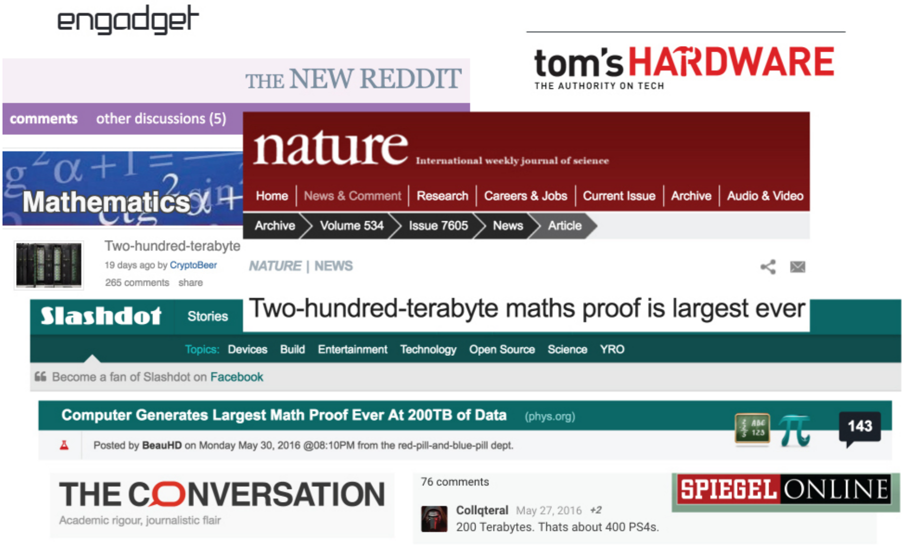
Logic and Mechanized Reasoning

Computer-Generated Propositional Proofs

Marijn J.H. Heule

Carnegie Mellon University

“The Largest Math Proof Ever”



engadget

THE NEW REDDIT

comments other discussions (5)

Mathematics

Two-hundred-terabyte

19 days ago by CryptoBeer

265 comments share

tom's HARDWARE
THE AUTHORITY ON TECH

nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video

Archive > Volume 534 > Issue 7605 > News > Article

NATURE | NEWS

Two-hundred-terabyte maths proof is largest ever

Slashdot Stories

Topics: Devices Build Entertainment Technology Open Source Science YRO

Become a fan of Slashdot on Facebook

Computer Generates Largest Math Proof Ever At 200TB of Data (phys.org)

143

Posted by BeauHD on Monday May 30, 2016 @08:10PM from the red-pill-and-blue-pill dept.

THE CONVERSATION
Academic rigour, journalistic flair

76 comments

Collqteral May 27, 2016 +2
200 Terabytes. Thats about 400 PS4s.

SPIEGEL ONLINE

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

Certifying Satisfiability and Unsatisfiability

- Certifying satisfiability of a formula is easy:

$$(p \vee q) \wedge (\neg p \vee \neg q) \wedge (\neg q \vee \neg r)$$

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

$\Gamma := (\neg p \vee \neg q \vee r) \wedge (\neg r) \wedge (p \vee \neg q) \wedge (\neg s \vee q) \wedge (s)$

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

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

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.

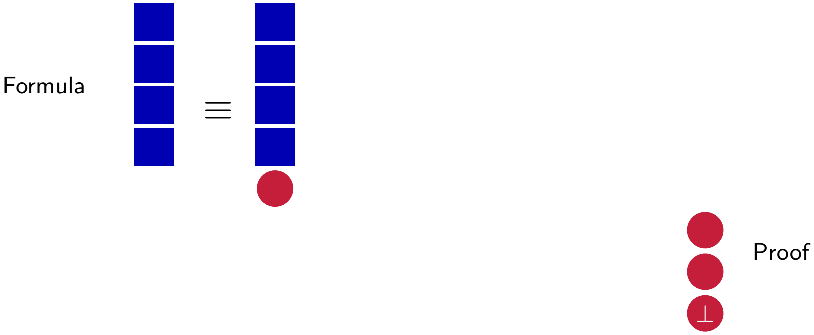
Clausal Proofs

Reduce the size of the proof by only storing added clauses

Formula
Proof
$\perp$

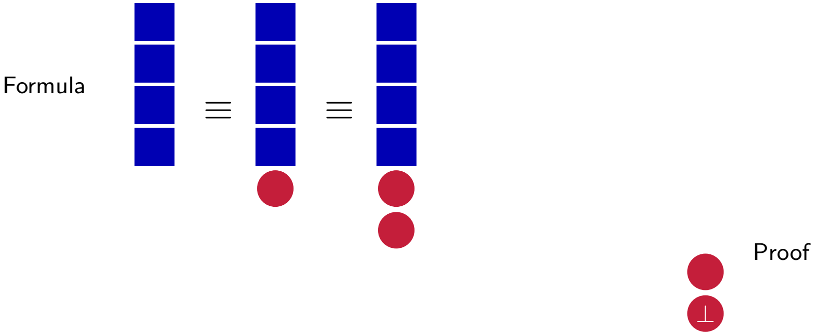
Clausal Proofs

Reduce the size of the proof by only storing added clauses



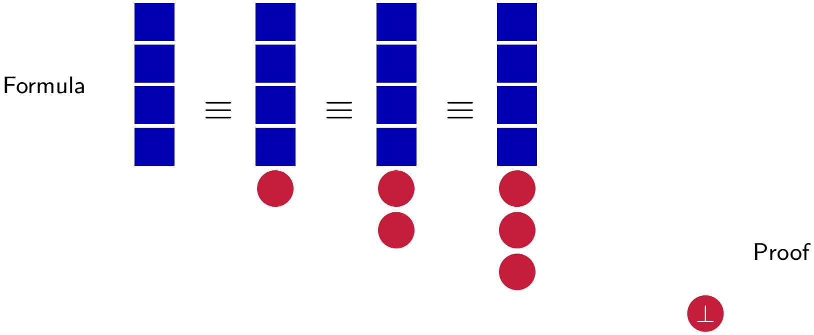
Clausal Proofs

Reduce the size of the proof by only storing added clauses



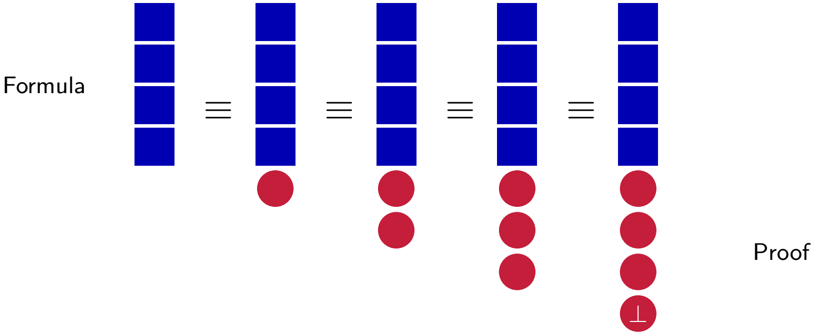
Clausal Proofs

Reduce the size of the proof by only storing added clauses



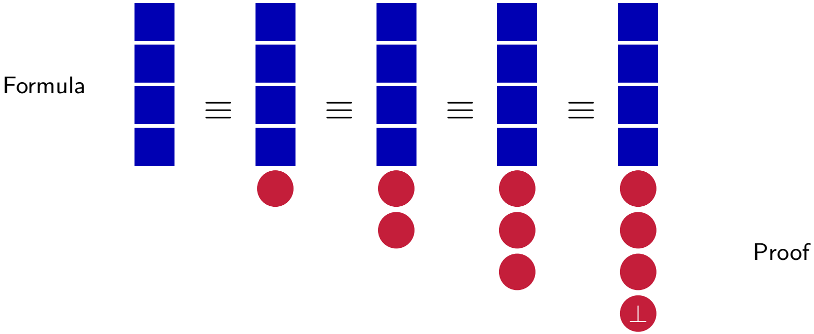
Clausal Proofs

Reduce the size of the proof by only storing added clauses



Clausal Proofs

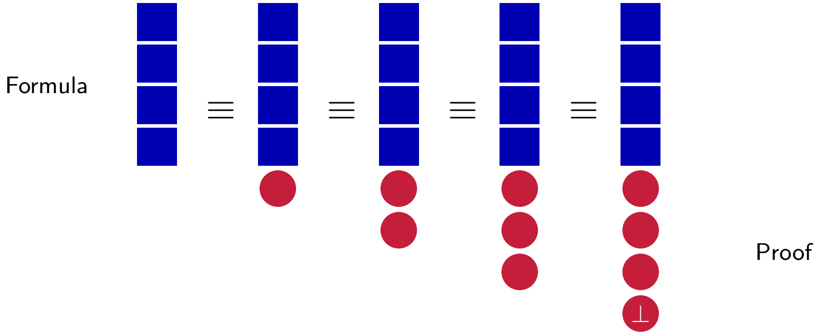
Reduce the size of the proof by only storing added clauses



- 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

Proof-Producing Tools

Beyond NP

Traditional Proofs vs. Interference-Based Proofs

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

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

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.

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

➥ 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:

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

- 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)?

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

Resolution proofs of PHP_n must be exponential [Haken 1985]

Cook constructed polynomial-sized ER proofs of PHP_n

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

Blocked Clauses [Kullmann 1999]

Definition (Blocked Clause)

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

Example

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

Example

Consider the formula $(p \lor q) \land (p \lor \neg q \lor \neg r) \land (\neg p \lor r)$*.*
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 $C \lor p$ that is blocked on p w.r.t. Γ. Let assignment τ satisfy Γ, but falsify $C \lor p$. Note that all clauses $D \lor \neg p$ are doubly satisfied by τ. Flipping p to true in τ satisfies both Γ and $C \lor p$.

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

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

- 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 $(p \lor q) \land (p \lor \neg q \lor \neg r) \land (\neg p \lor r)$*.*
After removing either $(p \lor \neg q \lor \neg r)$ *or* $(\neg p \lor r)$*, the clause* $(p \lor q)$ *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)

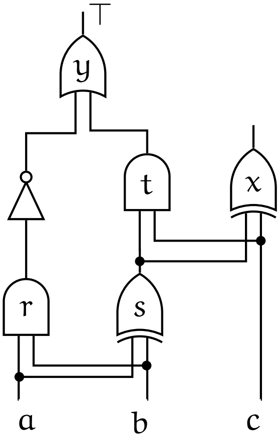
$(\neg y \lor t \lor \neg r)$
$(y \lor \neg t)$
$(y \lor r)$

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

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

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

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



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)

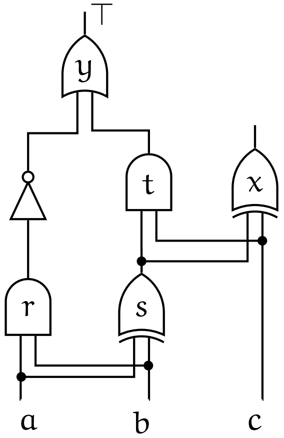
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)

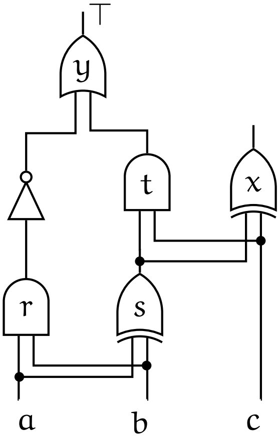
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)

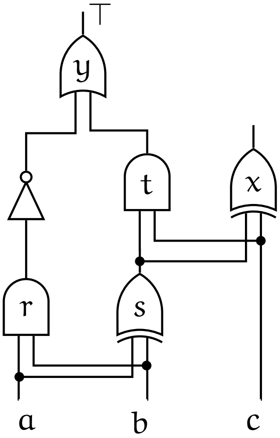
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)

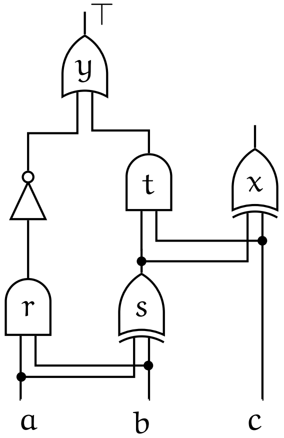
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)

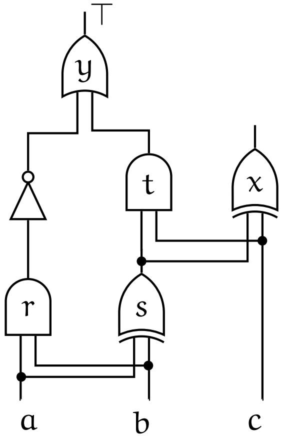
$(\neg y \vee t \vee \neg r)$
~~$(y \vee \neg t)$~~
~~$(y \vee r)$~~

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

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

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

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



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)

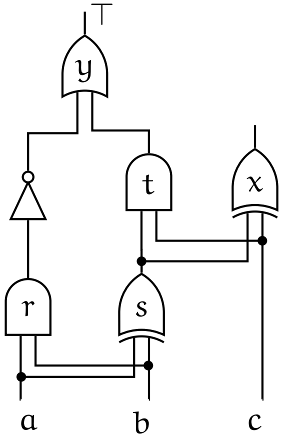
$(\neg y \lor t \lor \neg r)$
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)

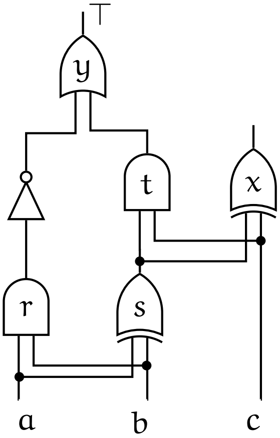
~~$(\neg y \lor t \lor \neg r)$~~
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)~~

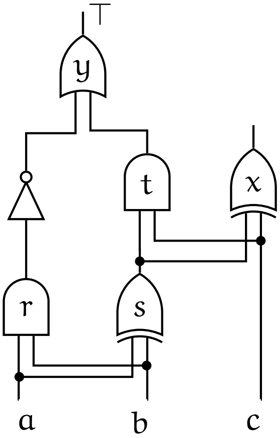
~~$(\neg y \lor t \lor \neg r)$~~
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)~~

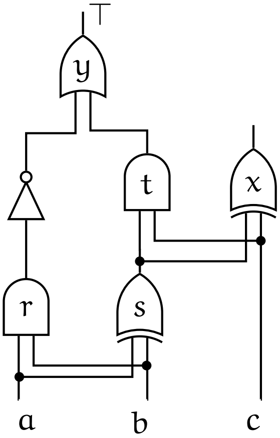
~~$(\neg y \lor t \lor \neg r)$~~
~~$(y \lor \neg t)$~~
~~$(y \lor r)$~~

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

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

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

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



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)~~

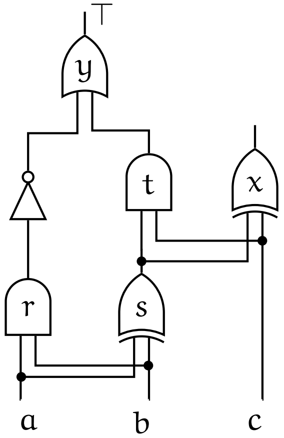
~~$(\neg y \vee t \vee \neg r)$~~
~~$(y \vee \neg t)$~~
~~$(y \vee r)$~~

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

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

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

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



Solution Reconstruction / Repair

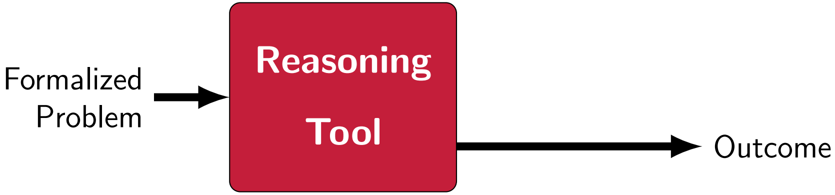
Input:

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

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

```
1: while S.size () do
2:     $\langle C, l \rangle$ := S.pop ()
3:     if $\tau$ falsifies $C$ then $\tau := \tau \cup \{l = \top\}$
4: end while
5: return $\tau$
```

Automated Reasoning Programs



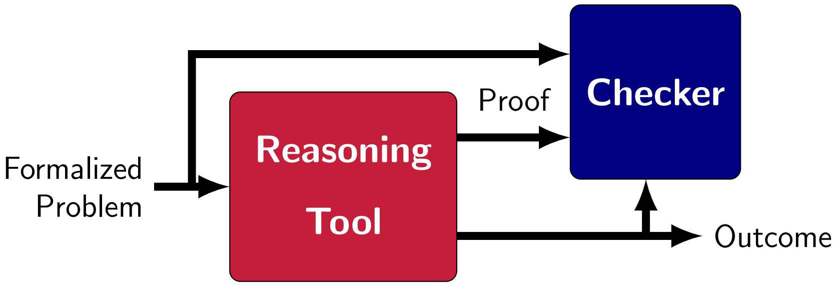
Standard Implementations

- Lingering doubt about whether result can be trusted
- If find bug in tool, must rerun all prior verifications

Formally Verified Tools

- Hard to develop
- Hard to make scalable

Proof-Generating Automated Reasoning Programs



Proof-Generating Tools

- Only need to prove individual executions, not entire program
- Can have bugs in tool but still trust result
- Can we trust the checker?
  - Simple algorithms and implementation
  - Ideally formally verified

Proof-Generating Tools: Arbitrarily Complex Solvers

Proof-generating tools with verified checkers is a powerful idea:

- Don't worry about correctness or completeness of tools;
- Facilitates making tools more complex and efficient; while
- Full confidence in results. [Heule, Hunt, Kaufmann, Wetzler '17]

Formally verified checkers now also used in industry

All Propositional Reasoning in One Rule

All SAT techniques can be expressed with a single rule:

- If every assignment τ that would satisfy formula Γ and falsify constraint C can be repaired into τ' that would satisfy $\Gamma \wedge C$, then adding C to Γ preserves satisfiability
- Restriction: Validity of repair checkable in polynomial time

All Propositional Reasoning in One Rule

All SAT techniques can be expressed with a single rule:

- If every assignment τ that would satisfy formula Γ and falsify constraint C can be repaired into τ' that would satisfy $\Gamma \wedge C$, then adding C to Γ preserves satisfiability
- Restriction: Validity of repair checkable in polynomial time

Example

Let $\Gamma = (p \vee q) \wedge (\neg p \vee \neg q)$ and constraint $C = (p \vee \neg q)$

- Only $\tau := \{p = 0, q = 1\}$ satisfies Γ and falsifies C
- This assignment can be repaired by making $p = 1, q = 0$
- So Γ and $\Gamma \wedge C$ are equisatisfiable

All Propositional Reasoning in One Rule

All SAT techniques can be expressed with a single rule:

- If every assignment τ that would satisfy formula Γ and falsify constraint C can be repaired into τ' that would satisfy $\Gamma \wedge C$, then adding C to Γ preserves satisfiability
- Restriction: Validity of repair checkable in polynomial time

Example

Let $\Gamma = (p \vee q) \wedge (\neg p \vee \neg q)$ and constraint $C = (p \vee \neg q)$

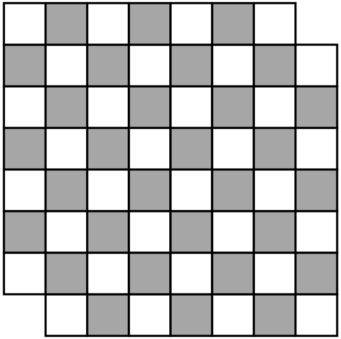
- Only $\tau := \{p = 0, q = 1\}$ satisfies Γ and falsifies C
- This assignment can be repaired by making $p = 1, q = 0$
- So Γ and $\Gamma \wedge C$ are equisatisfiable

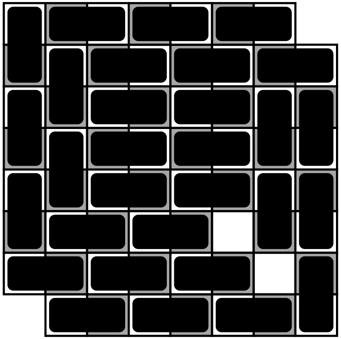
A proof is a sequence of constraint-repair pairs ending with $\perp, \{\}$

- Check each constraint-repair pair in polynomial time
- Extend the formula with the new constraint

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

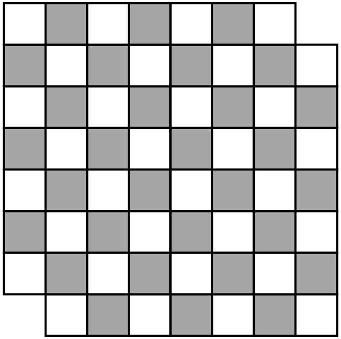
Can a chessboard be fully covered with dominos after removing two diagonally opposite corner squares?

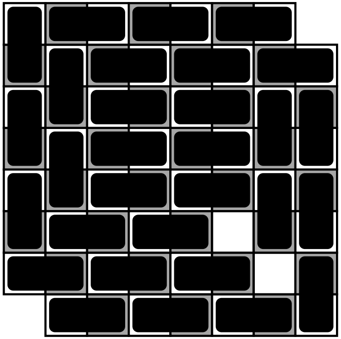




Mutilated Chessboards: “A Tough Nut to Crack” [McCarthy]

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



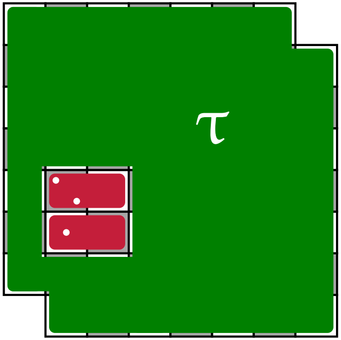


Easy to refute based on the following two observations:

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

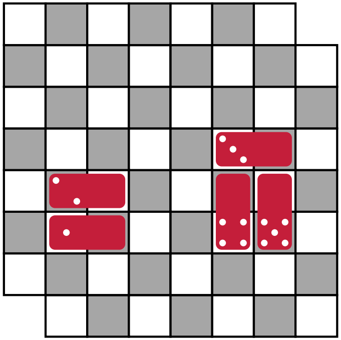
An Alternative Argument for Mutilated Chessboards

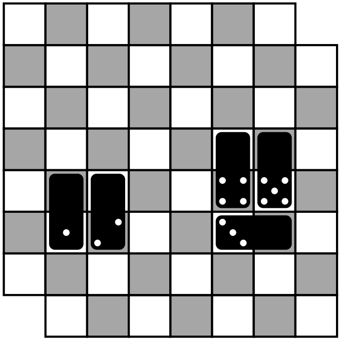
The chessboard pattern argument is challenging to find, but an alternative short argument can be found automatically...



- Γ: the problem constraints
- C: no two horizontal dominos on top of each other
- τ: "satisfies" Γ, falsifies C
- repair: replace the horizontal dominos by vertical dominos covering the same cells
- Γ and $\Gamma \wedge C$ equisatisfiable

Symbolic AI is Local Reasoning





These constraint-repair patterns can be automatically detected

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

Symbolic AI tools produce proofs that can be very different compared to human-made proofs for the same problem

Beyond NP: The Collatz Conjecture

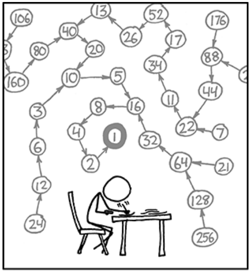
Resolving foundational algorithm questions

$$\mathrm{Col}(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (3n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

Does `while(n > 1) n = Col(n);` terminate?

Find a non-negative function $\mathrm{fun}(n)$ s.t.

$$\forall n > 1 : \mathrm{fun}(n) > \mathrm{fun}(\mathrm{Col}(n))$$



THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

source: xkcd.com/710

Beyond NP: The Collatz Conjecture

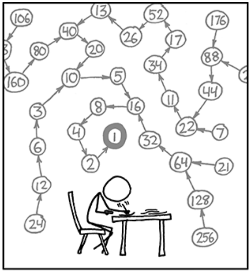
Resolving foundational algorithm questions

$$\mathrm{Col}(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (3n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

Does $\mathtt{while}(n > 1)\ n = \mathrm{Col}(n);$ terminate?

Find a non-negative function $\mathrm{fun}(n)$ s.t.

$$\forall n > 1 : \mathrm{fun}(n) > \mathrm{fun}(\mathrm{Col}(n))$$



THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

source: xkcd.com/710

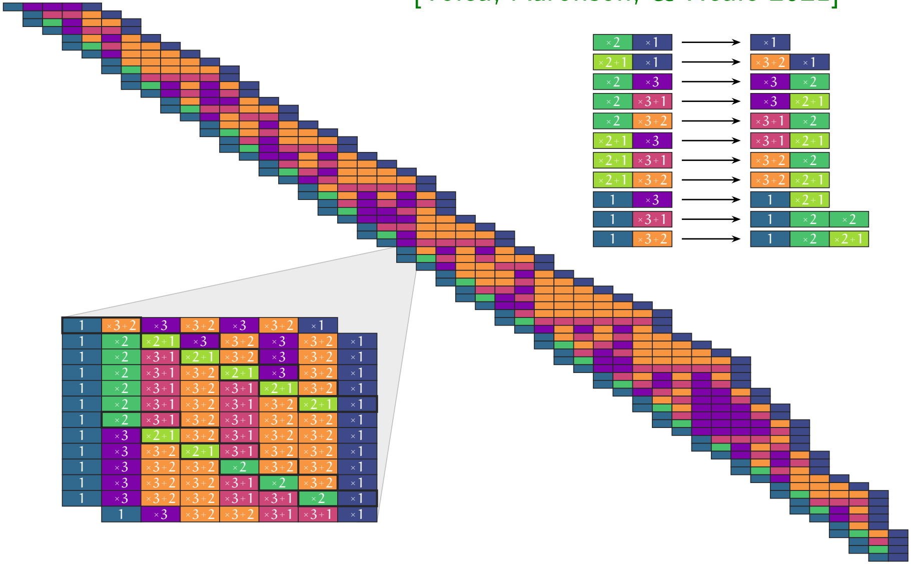
Can we construct a function s.t. $\mathrm{fun}(n) > \mathrm{fun}(\mathrm{Col}(n))$ holds?

$\mathrm{fun}(3)$	$\mathrm{fun}(5)$	$\mathrm{fun}(8)$	$\mathrm{fun}(4)$	$\mathrm{fun}(2)$	$\mathrm{fun}(1)$
5	4	3	2	1	0

Collatz Conjecture: Studying a Rewrite System

[Yolcu, Aaronson, & Heule 2021]

×2	×1	→	×1		
×2+1	×1	→	×3+2	×1	
×2	×3	→	×3	×2	
×2	×3+1	→	×3	×2+1	
×2	×3+2	→	×3+1	×2	
×2+1	×3	→	×3+1	×2+1	
×2+1	×3+1	→	×3+2	×2	
×2+1	×3+2	→	×3+2	×2+1	
1	×3	→	1	×2+1	
1	×3+1	→	1	×2	×2
1	×3+2	→	1	×2	×2+1



Collatz Conjecture: Successes and Challenges

Success. Rewrite system with 11 rules:

- Their termination solves Collatz.
- Our tool proves the termination of any subset of 10 rules.

Collatz Conjecture: Successes and Challenges

Success. Rewrite system with 11 rules:

- Their termination solves Collatz.
- Our tool proves the termination of any subset of 10 rules.

Success. Our tool proves termination of Farkas' variant:

$$F(n) = \begin{cases} \frac{n-1}{3} & \text{if } n \equiv 1 \pmod 3 \\ \frac{n}{2} & \text{if } n \equiv 0 \text{ or } n \equiv 2 \pmod 6 \\ \frac{3n+1}{2} & \text{if } n \equiv 3 \text{ or } n \equiv 5 \pmod 6 \end{cases}$$

Collatz Conjecture: Successes and Challenges

Success. Rewrite system with 11 rules:

- Their termination solves Collatz.
- Our tool proves the termination of any subset of 10 rules.

Success. Our tool proves termination of Farkas' variant:

$$F(n) = \begin{cases} \frac{n-1}{3} & \text{if } n \equiv 1 \pmod 3 \\ \frac{n}{2} & \text{if } n \equiv 0 \text{ or } n \equiv 2 \pmod 6 \\ \frac{3n+1}{2} & \text{if } n \equiv 3 \text{ or } n \equiv 5 \pmod 6 \end{cases}$$

Challenge ($500). An easier generalized Collatz problem is open:

$$H(n) = \begin{cases} \frac{3n}{4} & \text{if } n \equiv 0 \pmod 4 \\ \frac{9n+1}{8} & \text{if } n \equiv 7 \pmod 8 \\ \perp & \text{otherwise} \end{cases}$$

Takeaways

Successes, Advances, and Trust:

- A performance boost of symbolic AI technology allows solving challenges in mathematics
- Creative, but possibly gigantic proofs can be validated using formally-verified checkers
- Future: combine symbolic AI and ML

Challenges ready for symbolic AI + ML?

- Chromatic number of the plane
- Optimal matrix multiplication
- Hadamard conjecture
- Collatz conjecture
- …