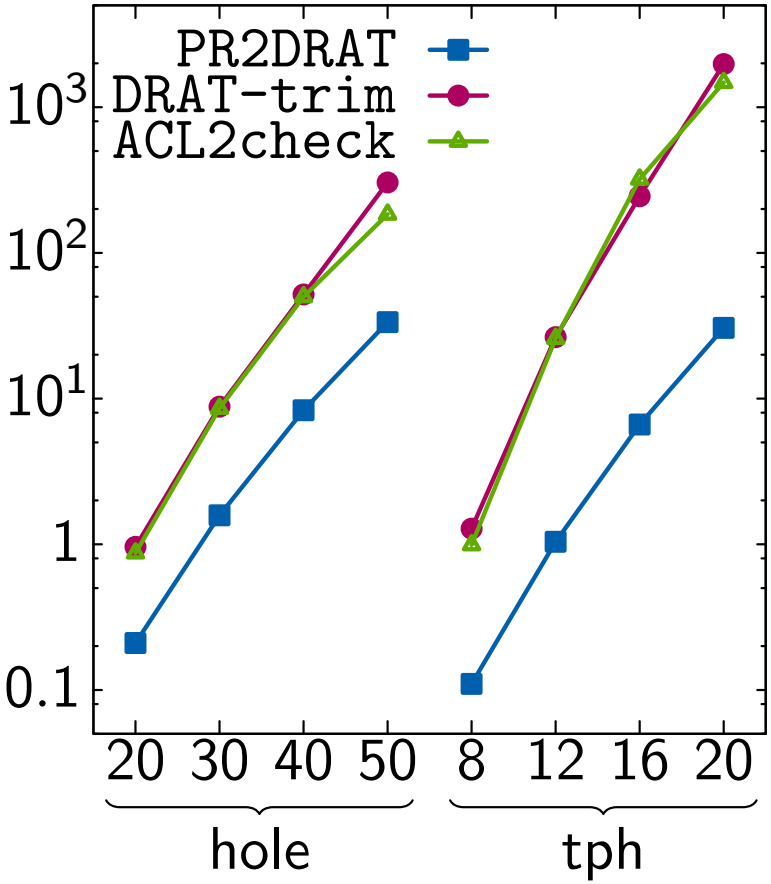
- The optimized DRAT proofs were converted to CLRAT proofs;
- Which are validated using formally-verified checker `ACL2check`.

Comparing Size (Clause Additions) of DRAT Proofs

	input		PR	DRAT proofs (#add)		
formula	#var	#cls	#add	existing	plain	optimized
hole20	420	4,221	2,870	49,410	94,901	**26,547**
hole30	930	13,981	9,455	234,195	422,101	**89,827**
hole40	1,640	32,841	22,140	715,030	1,241,126	**213,107**
hole50	2,550	63,801	42,925	1,708,915	2,893,476	**416,387**
tph8	136	5,457	1,156	253,958	86,216	**25,204**
tph12	300	27,625	3,950	1,966,472	612,108	**127,296**
tph16	528	87,329	9,416	—	2,490,672	**401,004**
tph20	820	213,241	18,450	—	7,440,692	**976,376**
Urq-s5-b1	106	714	620	—	30,235	**28,189**
Urq-s5-b2	107	742	606	—	34,535	**32,574**
Urq-s5-b3	121	1,116	692	—	44,117	**41,230**
Urq-s5-b4	114	888	636	—	40,598	**37,978**

Certification of PR proofs





Proof sizes in bytes (left) and verification time (right).

Verification time of Urquhart instances less than a second.

Conclusions

Conclusions

PR proofs can be converted efficiently into DRAT proofs:

- Requires the introduction of only a single new variable;
- The size blowup is $\mathcal{O}(N^2)$, but can be linear in practice;
- This shows that strength of these proof systems is equivalent.

We implemented a tool that performs the conversion:

- The tool `PR2DRAT` is open source;
- Supports various conversion variants and optimizations;
- Produces DRAT proofs that are smaller than existing ones;
- Validation of PR proofs with the final step being formally verified.

What a Difference a Variable Makes

Marijn J.H. Heule and Armin Biere

UT Austin and JKU Linz

TACAS 2018 April 18, 2018