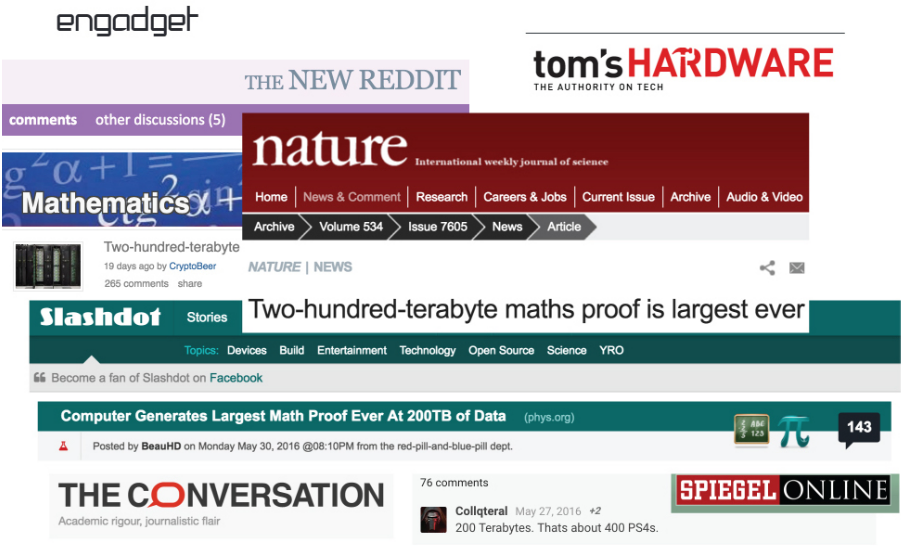
Logic and Mechanized Reasoning
Computer-Generated Propositional Proofs

Marijn J.H. Heule

Carnegie Mellon University

"The Largest Math Proof Ever"



Motivation

SAT solvers can efficiently solve many application problems

However, for various small problems the runtime is exponential

- Pigeon-hole formulas, Tseitin formulas, mutilated chessboards
- ... these formulas require exponential resolution proofs

Motivation

SAT solvers can efficiently solve many application problems

However, for various small problems the runtime is exponential

- Pigeon-hole formulas, Tseitin formulas, mutilated chessboards
- ... these formulas require exponential resolution proofs

Several solvers go beyond resolution to solve them efficiently

- In which proof systems can we express the reasoning?
- How effective is "Without Loss of Satisfaction" reasoning?
- What are the limitations of this kind of reasoning?

Motivation

SAT solvers can efficiently solve many application problems

However, for various small problems the runtime is exponential

- Pigeon-hole formulas, Tseitin formulas, mutilated chessboards
- ... these formulas require exponential resolution proofs

Several solvers go beyond resolution to solve them efficiently

- In which proof systems can we express the reasoning?
- How effective is "Without Loss of Satisfaction" reasoning?
- What are the limitations of this kind of reasoning?

Research motivated by advancing the techniques and verification

Proofs of Unsatisfiability

Beyond Resolution

Strong Extension-Free Proof Systems

Beyond Symmetry Breaking

Proofs of Unsatisfiability

Beyond Resolution

Strong Extension-Free Proof Systems

Beyond Symmetry Breaking

Certifying Satisfiability and Unsatisfiability

- Certifying satisfiability of a formula is easy:

$$(p \lor q) \land (\neg p \lor \neg q) \land (\neg q \lor \neg r)$$

Certifying Satisfiability and Unsatisfiability

- Certifying satisfiability of a formula is easy:
  - Just consider a satisfying assignment: $p \, \neg q \, r$

    $$(p \lor q) \land (\neg p \lor \neg q) \land (\neg q \lor \neg r)$$

  - 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: $p \neg q \, r$

    $$(p \lor q) \land (\neg p \lor \neg q) \land (\neg q \lor \neg r)$$

  - 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 (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 (polynomial-time) checkable...
    ... but can be of exponential size with respect to a formula.
    The size of the proof usually linear in the runtime of the solver.

What Is a Proof in SAT?

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

- 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:

$$\frac{C \lor p \qquad \neg p \lor D}{C \lor D}$$

  - C_m is the empty clause (containing no literals), denoted by $\bot$.
  - There exists a resolution proof for every unsatisfiable formula.

Resolution Proofs

Example

$\Gamma := (\neg p \lor \neg q \lor r) \land (\neg r) \land (p \lor \neg q) \land (\neg s \lor q) \land (s)$

Resolution proof: $(\neg p \lor \neg q \lor r)$, $(\neg r)$, $(\neg p \lor \neg q)$, $(p \lor \neg q)$, $(\neg q)$, $(\neg s \lor q)$, $(\neg s)$, (s), $\bot$

$$\dfrac{\dfrac{\neg s \lor q \qquad \dfrac{\dfrac{\neg p \lor \neg q \lor r \qquad \neg r}{\neg p \lor \neg q} \qquad p \lor \neg q}{\neg q}}{\neg s} \qquad s}{\bot}$$

Resolution Proofs

Example

$\Gamma := (\neg p \lor \neg q \lor r) \land (\neg r) \land (p \lor \neg q) \land (\neg s \lor q) \land (s)$

Resolution proof: $(\neg p \lor \neg q \lor r)$, $(\neg r)$, $(\neg p \lor \neg q)$, $(p \lor \neg q)$, $(\neg q)$, $(\neg s \lor q)$, $(\neg s)$, (s), $\bot$

$$\dfrac{\dfrac{\neg s \lor q \qquad \dfrac{\dfrac{\neg p \lor \neg q \lor r \qquad \neg r}{\neg p \lor \neg q} \qquad p \lor \neg q}{\neg q}}{\neg s} \qquad s}{\bot}$$

Drawbacks of resolution:

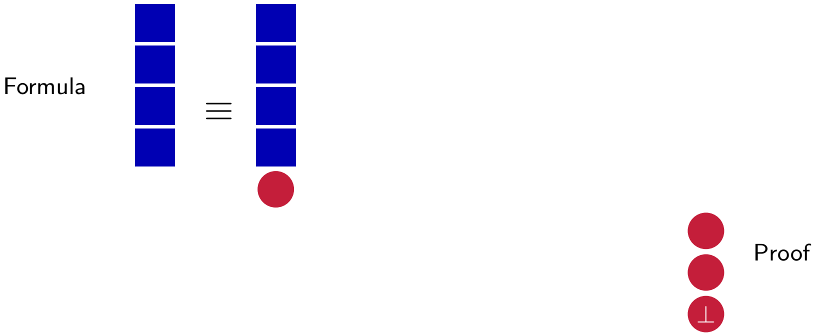
- For many seemingly simple formulas, there are only resolution proofs of exponential size.
- State-of-the-art techniques are not succinctly expressible.

Clausal Proofs

Reduce the size of the proof by only storing added clauses

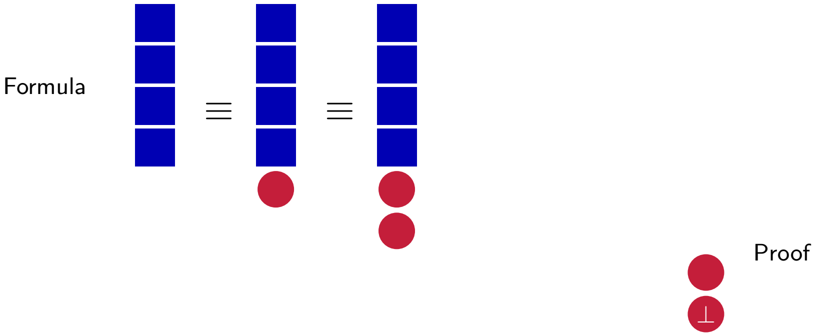
Clausal Proofs

Reduce the size of the proof by only storing added clauses



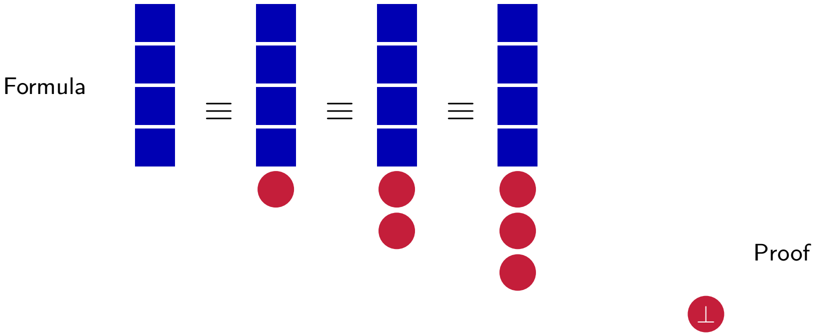
Clausal Proofs

Reduce the size of the proof by only storing added clauses



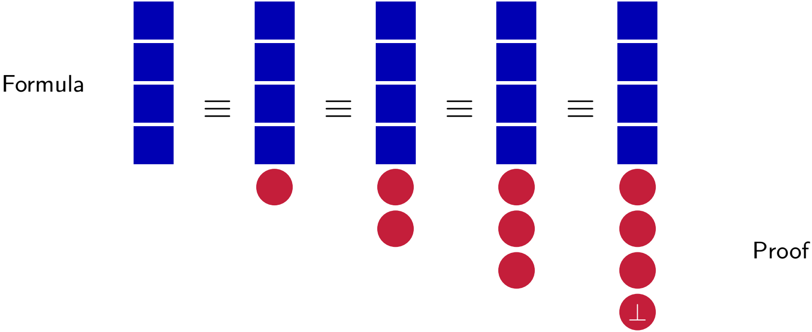
Clausal Proofs

Reduce the size of the proof by only storing added clauses



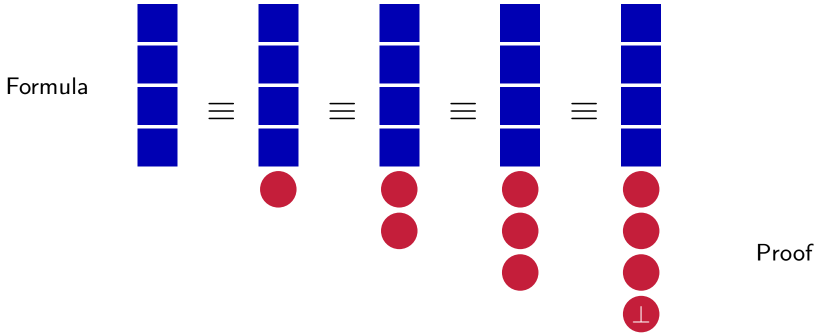
Clausal Proofs

Reduce the size of the proof by only storing added clauses



Clausal Proofs

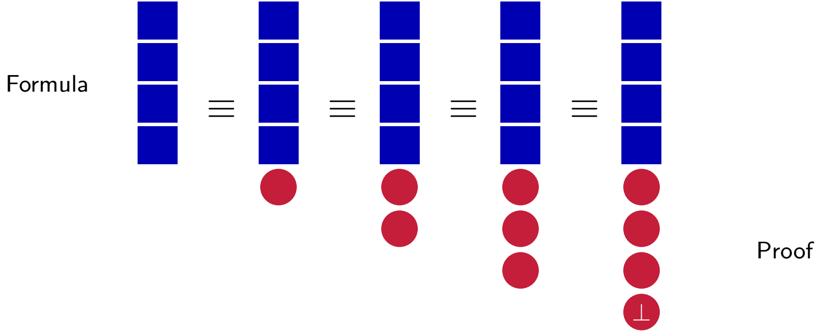
Reduce the size of the proof by only storing added clauses



- Clauses whose addition preserves satisfiability are redundant.
- Checking redundancy should be efficient.

Clausal Proofs

Reduce the size of the proof by only storing added clauses



- Clauses whose addition preserves satisfiability are redundant.
- Checking redundancy should be efficient.

➥ Idea: Only add clauses that fulfill an efficiently checkable redundancy criterion.

Proofs of Unsatisfiability

Beyond Resolution

Strong Extension-Free Proof Systems

Beyond Symmetry Breaking

Traditional Proofs vs. Interference-Based Proofs

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

$$\frac{C \lor p \qquad \neg p \lor D}{C \lor D}\ (\text{RES}) \qquad \frac{A \qquad A \rightarrow B}{B}\ (\text{MP})$$

Traditional Proofs vs. Interference-Based Proofs

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

$$\frac{C \lor p \qquad \neg p \lor D}{C \lor D}\,\text{(RES)} \qquad\qquad \frac{A \qquad A \to B}{B}\,\text{(MP)}$$

➥ 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.

$$\frac{C \lor p \qquad \neg p \lor D}{C \lor D}\ \text{(RES)} \qquad \frac{A \qquad A \rightarrow B}{B}\ \text{(MP)}$$

➥ 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.

Early work on reasoning beyond resolution

The early SAT decision procedures used the Pure Literal rule [Davis and Putnam 1960; Davis, Logemann and Loveland 1962]:

$$\frac{\neg p \notin \Gamma}{(p)}\ \text{(pure)}$$

Early work on reasoning beyond resolution

The early SAT decision procedures used the Pure Literal rule [Davis and Putnam 1960; Davis, Logemann and Loveland 1962]:

$$\frac{\neg p \notin \Gamma}{(p)} \text{ (pure)}$$

Extended Resolution (ER) [Tseitin 1966]

- Combines resolution with the Extension rule:

$$\frac{p \notin \Gamma \qquad \neg p \notin \Gamma}{(p \vee \neg a \vee \neg b) \wedge (\neg p \vee a) \wedge (\neg p \vee b)} \text{ (ER)}$$

- Equivalently, adds the definition $p := \mathrm{AND}(a, b)$
- Can be considered the first interference-based proof system
- Is very powerful: Only modest lower bounds results are known

Short Proofs of Pigeon Hole Formulas [Cook 1967]

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

$$PHP_n := \bigwedge_{1 \le i \le n+1} (p_{1,i} \vee \cdots \vee p_{n,i}) \wedge \bigwedge_{1 \le h \le n,\ 1 \le i < j \le n+1} (\neg p_{h,i} \vee \neg p_{h,j})$$

Resolution proofs of PHP_n must be exponential [Haken 1985]

Cook constructed polynomial-sized ER proofs of PHP_n

Short Proofs of Pigeon Hole Formulas [Cook 1967]

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

$$PHP_n := \bigwedge_{1 \leq i \leq n+1} (p_{1,i} \vee \cdots \vee p_{n,i}) \wedge \bigwedge_{1 \leq h \leq n,\, 1 \leq i < j \leq n+1} (\neg p_{h,i} \vee \neg p_{h,j})$$

Resolution proofs of PHP_n must be exponential [Haken 1985]

Cook constructed polynomial-sized ER proofs of PHP_n

However, these proofs require introducing new variables:

- Hard to find such proofs automatically
- Existing ER approaches produce exponentially large proofs
- How to get rid of this hurdle? First approach: blocked clauses...

Blocked Clauses [Kullmann 1999]

Definition (Blocked Clause)

A clause $(C \vee p)$ is a blocked on p w.r.t. a CNF formula Γ if for every clause $(D \vee \neg p) \in \Gamma$, resolvent $C \vee D$ is a tautology.

Blocked Clauses [Kullmann 1999]

Definition (Blocked Clause)

A clause $(C \vee p)$ is a blocked on p w.r.t. a CNF formula Γ if for every clause $(D \vee \neg p) \in \Gamma$, resolvent $C \vee D$ is a tautology.

Example

Consider the formula $(p \vee q) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee r)$.
Second clause is blocked by both p *and* $\neg r$.
Third clause is blocked by p

Theorem

Adding or removing a blocked clause preserves (un)satisfiability.

Blocked Clauses [Kullmann 1999]

Definition (Blocked Clause)

A clause $(C \vee p)$ is a blocked on p w.r.t. a CNF formula Γ if for every clause $(D \vee \neg p) \in \Gamma$, resolvent $C \vee D$ is a tautology.

Example

Consider the formula $(p \vee q) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee r)$.
Second clause is blocked by both p *and* $\neg r$.
Third clause is blocked by p

Theorem

Adding or removing a blocked clause preserves (un)satisfiability.

Proof sketch: Given a formula Γ and a clause $C \vee p$ that is blocked on p w.r.t. Γ. Let assignment τ satisfy Γ, but falsify $C \vee p$. Note that all clauses $D \vee \neg p$ are doubly satisfied by τ. Flipping p to true in τ satisfies both Γ and $C \vee p$.

Blocked Clause Addition and Blocked Clause Elimination

The Blocked Clause proof system (BC) combines the resolution rule with the addition of blocked clauses.

- BC generalizes ER [Kullmann 1999]
- Recall

$$\frac{p \notin \Gamma \qquad \neg p \notin \Gamma}{(p \vee \neg a \vee \neg b) \wedge (\neg p \vee a) \wedge (\neg p \vee b)} \text{(ER)}$$

- The ER clauses are blocked on the literals p and $\neg p$ w.r.t. Γ

Blocked Clause Addition and Blocked Clause Elimination

The Blocked Clause proof system (BC) combines the resolution rule with the addition of blocked clauses.

- BC generalizes ER [Kullmann 1999]
- Recall

$$\frac{p \notin \Gamma \qquad \neg p \notin \Gamma}{(p \vee \neg a \vee \neg b) \wedge (\neg p \vee a) \wedge (\neg p \vee b)} \text{ (ER)}$$

- The ER clauses are blocked on the literals p and $\neg p$ w.r.t. Γ

Blocked clause elimination used in preprocessing and inprocessing

- Simulates many circuit optimization techniques
- Removes redundant Pythagorean Triples

Blocked Clause Elimination (BCE)

Definition (BCE)

While there is a blocked clause C in a CNF Γ, remove C from Γ.

Example

Consider $(p \vee q) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee r)$.
After removing either $(p \vee \neg q \vee \neg r)$ *or* $(\neg p \vee r)$*, the clause* $(p \vee q)$ *becomes blocked (no clause with either* $\neg q$ *or* $\neg p$*).*

An extreme case in which BCE removes all clauses!

Blocked Clause Elimination (BCE)

Definition (BCE)

While there is a blocked clause C in a CNF Γ, remove C from Γ.

Example

Consider $(p \vee q) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee r)$.
After removing either $(p \vee \neg q \vee \neg r)$ *or* $(\neg p \vee r)$, *the clause* $(p \vee q)$ *becomes blocked (no clause with either* $\neg q$ *or* $\neg p$*).*

An extreme case in which BCE removes all clauses!

Proposition

BCE is confluent, i.e., has a unique fixpoint

- Blocked clauses stay blocked w.r.t. removal

BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

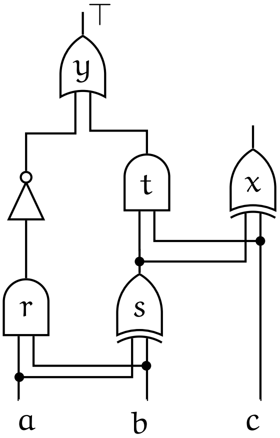
$(\neg y \lor t \lor \neg r)$
$(y \lor \neg t)$
$(y \lor r)$

$(\neg x \lor s \lor c)$
$(\neg x \lor \neg s \lor \neg c)$
$(x \lor s \lor \neg c)$
$(x \lor \neg s \lor c)$

$(t \lor \neg s \lor \neg c)$
$(\neg t \lor s)$
$(\neg t \lor c)$

$(r \lor \neg a \lor \neg b)$
$(\neg r \lor a)$
$(\neg r \lor b)$

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
$(s \lor a \lor \neg b)$
$(s \lor \neg a \lor b)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

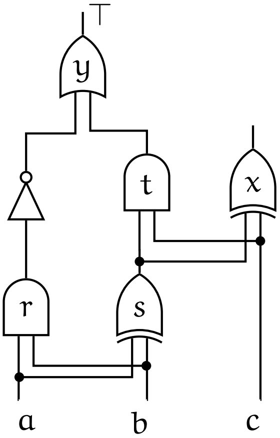
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

$(\neg x \lor s \lor c)$
$(\neg x \lor \neg s \lor \neg c)$
$(x \lor s \lor \neg c)$
$(x \lor \neg s \lor c)$

$(t \lor \neg s \lor \neg c)$
$(\neg t \lor s)$
$(\neg t \lor c)$

$(r \lor \neg a \lor \neg b)$
$(\neg r \lor a)$
$(\neg r \lor b)$

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
$(s \lor a \lor \neg b)$
$(s \lor \neg a \lor b)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

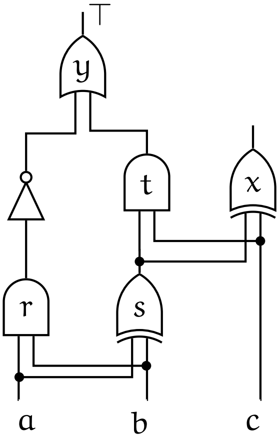
$(\neg y \vee t \vee \neg r)$
~~$(y \vee \neg t)$~~
~~$(y \vee r)$~~

~~$(\neg x \vee s \vee c)$~~
~~$(\neg x \vee \neg s \vee \neg c)$~~
~~$(x \vee s \vee \neg c)$~~
~~$(x \vee \neg s \vee c)$~~

$(t \vee \neg s \vee \neg c)$
$(\neg t \vee s)$
$(\neg t \vee c)$

$(r \vee \neg a \vee \neg b)$
$(\neg r \vee a)$
$(\neg r \vee b)$

$(\neg s \vee a \vee b)$
$(\neg s \vee \neg a \vee \neg b)$
$(s \vee a \vee \neg b)$
$(s \vee \neg a \vee b)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

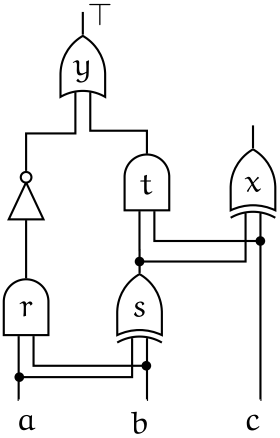
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

~~$(\neg x \lor s \lor c)$~~
~~$(\neg x \lor \neg s \lor \neg c)$~~
~~$(x \lor s \lor \neg c)$~~
~~$(x \lor \neg s \lor c)$~~

~~$(t \lor \neg s \lor \neg c)$~~
$(\neg t \lor s)$
$(\neg t \lor c)$

$(r \lor \neg a \lor \neg b)$
$(\neg r \lor a)$
$(\neg r \lor b)$

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
$(s \lor a \lor \neg b)$
$(s \lor \neg a \lor b)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

- BCE converts the Tseitin encoding to Plaisted Greenbaum
- BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

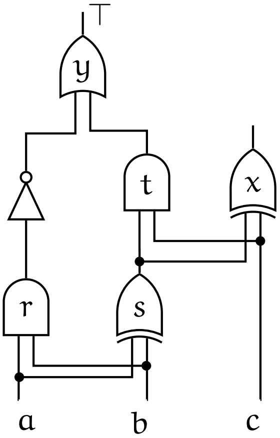
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

~~$(\neg x \lor s \lor c)$~~
~~$(\neg x \lor \neg s \lor \neg c)$~~
~~$(x \lor s \lor \neg c)$~~
~~$(x \lor \neg s \lor c)$~~

~~$(t \lor \neg s \lor \neg c)$~~
$(\neg t \lor s)$
$(\neg t \lor c)$

$(r \lor \neg a \lor \neg b)$
~~$(\neg r \lor a)$~~
~~$(\neg r \lor b)$~~

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
$(s \lor a \lor \neg b)$
$(s \lor \neg a \lor b)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

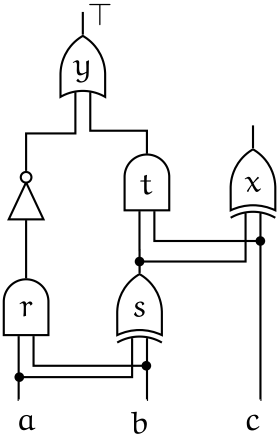
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

~~$(\neg x \lor s \lor c)$~~
~~$(\neg x \lor \neg s \lor \neg c)$~~
~~$(x \lor s \lor \neg c)$~~
~~$(x \lor \neg s \lor c)$~~

~~$(t \lor \neg s \lor \neg c)$~~
$(\neg t \lor s)$
$(\neg t \lor c)$

$(r \lor \neg a \lor \neg b)$
~~$(\neg r \lor a)$~~
~~$(\neg r \lor b)$~~

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
~~$(s \lor a \lor \neg b)$~~
~~$(s \lor \neg a \lor b)$~~



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

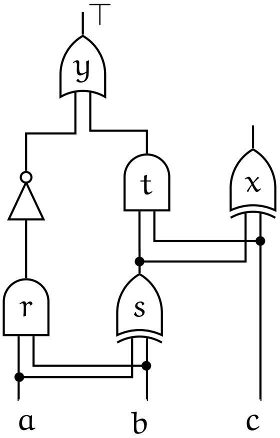
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

~~$(\neg x \lor s \lor c)$~~
~~$(\neg x \lor \neg s \lor \neg c)$~~
~~$(x \lor s \lor \neg c)$~~
~~$(x \lor \neg s \lor c)$~~

~~$(t \lor \neg s \lor \neg c)$~~
$(\neg t \lor s)$
$(\neg t \lor c)$

~~$(r \lor \neg a \lor \neg b)$~~
~~$(\neg r \lor a)$~~
~~$(\neg r \lor b)$~~

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
~~$(s \lor a \lor \neg b)$~~
~~$(s \lor \neg a \lor b)$~~



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

(y)

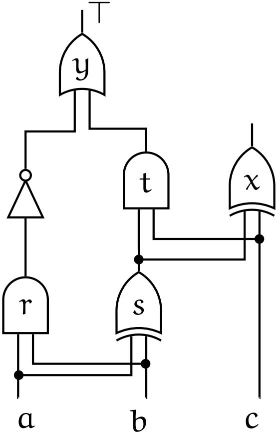
~~$(\neg y \vee t \vee \neg r)$~~
~~$(y \vee \neg t)$~~
~~$(y \vee r)$~~

~~$(\neg x \vee s \vee c)$~~
~~$(\neg x \vee \neg s \vee \neg c)$~~
~~$(x \vee s \vee \neg c)$~~
~~$(x \vee \neg s \vee c)$~~

~~$(t \vee \neg s \vee \neg c)$~~
$(\neg t \vee s)$
$(\neg t \vee c)$

~~$(r \vee \neg a \vee \neg b)$~~
~~$(\neg r \vee a)$~~
~~$(\neg r \vee b)$~~

$(\neg s \vee a \vee b)$
$(\neg s \vee \neg a \vee \neg b)$
~~$(s \vee a \vee \neg b)$~~
~~$(s \vee \neg a \vee b)$~~



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

~~(y)~~

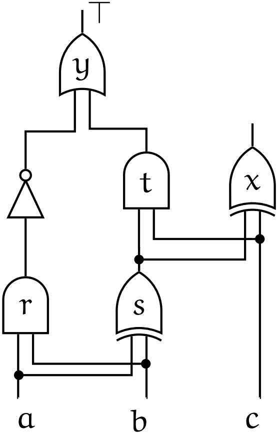
~~$(\neg y \lor t \lor \neg r)$~~
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

~~$(\neg x \lor s \lor c)$~~
~~$(\neg x \lor \neg s \lor \neg c)$~~
~~$(x \lor s \lor \neg c)$~~
~~$(x \lor \neg s \lor c)$~~

~~$(t \lor \neg s \lor \neg c)$~~
$(\neg t \lor s)$
$(\neg t \lor c)$

~~$(r \lor \neg a \lor \neg b)$~~
~~$(\neg r \lor a)$~~
~~$(\neg r \lor b)$~~

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
~~$(s \lor a \lor \neg b)$~~
~~$(s \lor \neg a \lor b)$~~



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

~~(y)~~

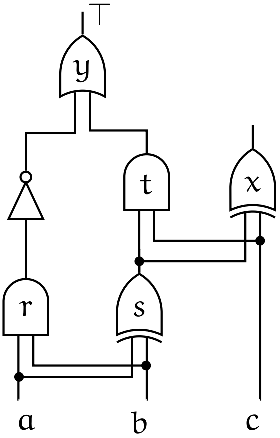
~~$(\neg y \lor t \lor \neg r)$~~
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

~~$(\neg x \lor s \lor c)$~~
~~$(\neg x \lor \neg s \lor \neg c)$~~
~~$(x \lor s \lor \neg c)$~~
~~$(x \lor \neg s \lor c)$~~

~~$(t \lor \neg s \lor \neg c)$~~
~~$(\neg t \lor s)$~~
~~$(\neg t \lor c)$~~

~~$(r \lor \neg a \lor \neg b)$~~
~~$(\neg r \lor a)$~~
~~$(\neg r \lor b)$~~

$(\neg s \lor a \lor b)$
$(\neg s \lor \neg a \lor \neg b)$
~~$(s \lor a \lor \neg b)$~~
~~$(s \lor \neg a \lor b)$~~



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseitin encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence, etc.

Example of circuit simplification by BCE on Tseitin encoding

~~(y)~~

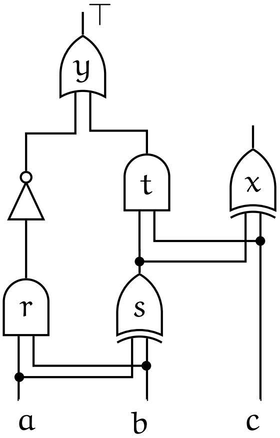
~~$(\neg y \vee t \vee \neg r)$~~
~~$(y \vee \neg t)$~~
~~$(y \vee r)$~~

~~$(\neg x \vee s \vee c)$~~
~~$(\neg x \vee \neg s \vee \neg c)$~~
~~$(x \vee s \vee \neg c)$~~
~~$(x \vee \neg s \vee c)$~~

~~$(t \vee \neg s \vee \neg c)$~~
~~$(\neg t \vee s)$~~
~~$(\neg t \vee c)$~~

~~$(r \vee \neg a \vee \neg b)$~~
~~$(\neg r \vee a)$~~
~~$(\neg r \vee b)$~~

~~$(\neg s \vee a \vee b)$~~
~~$(\neg s \vee \neg a \vee \neg b)$~~
~~$(s \vee a \vee \neg b)$~~
~~$(s \vee \neg a \vee b)$~~



Solution Reconstruction

Input:

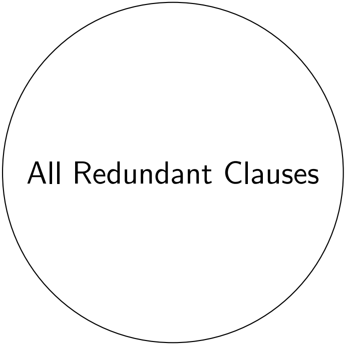
- stack S of eliminated blocked clauses
- formula Γ (without the blocked clauses)
- assignment τ that satisfies Γ

Output: an assignment that satisfies $\Gamma \wedge S$

while S.size () **do**
    $\langle C, l \rangle :=$ S.pop ()
    if τ falsifies C **then** $\tau := \tau \cup \{l = \top\}$
end while
return τ

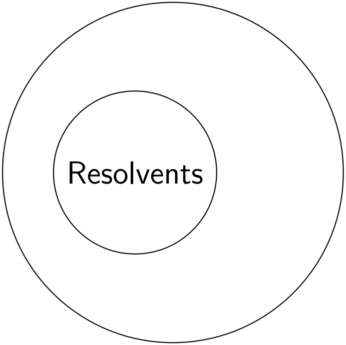
Redundant Clauses

- Strong proof systems allow adding many redundant clauses.



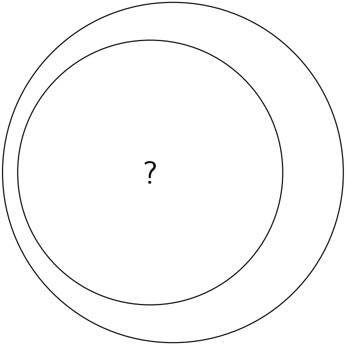
Redundant Clauses

- Strong proof systems allow adding many redundant clauses.



Redundant Clauses

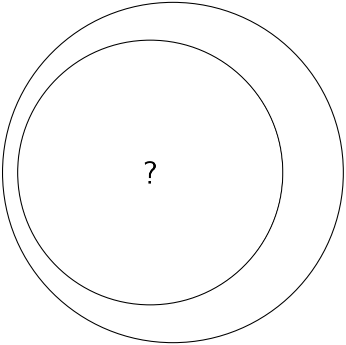
- Strong proof systems allow adding many redundant clauses.



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

Redundant Clauses

- Strong proof systems allow adding many redundant clauses.



- The new proof systems can give short proofs of formulas that are considered hard.
- Are stronger derivability notions still efficiently checkable?

Redundant clauses: Intuition

A proof is a sequence of redundant clauses ending with $\bot$

- (Informal): A clause C is redundant with respect to a formula Γ if adding C to Γ preserves satisfiability
- Redundancy should be checkable in polynomial time

Redundant clauses: Intuition

A proof is a sequence of redundant clauses ending with $\bot$

- (Informal): A clause C is redundant with respect to a formula Γ if adding C to Γ preserves satisfiability
- Redundancy should be checkable in polynomial time

Example $\Gamma = (\neg p \lor q) \land (p \lor q)$ $C = (p \lor \neg q)$

Redundant clauses: Intuition

A proof is a sequence of redundant clauses ending with $\bot$

- (Informal): A clause C is redundant with respect to a formula Γ if adding C to Γ preserves satisfiability
- Redundancy should be checkable in polynomial time

Example $\quad \Gamma = (\neg p \lor q) \land (p \lor q) \quad\quad C = (p \lor \neg q)$

When is a clause C redundant for a formula Γ (again informal)?

- Can assignments that satisfy Γ, but falsify C be repaired?
- Let τ be a (partial) assignment such that every assignment σ that satisfies Γ, but falsifies C will be turned into an assignment $\sigma \circ \tau$ that satisfies $\Gamma \land C$
- For the example, a valid repair is $\tau := \{p = \top\}$

Redundant Clauses: One Proof Rule

(Informal): A clause C is redundant w.r.t. a formula Γ if adding C to Γ preserves satisfiability

Example $\quad \Gamma = (\neg p \vee q) \wedge (p \vee q) \quad\quad C = (p \vee \neg q)$

Applying an assignment τ to a formula Γ removes satisfied clauses and falsified literals and is denoted by $[\![\Gamma]\!]_\tau$

Definition (Clause Redundancy)

A clause C is redundant w.r.t. formula Γ if there exists an assignment τ such that

$$\Gamma \wedge \neg C \models [\![\Gamma \wedge C]\!]_\tau$$

Note: this trivially holds if τ satisfies $\Gamma \wedge C$

Redundant Clauses: One Proof Rule

(Informal): A clause C is redundant w.r.t. a formula Γ if adding C to Γ preserves satisfiability

Example $\Gamma = (\neg p \vee q) \wedge (p \vee q)$ $C = (p \vee \neg q)$

Applying an assignment τ to a formula Γ removes satisfied clauses and falsified literals and is denoted by $[\![\Gamma]\!]_\tau$

Definition (Clause Redundancy)

A clause C is redundant w.r.t. formula Γ if there exists an assignment τ such that

$$\Gamma \wedge \neg C \models [\![\Gamma \wedge C]\!]_\tau$$

Note: this trivially holds if τ satisfies $\Gamma \wedge C$

Example

$$(\neg p \vee q) \wedge (p \vee q) \wedge \neg p \wedge q \models [\![(\neg p \vee q) \wedge (p \vee q) \wedge (p \vee \neg q)]\!]_p$$
$$\models q$$

Different Proof Systems Based on Restrictions on τ

To ensure polynomial time computation, we restrict entailment ($\models$) by unit propagation ($\vdash_1$).

A clause C is redundant with respect to a formula Γ if there exists an assignment τ such that

$$\Gamma \wedge \neg C \vdash_1 [\![\Gamma \wedge C]\!]_\tau$$

Proof systems differ based on the restrictions to τ:

- Reverse unit propagation (RUP): τ is empty
- Resolution asymmetric tautology (RAT): $|\tau| \leq 1$
- Propagation redundancy (PR): τ assigns any var to $\bot$ or $\top$
- Substitution redundancy (SR): τ can substitute variables

Different Proof Systems Based on Restrictions on τ

To ensure polynomial time computation, we restrict entailment ($\models$) by unit propagation ($\vdash_1$).

A clause C is redundant with respect to a formula Γ if there exists an assignment τ such that

$$\Gamma \wedge \neg C \vdash_1 [\![\Gamma \wedge C]\!]_\tau$$

Proof systems differ based on the restrictions to τ:

- Reverse unit propagation (RUP): τ is empty
- Resolution asymmetric tautology (RAT): $|\tau| \leq 1$
- Propagation redundancy (PR): τ assigns any var to $\perp$ or $\top$
- Substitution redundancy (SR): τ can substitute variables

focus on variants without new variables

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.

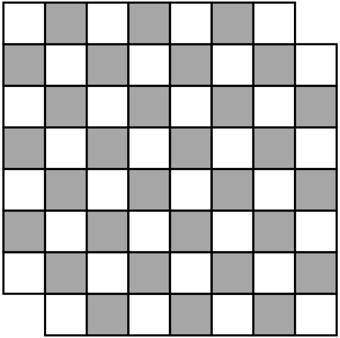
- The proofs consist only of binary and unit clauses.
- All proofs are linear in the size of the formula.
- Only original variables appear in the proof.

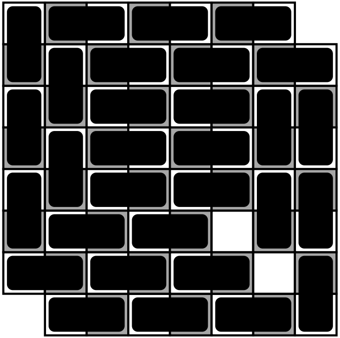
➥ The PR proofs are smaller than Cook's ER proofs.

- All resolution proofs must be exponential in size.
- Similar proofs can also be computed automatically.

Mutilated Chessboards: “A Tough Nut to Crack” [McCarthy]

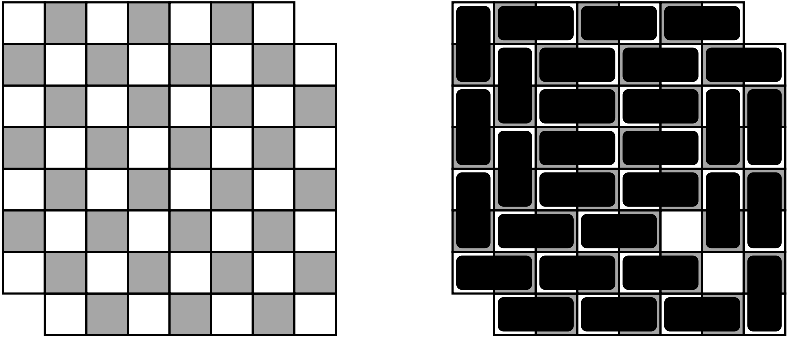
Can a chessboard be fully covered with dominos after removing two diagonally opposite corner squares?





Mutilated Chessboards: “A Tough Nut to Crack” [McCarthy]

Can a chessboard be fully covered with dominos after removing two diagonally opposite corner squares?

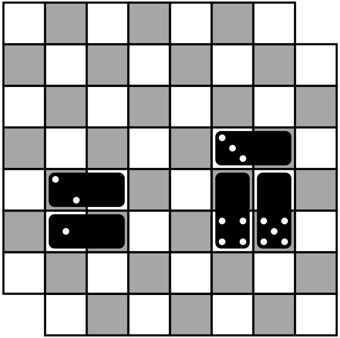


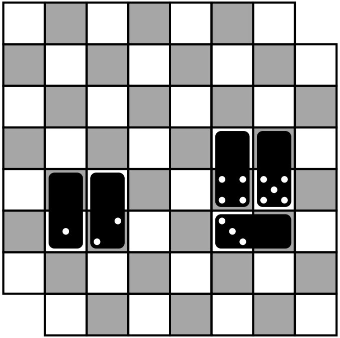
Easy to refute based on the following two observations:

- There are more white squares than black squares; and
- A domino covers exactly one white and one black square.

Mutilated Chessboards: A Computer-Generated Proof

Modern SAT solvers produce proofs that can be very different compared to human-made proofs for the same problem.





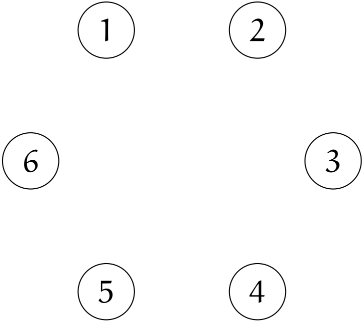
The two patterns can be automatically detected and blocked

- This reduces the number of explored states exponentially
- The PR proofs are linear in the formula size

Ramsey Numbers

Ramsey Number $R(k)$: What is the smallest n such that any graph with n vertices has either a clique or a co-clique of size k?

$$\begin{aligned} R(3) &= 6 \\ R(4) &= 18 \\ 43 \leq\ R(5) &\leq 48 \end{aligned}$$



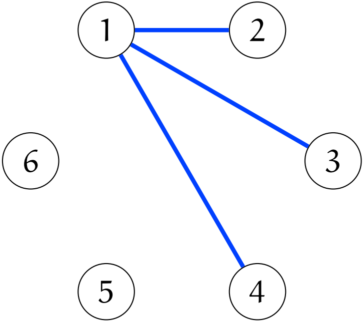
SAT solvers can determine that $R(4) = 18$ in 1 second using symmetry breaking; w/o symmetry breaking it requires weeks.

Symmetry breaking can be validated using RAT [CADE'15]

Ramsey Numbers

Ramsey Number $R(k)$: What is the smallest n such that any graph with n vertices has either a clique or a co-clique of size k?

$$\begin{aligned} R(3) &= 6 \\ R(4) &= 18 \\ 43 \leq \quad R(5) &\leq 48 \end{aligned}$$



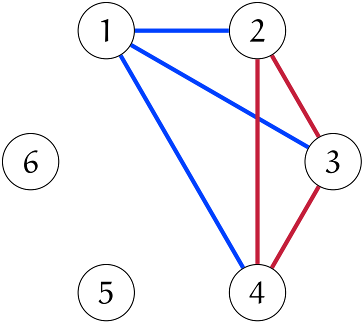
SAT solvers can determine that $R(4) = 18$ in 1 second using symmetry breaking; w/o symmetry breaking it requires weeks.

Symmetry breaking can be validated using RAT [CADE'15]

Ramsey Numbers

Ramsey Number $R(k)$: What is the smallest n such that any graph with n vertices has either a clique or a co-clique of size k?

$$
\begin{aligned}
R(3) &= 6 \\
R(4) &= 18 \\
43 \leq \; R(5) &\leq 48
\end{aligned}
$$



SAT solvers can determine that $R(4) = 18$ in 1 second using symmetry breaking; w/o symmetry breaking it requires weeks.

Symmetry breaking can be validated using RAT [CADE'15]

SR Proof of Ramsey Number Three (I)

Variable $e_{i,j}$: edge (i, j) is colored red ($\top$) or blue ($\bot$)

Step one: sort the edges adjacent to vertex v_1 (blue first).

- $(\neg e_{1,2} \vee e_{1,3})$
- τ applies the mapping $(v_2, v_3)(v_3, v_2)$ or specially
- $\tau := \{e_{1,2} = \bot, e_{1,3} = \top, e_{2,4} = e_{3,4}, e_{3,4} = e_{2,4}, e_{2,5} = e_{3,5}, e_{3,5} = e_{2,5}\}$

SR Proof of Ramsey Number Three (I)

Variable $e_{i,j}$: edge (i, j) is colored red ($\top$) or blue ($\bot$)

Step one: sort the edges adjacent to vertex v_1 (blue first).

- $(\neg e_{1,2} \vee e_{1,3})$
- τ applies the mapping $(v_2, v_3)(v_3, v_2)$ or specially
- $\tau := \{e_{1,2} = \bot, e_{1,3} = \top, e_{2,4} = e_{3,4}, e_{3,4} = e_{2,4}, e_{2,5} = e_{3,5}, e_{3,5} = e_{2,5}\}$

- $(\neg e_{1,3} \vee e_{1,4})$
- τ applies the mapping $(v_2, v_3, v_4)(v_3, v_4, v_2)$

SR Proof of Ramsey Number Three (I)

Variable $e_{i,j}$: edge (i, j) is colored red ($\top$) or blue ($\bot$)

Step one: sort the edges adjacent to vertex v_1 (blue first).

- $(\neg e_{1,2} \vee e_{1,3})$
- τ applies the mapping $(v_2, v_3)(v_3, v_2)$ or specially
- $\tau := \{e_{1,2} = \bot, e_{1,3} = \top, e_{2,4} = e_{3,4}, e_{3,4} = e_{2,4}, e_{2,5} = e_{3,5}, e_{3,5} = e_{2,5}\}$

- $(\neg e_{1,3} \vee e_{1,4})$
- τ applies the mapping $(v_2, v_3, v_4)(v_3, v_4, v_2)$

- $(\neg e_{1,4} \vee e_{1,5})$
- τ applies the mapping $(v_2, v_3, v_4, v_5)(v_3, v_4, v_5, v_2)$

SR Proof of Ramsey Number Three (I)

Variable $e_{i,j}$: edge (i,j) is colored red ($\top$) or blue ($\bot$)

Step one: sort the edges adjacent to vertex v_1 (blue first).

- $(\neg e_{1,2} \vee e_{1,3})$
- τ applies the mapping $(v_2, v_3)(v_3, v_2)$ or specially
- $\tau := \{e_{1,2} = \bot, e_{1,3} = \top, e_{2,4} = e_{3,4}, e_{3,4} = e_{2,4}, e_{2,5} = e_{3,5}, e_{3,5} = e_{2,5}\}$

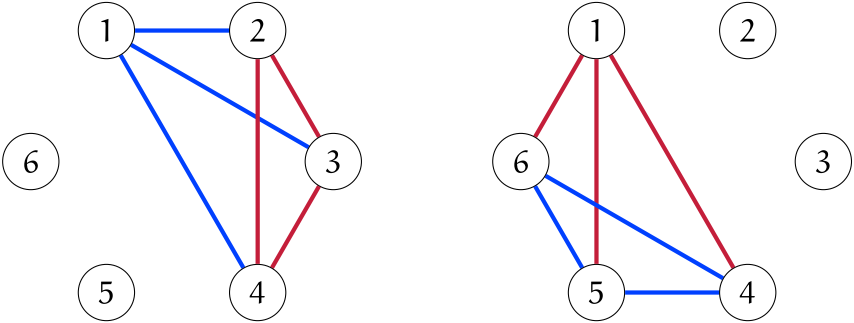
- $(\neg e_{1,3} \vee e_{1,4})$
- τ applies the mapping $(v_2, v_3, v_4)(v_3, v_4, v_2)$

- $(\neg e_{1,4} \vee e_{1,5})$
- τ applies the mapping $(v_2, v_3, v_4, v_5)(v_3, v_4, v_5, v_2)$

- $(\neg e_{1,5} \vee e_{1,6})$
- τ applies the mapping $(v_2, v_3, v_4, v_5, v_6)(v_3, v_4, v_5, v_6, v_2)$

SR Proof of Ramsey Number Three (II)

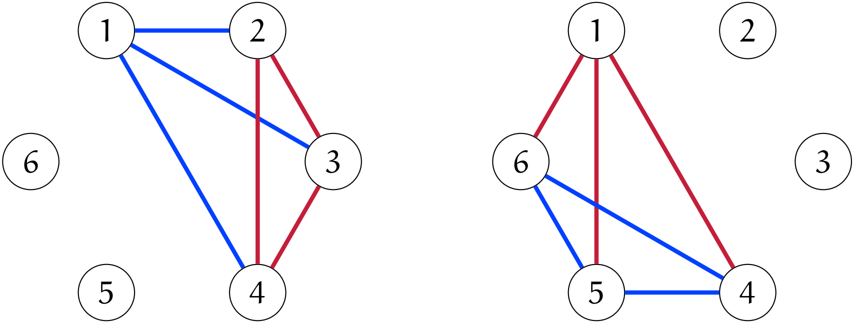
With sorted edges you can learn $(\neg e_{1,4})$ and $(e_{1,4})$ directly



The end result is a 6-clause SR proof for $R(3) \leq 6$

SR Proof of Ramsey Number Three (II)

With sorted edges you can learn $(\neg e_{1,4})$ and $(e_{1,4})$ directly



The end result is a 6-clause SR proof for $R(3) \leq 6$

Similarly, a 38-clause SR proof can be constructed for $R(4) \leq 18$.

Only Useful for Symmetry Breaking?

The hardness of the example formulas were due symmetries

- Global symmetries (e.g. Ramsey)
- Local symmetries (e.g. mutilated chessboards)

The WLOS proof systems can break symmetries

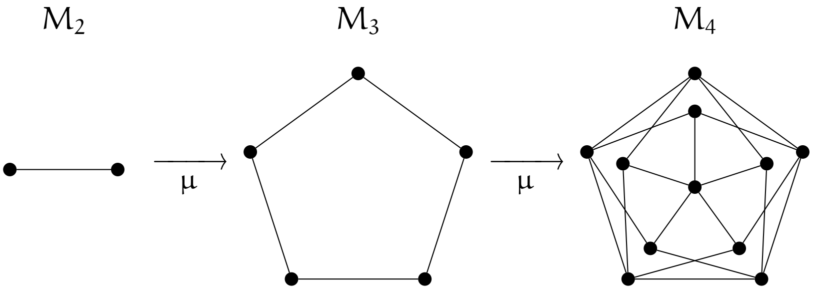
- without new variables
- ... although sometimes it requires the strength of SR

Can they also provide strong non-symmetry reasoning?

Mycielski Graphs

M_k is a triangle-free graph with chromatic number k

- M_2: a path of length 2
- M_3: a cycle of length 5
- M_i can be constructed from M_{i+1} using operation μ
- These graph coloring problems are extremely hard



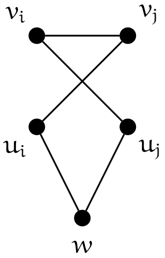
Mycielski Operation

M_k is a triangle-free graph with chromatic number k

- M_2: a path of length 2
- M_3: a cycle of length 5
- M_i can be constructed from M_{i+1} using operation μ

Operation μ works as follows on graph $G = (V, E)$:

1. Start with G
2. For $v_i \in V$, create a copy vertex u_i
3. For $(v_i, v_j) \in E$, add edges (v_i, u_j), (u_i, v_j)
4. Add a vertex w and all edges (w, u_i)



Short PR Proofs of Mycielski Graphs

Short PR proofs can be constructed using this observation:

- If no neighbor of u_i has color c, then u_i can have color c
- If a vertex v_i has color c and its corresponding w doesn't, then no neighbor of the corresponding u_i has color c.
- So the above is enough to color u_i with color c
- The PR clauses $(v_{i,c} \vee \neg w_{i,c} \vee u_{i,c})$ enable a linear-size proof
- Repair: $u_{i,c} = \top$ and $u_{i,d} = \bot$ for all $d \neq c$