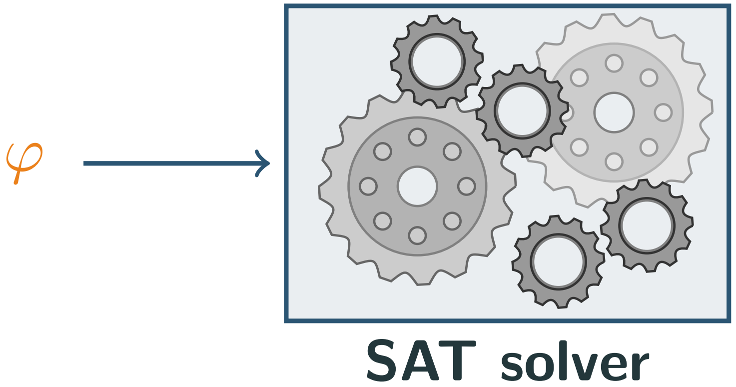
SAT Solving and Its Applications

SAT Solving

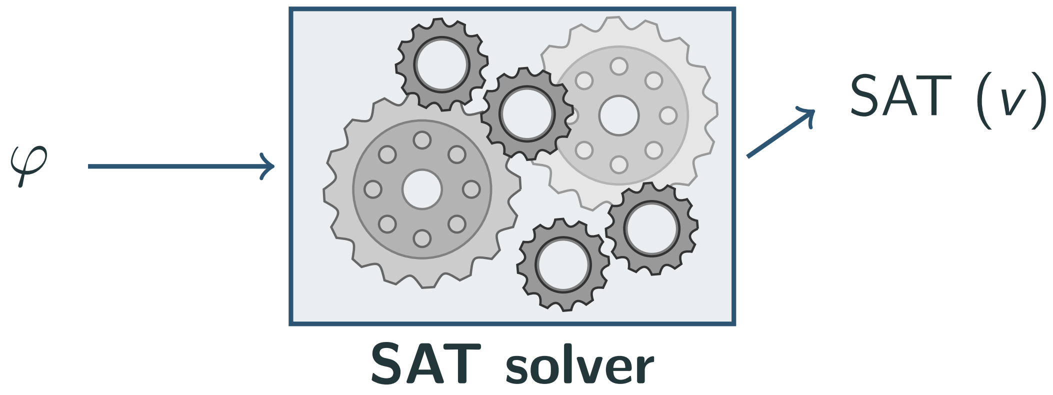
input: propositional formula φ



SAT Solving

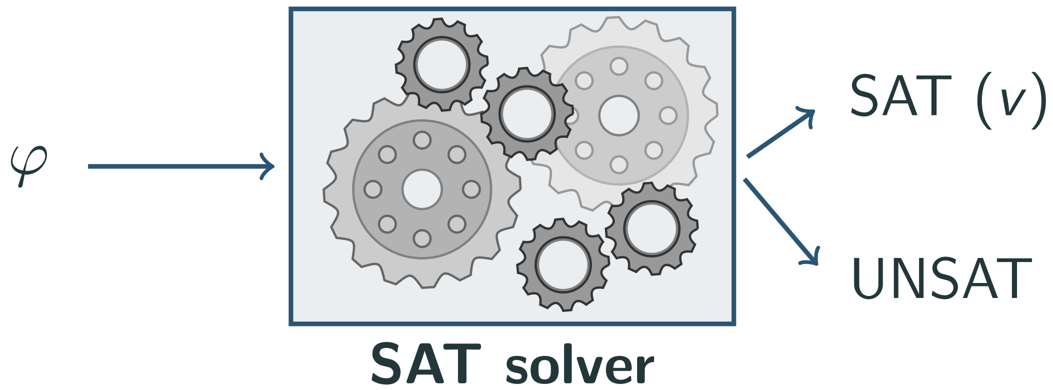
input: propositional formula φ

output: SAT + valuation v such that $v(\varphi) = T$ if φ satisfiable



SAT Solving

input:	propositional formula φ	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ satisfiable
	UNSAT	otherwise



SAT Solving

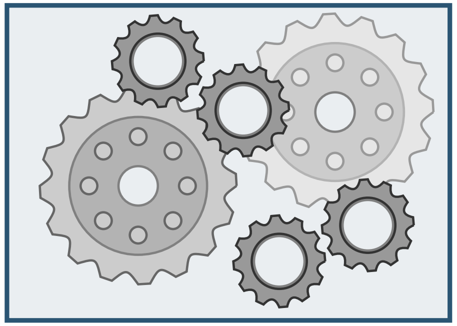
input: propositional formula φ

output: SAT + valuation v such that $v(\varphi) = T$ if φ satisfiable

UNSAT otherwise

$\varphi \longrightarrow$

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



SAT solver

SAT (v)

$v(p) = \mathrm{T}$
$v(q) = \mathrm{F}$
$v(r) = \mathrm{F}$

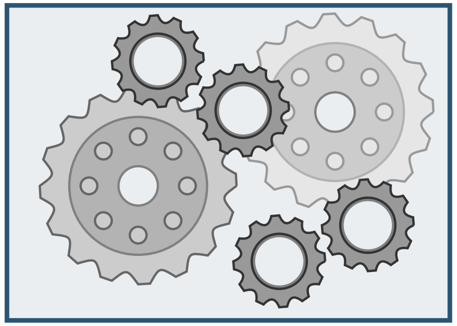
UNSAT

SAT Solving

input:	propositional formula φ	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ satisfiable
	UNSAT	otherwise

φ

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



SAT solver

SAT (v) $\quad v(p) = \mathsf{T}$, $v(q) = \mathsf{F}$, $v(r) = \mathsf{F}$

UNSAT

Terminology

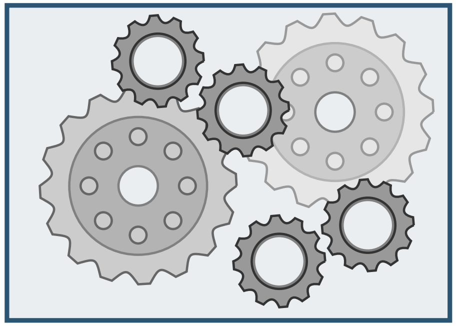
- decision problem P is problem with answer yes or no

SAT Solving

input:	propositional formula φ	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ satisfiable
	UNSAT	otherwise

φ $\longrightarrow$

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



SAT solver

SAT (v) $\quad v(p) = \mathrm{T}$, $v(q) = \mathrm{F}$, $v(r) = \mathrm{F}$

UNSAT

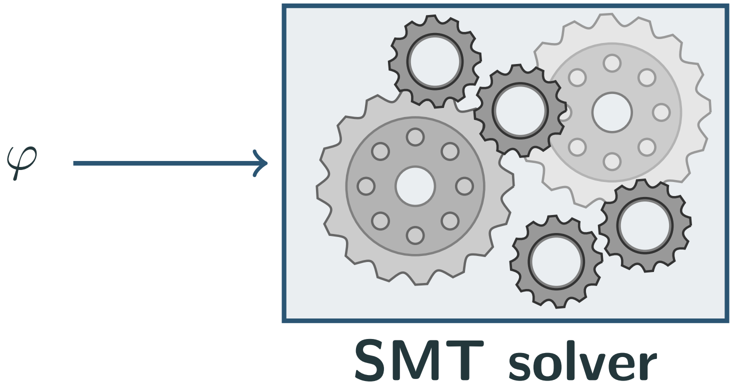
Terminology

- decision problem P is problem with answer yes or no
- SAT encoding of decision problem P is propositional formula φ_P such that

$$\text{answer to } P \text{ is yes} \iff \varphi_P \text{ is satisfiable}$$

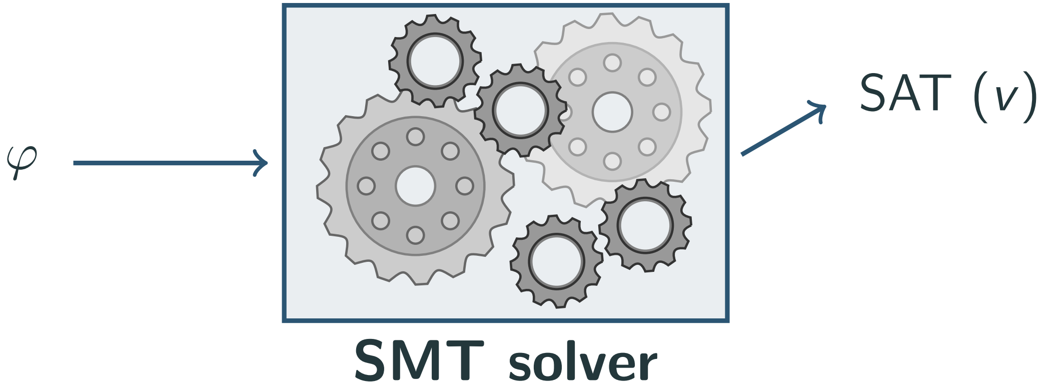
SMT Solving

input: formula φ involving theory T



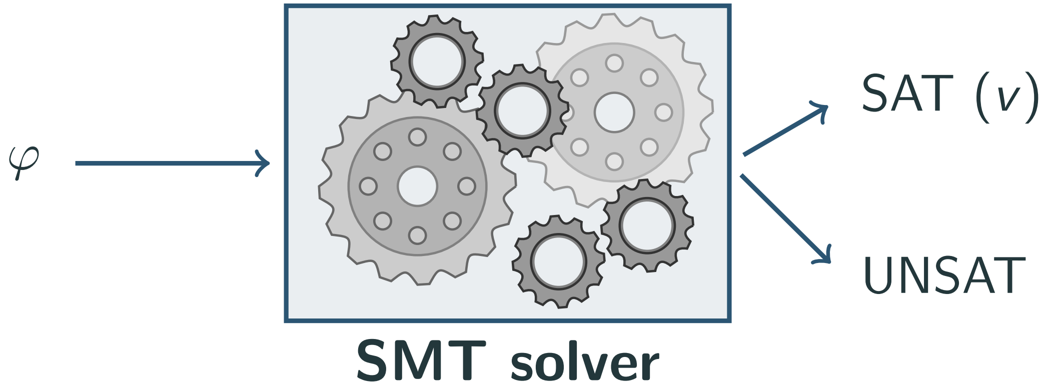
SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable



SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

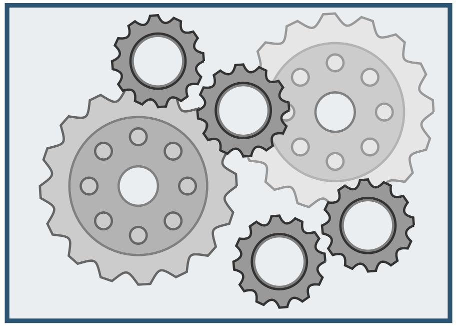


SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

φ

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (v) $\quad v(a) = 3 \quad v(b) = 0 \quad v(c) = 0 \quad v(p) = \mathrm{T}$

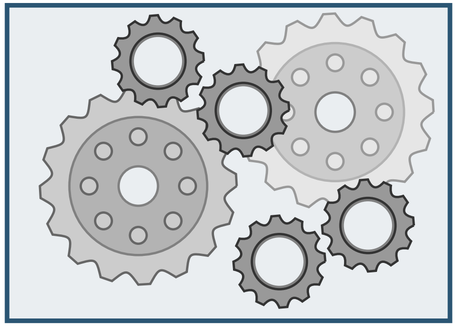
UNSAT

SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

φ

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (v) $\quad v(a) = 3 \quad v(b) = 0 \quad v(c) = 0 \quad v(p) = \mathrm{T}$

UNSAT

Example (Theories)

- arithmetic $\qquad 2a + b \geqslant c \vee (a = 0 \wedge p)$

SMT Solving

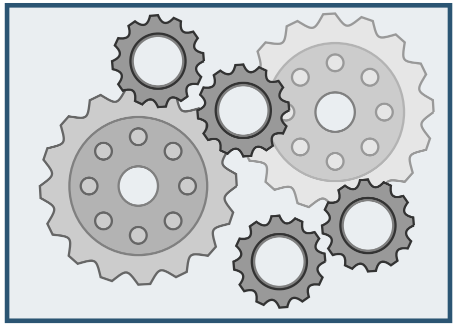
input: formula φ involving theory T

output: SAT + valuation v such that $v(\varphi) = T$ if φ is T-satisfiable

UNSAT otherwise

φ

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (v) $v(a) = 3$ $v(b) = 0$ $v(c) = 0$ $v(p) = \mathrm{T}$

UNSAT

Example (Theories)

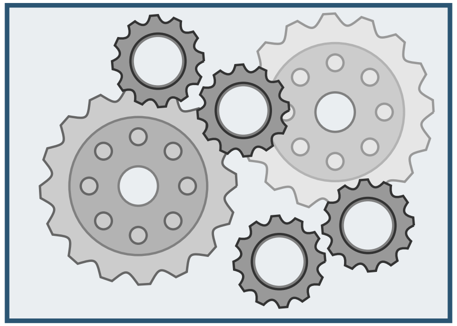
- arithmetic $2a + b \geqslant c \vee (a = 0 \wedge p)$
- uninterpreted functions $\mathrm{f}(x, y) \neq \mathrm{f}(y, x) \wedge \mathrm{g}(\mathrm{f}(x, x)) = \mathrm{g}(y)$

SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

φ $\longrightarrow$

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (v) $\quad v(a) = 3 \quad v(b) = 0 \quad v(c) = 0 \quad v(p) = \mathrm{T}$

UNSAT

Example (Theories)

- arithmetic $\quad 2a + b \geqslant c \vee (a = 0 \wedge p)$
- uninterpreted functions $\quad \mathrm{f}(x, y) \neq \mathrm{f}(y, x) \wedge \mathrm{g}(\mathrm{f}(x, x)) = \mathrm{g}(y)$
- bit vectors $\quad ((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$

SMT Solving

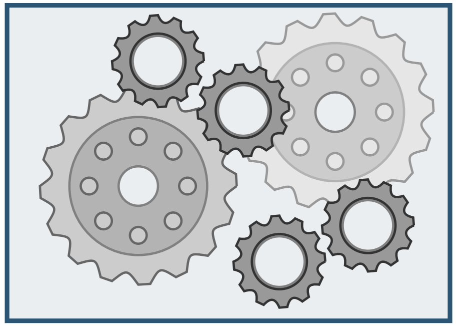
input: formula φ involving theory T

output: SAT + valuation v such that $v(\varphi) = T$ if φ is T-satisfiable

UNSAT otherwise

φ

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (v) $v(a) = 3$ $v(b) = 0$ $v(c) = 0$ $v(p) = \mathrm{T}$

UNSAT

Example (Theories)

- arithmetic $2a + b \geqslant c \vee (a = 0 \wedge p)$
- uninterpreted functions $\mathrm{f}(x, y) \neq \mathrm{f}(y, x) \wedge \mathrm{g}(\mathrm{f}(x, x)) = \mathrm{g}(y)$
- bit vectors $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$

Terminology

- SMT encoding over theory T of decision problem P is formula φ_P such that

$$\text{answer to } P \text{ is yes} \quad \Longleftrightarrow \quad \varphi_P \text{ is satisfiable}$$

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions “main questions” with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 “hard”, “medium”, and “easy” main questions

- how can software find valid question set?

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

- how can software find valid question set?

SAT Encoding

- variables q_i for $1 \leqslant i \leqslant 1500$

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

- how can software find valid question set?

SAT Encoding

- variables q_i for $1 \leqslant i \leqslant 1500$
- idea: valuation v sets $v(q_i) = \mathrm{T}$ if question i is included, $v(q_i) = \mathrm{F}$ otherwise

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

- how can software find valid question set?

SAT Encoding

- variables q_i for $1 \leqslant i \leqslant 1500$
- idea: valuation v sets $v(q_i) = \mathrm{T}$ if question i is included, $v(q_i) = \mathrm{F}$ otherwise
- $\sum_{i \in Q_{\text{xroads}}} q_i \geqslant 12$

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

- how can software find valid question set?

SAT Encoding

- variables q_i for $1 \leqslant i \leqslant 1500$
- idea: valuation v sets $v(q_i) = \mathrm{T}$ if question i is included, $v(q_i) = \mathrm{F}$ otherwise

- $\sum_{i \in Q_{\text{xroads}}} q_i \geqslant 12$
- $\sum_{i \in Q_{\text{pictures}}} q_i \geqslant 12$

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

- how can software find valid question set?

SAT Encoding

- variables q_i for $1 \leqslant i \leqslant 1500$
- idea: valuation v sets $v(q_i) = \mathrm{T}$ if question i is included, $v(q_i) = \mathrm{F}$ otherwise

- $\sum_{i \in Q_{\text{xroads}}} q_i \geqslant 12$
- $\sum_{i \in Q_{\text{pictures}}} q_i \geqslant 12$
- $\sum_{i \in Q_{\text{hard}}} q_i \geqslant 5$
- ...

Application: Driving License Test

Problem

Austrian driving license test consists of 80 questions out of 1500 such that the following conditions are satisfied:

- 30 questions "main questions" with 3 sub-questions each
- at least 12 main questions must be about crossroads
- at least 12 questions must have pictures
- at least 5 "hard", "medium", and "easy" main questions

- how can software find valid question set?

SAT Encoding

- variables q_i for $1 \leqslant i \leqslant 1500$
- idea: valuation v sets $v(q_i) = \mathrm{T}$ if question i is included, $v(q_i) = \mathrm{F}$ otherwise

- $\sum_{i \in Q_{\text{xroads}}} q_i \geqslant 12$
- $\sum_{i \in Q_{\text{pictures}}} q_i \geqslant 12$
- $\sum_{i \in Q_{\text{hard}}} q_i \geqslant 5$
- $\ldots$

Result

easy generation of valid question sets (with some random preselection)

Application: Pythagorean Triples

Problem

Can one color all natural numbers with two colors such that whenever $x^2 + y^2 = z^2$ not all of x, y, and z have same color?

Application: Pythagorean Triples

Problem

Can one color all natural numbers with two colors such that whenever $x^2 + y^2 = z^2$ not all of x, y, and z have same color?

Example

$$3^2 + 4^2 = 5^2 \qquad 5^2 + 12^2 = 13^2$$

(a)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✓
(b)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✗

Application: Pythagorean Triples

Problem

Can one color all natural numbers with two colors such that whenever $x^2 + y^2 = z^2$ not all of x, y, and z have same color?

Example

$$3^2 + 4^2 = 5^2 \qquad 5^2 + 12^2 = 13^2$$

(a)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✓
(b)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✗

SAT Encoding

- variables x_i for $1 \leqslant i \leqslant n$ such that x_i becomes true iff it is colored red

Application: Pythagorean Triples

Problem

Can one color all natural numbers with two colors such that whenever $x^2 + y^2 = z^2$ not all of x, y, and z have same color?

Example

$$3^2 + 4^2 = 5^2 \qquad 5^2 + 12^2 = 13^2$$

(a)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✓
(b)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✗

SAT Encoding

- variables x_i for $1 \leqslant i \leqslant n$ such that x_i becomes true iff it is colored red
- SAT encoding: for all $a^2 + b^2 = c^2$ include $(x_a \vee x_b \vee x_c) \wedge (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$ (+ symmetry breaking, simplification, heuristics)

Application: Pythagorean Triples

Problem

Can one color all natural numbers with two colors such that whenever $x^2 + y^2 = z^2$ not all of x, y, and z have same color?

Example

$$3^2 + 4^2 = 5^2 \qquad 5^2 + 12^2 = 13^2$$

(a)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✓
(b)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✗

SAT Encoding

- variables x_i for $1 \leqslant i \leqslant n$ such that x_i becomes true iff it is colored red
- SAT encoding: for all $a^2 + b^2 = c^2$ include $(x_a \vee x_b \vee x_c) \wedge (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$ (+ symmetry breaking, simplification, heuristics)

Result: No. Coloring exists only up to 7,825.

Application: Pythagorean Triples

Problem

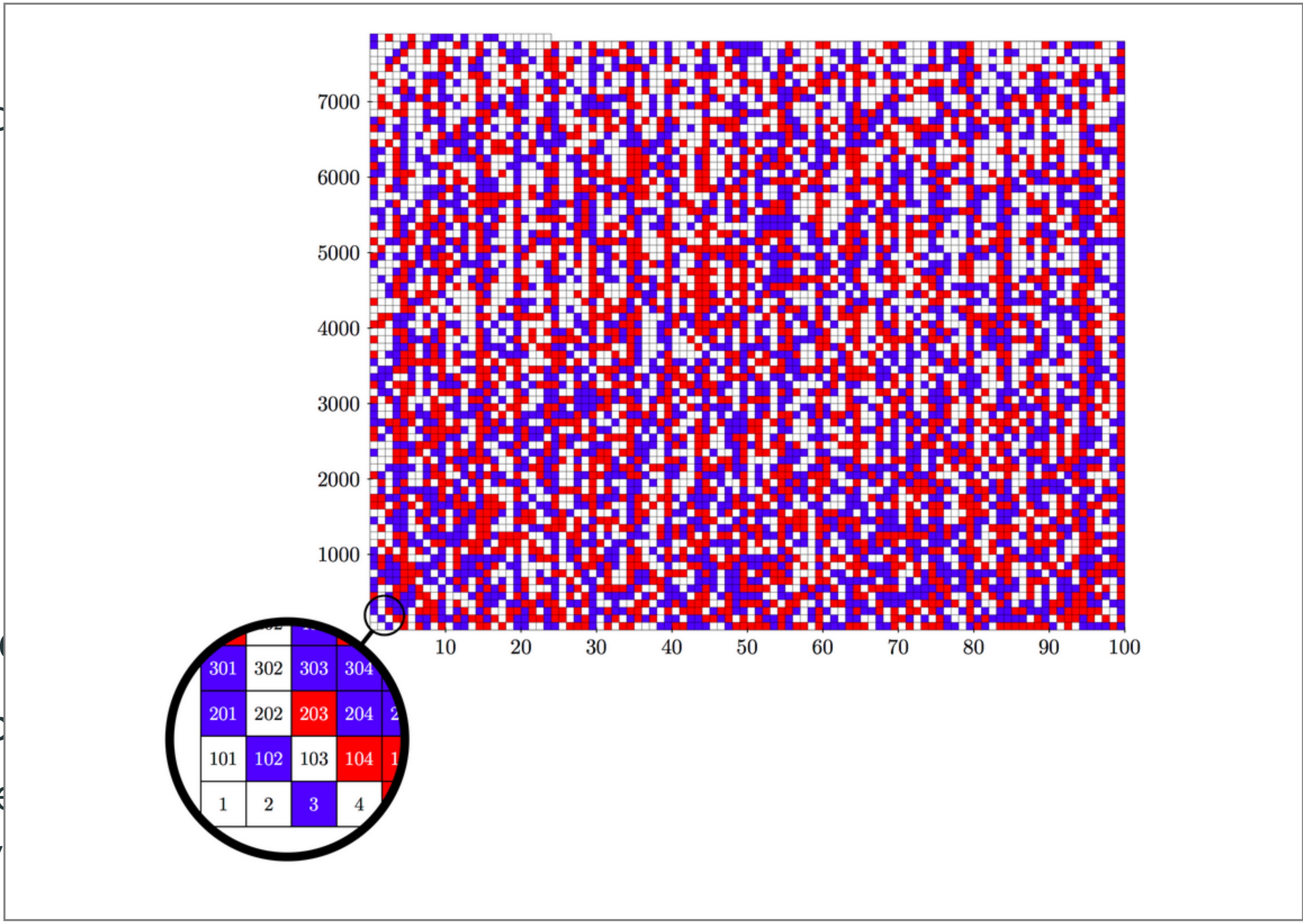
Can one co

$x^2 + y^2 =$

Example

(a) 1 … ✓

(b) 1 … ✗



SAT Enc

- ▶ variab … red
- ▶ SAT e … $_b \vee \bar{x}_c)$
  (+ sy

Result: No. Coloring exists only up to 7,825.

Application: Pythagorean Triples

Problem

Can one color all natural numbers with two colors such that whenever $x^2 + y^2 = z^2$ not all of x, y, and z have same color?

Example

$$3^2 + 4^2 = 5^2 \qquad 5^2 + 12^2 = 13^2$$

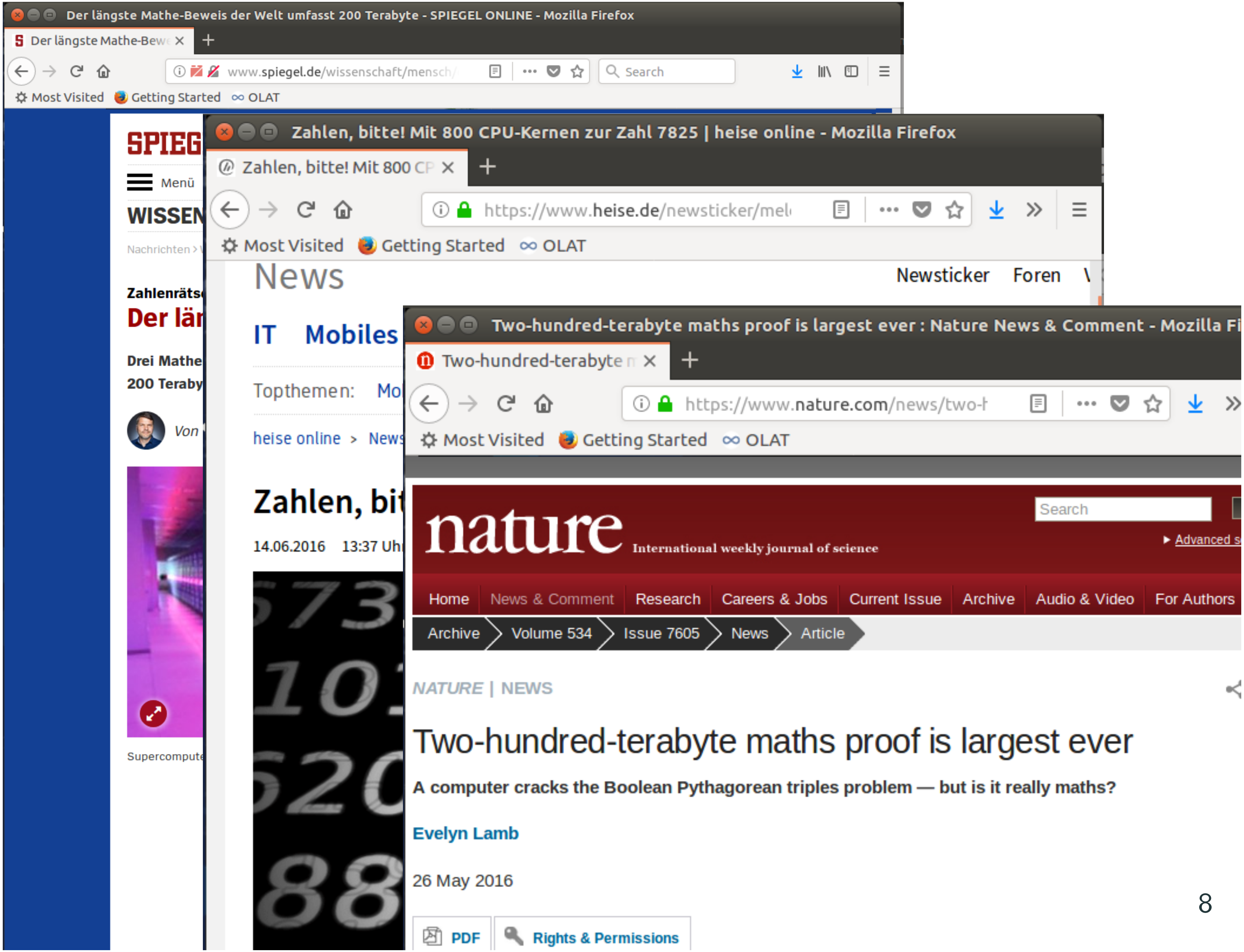
(a)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✓
(b)	1	2	3	4	5	6	7	8	9	10	11	12	13	...	✗

SAT Encoding

- variables x_i for $1 \leqslant i \leqslant n$ such that x_i becomes true iff it is colored red
- SAT encoding: for all $a^2 + b^2 = c^2$ include $(x_a \vee x_b \vee x_c) \wedge (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$ (+ symmetry breaking, simplification, heuristics)

Result: No. Coloring exists only up to 7,825.

1000s of variables, solving time 2 days with 800 processors, 200 TB of proof


Der längste Mathe-Beweis der Welt umfasst 200 Terabyte - SPIEGEL ONLINE - Mozilla Firefox
www.spiegel.de/wissenschaft/mensch/
Zahlen, bitte! Mit 800 CPU-Kernen zur Zahl 7825 | heise online - Mozilla Firefox
https://www.heise.de/newsticker/mel
News
Newsticker
Foren
IT
Mobiles
Topthemen:
heise online > News
Zahlen, bi
14.06.2016 13:37 Uh
Two-hundred-terabyte maths proof is largest ever : Nature News & Comment - Mozilla Fi
https://www.nature.com/news/two-t
nature
International weekly journal of science
Home
News & Comment
Research
Careers & Jobs
Current Issue
Archive
Audio & Video
For Authors
Archive
Volume 534
Issue 7605
News
Article
NATURE | NEWS
Two-hundred-terabyte maths proof is largest ever
A computer cracks the Boolean Pythagorean triples problem — but is it really maths?
Evelyn Lamb
26 May 2016
PDF
Rights & Permissions

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n-1$ rounds each
(+ venue restrictions, break restrictions, $\ldots$)

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n - 1$ rounds each (+ venue restrictions, break restrictions, ...)

Example (Österreichische Fußball-Bundesliga)

10 teams play in 4 periods (9 rounds each), periods 1 & 2 and 3 & 4 mirrored

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n-1$ rounds each (+ venue restrictions, break restrictions, ...)

Example (Österreichische Fußball-Bundesliga)

10 teams play in 4 periods (9 rounds each), periods 1 & 2 and 3 & 4 mirrored

(Part of) SAT Encoding

- variable x_{ijpr} is true if team i plays team j at home in period p, round r

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n-1$ rounds each (+ venue restrictions, break restrictions, . . .)

Example (Österreichische Fußball-Bundesliga)

10 teams play in 4 periods (9 rounds each), periods 1 & 2 and 3 & 4 mirrored

(Part of) SAT Encoding

- variable x_{ijpr} is true if team i plays team j at home in period p, round r
- $$\bigwedge_{i,p,r} \bigvee_{j \neq i} (x_{ijpr} \vee x_{jipr})$$ each team plays in every round

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n-1$ rounds each (+ venue restrictions, break restrictions, ...)

Example (Österreichische Fußball-Bundesliga)

10 teams play in 4 periods (9 rounds each), periods 1 & 2 and 3 & 4 mirrored

(Part of) SAT Encoding

- variable x_{ijpr} is true if team i plays team j at home in period p, round r
- $$\bigwedge_{i,p,r} \bigvee_{j \neq i} (x_{ijpr} \vee x_{jipr})$$ each team plays in every round

  $$\bigwedge_{i,p,r} \bigwedge_{j \neq i} \bigwedge_{k \neq i \wedge k \neq j} (x_{ijpr} \rightarrow \neg(x_{ikpr} \vee x_{kipr}))$$ each team plays at most once in every round

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n-1$ rounds each (+ venue restrictions, break restrictions, ...)

Example (Österreichische Fußball-Bundesliga)

10 teams play in 4 periods (9 rounds each), periods 1 & 2 and 3 & 4 mirrored

(Part of) SAT Encoding

- variable x_{ijpr} is true if team i plays team j at home in period p, round r
- $$\bigwedge_{i,p,r} \bigvee_{j \neq i} (x_{ijpr} \vee x_{jipr})$$ each team plays in every round
  $$\bigwedge_{i,p,r} \bigwedge_{j \neq i} \bigwedge_{k \neq i \wedge k \neq j} (x_{ijpr} \to \neg(x_{ikpr} \vee x_{kipr}))$$ each team plays at most once in every round
  $$\bigwedge_{i,j,r} (x_{ij1r} \to x_{ji2r}) \wedge (x_{ij3r} \to x_{ji4r})$$ mirror rounds 1& 2 and 3& 4

Application: Tournament Scheduling

Problem: Round Robin Scheduling

Schedule sports league tournament for n teams, p periods of $n-1$ rounds each (+ venue restrictions, break restrictions, . . .)

Example (Österreichische Fußball-Bundesliga)

10 teams play in 4 periods (9 rounds each), periods 1 & 2 and 3 & 4 mirrored

(Part of) SAT Encoding

- variable x_{ijpr} is true if team i plays team j at home in period p, round r
- $$\bigwedge_{i,p,r} \bigvee_{j \neq i} (x_{ijpr} \vee x_{jipr})$$ each team plays in every round

  $$\bigwedge_{i,p,r} \bigwedge_{j \neq i} \bigwedge_{k \neq i \wedge k \neq j} (x_{ijpr} \rightarrow \neg(x_{ikpr} \vee x_{kipr}))$$ each team plays at most once in every round

  $$\bigwedge_{i,j,r} (x_{ij1r} \rightarrow x_{ji2r}) \wedge (x_{ij3r} \rightarrow x_{ji4r})$$ mirror rounds 1& 2 and 3& 4

Result

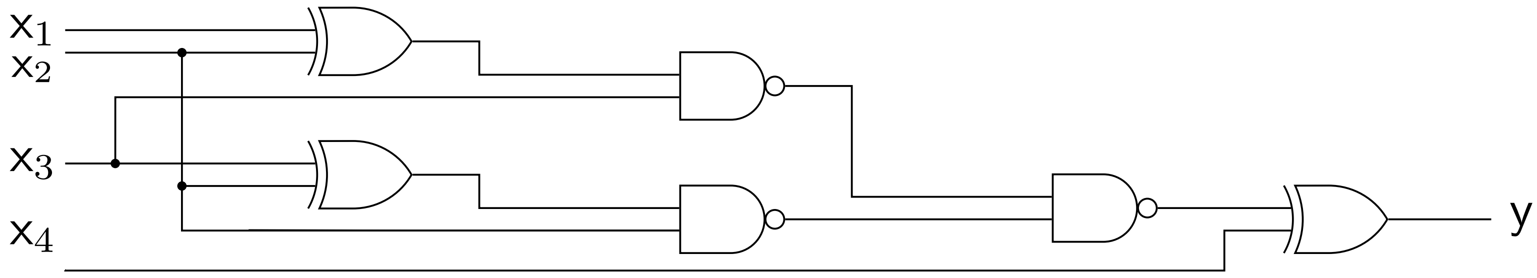
SAT scheduling is 100x faster than previous industrial scheduling tools

Application: Hardware Verification

Problem

- errors in hardware chips are costly (Intel paid $475 million for FDIV bug)

Example (Formal Circuit Model)

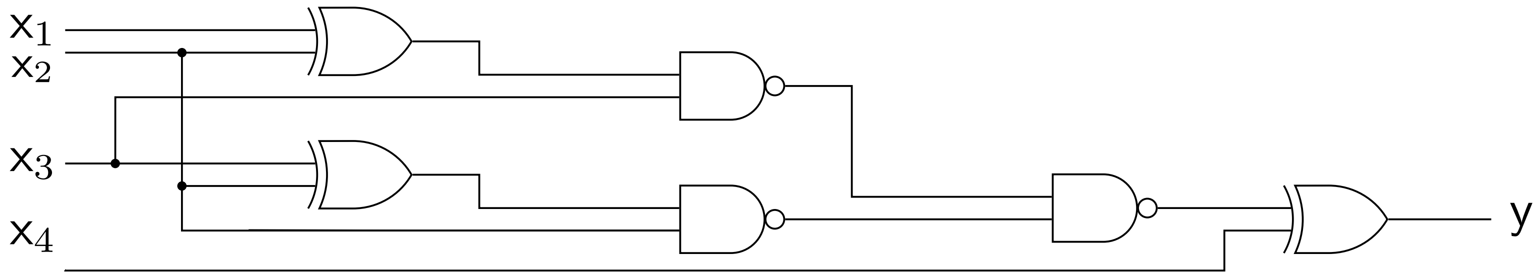


Application: Hardware Verification

Problem

- errors in hardware chips are costly (Intel paid $475 million for FDIV bug)
- testing is not enough to guarantee desired behavior

Example (Formal Circuit Model)

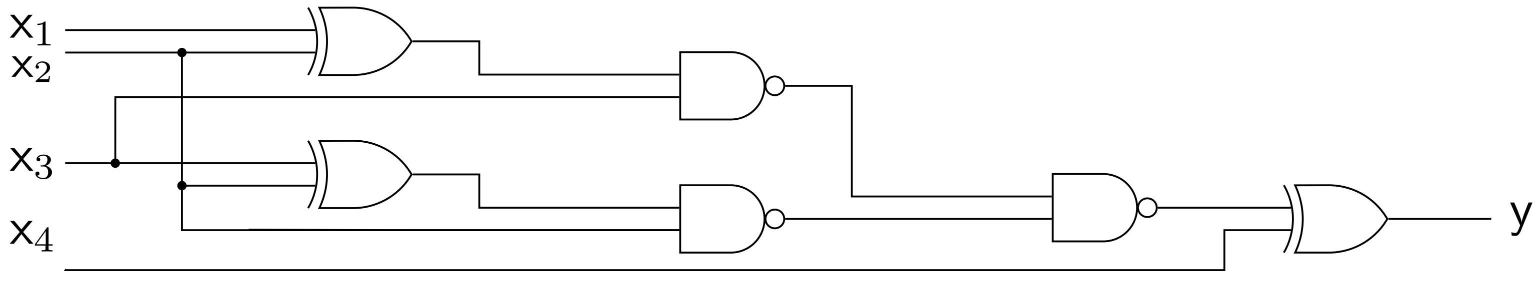


Application: Hardware Verification

Problem

- errors in hardware chips are costly (Intel paid $475 million for FDIV bug)
- testing is not enough to guarantee desired behavior

Example (Formal Circuit Model)



SAT Encoding

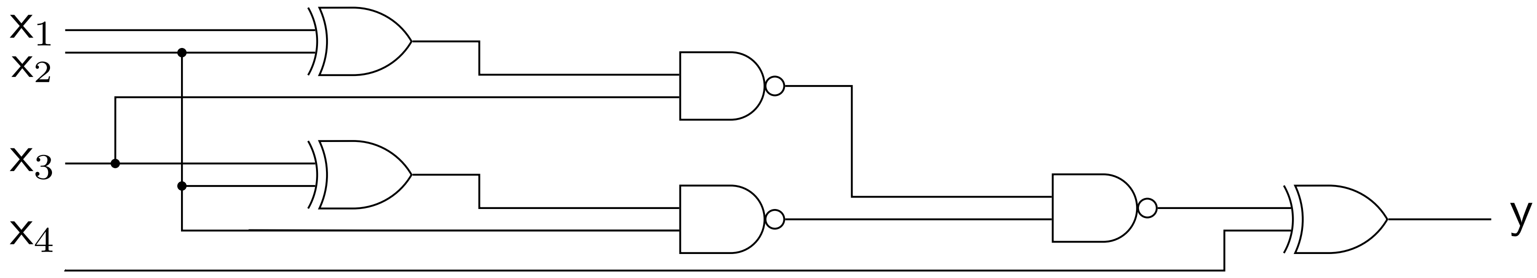
- variables for input and output
- SAT formulas for implemented behavior and expected behavior (specification)
- check for equivalence

Application: Hardware Verification

Problem

- errors in hardware chips are costly (Intel paid $475 million for FDIV bug)
- testing is not enough to guarantee desired behavior

Example (Formal Circuit Model)



SAT Encoding

- variables for input and output
- SAT formulas for implemented behavior and expected behavior (specification)
- check for equivalence

Impact

- ensured correctness, more reliable hardware components (formal verification)
- manufacturers rely on SAT-based verification since beginning of 2000s
  e.g., Intel Core i7 implements over 2700 distinct verified microinstructions

Propositional Logic Revisited

Concepts

- literal
- formula
- assignment
- satisfiability and validity
- negation normal form (NNF)
- conjunctive normal form (CNF)
- disjunctive normal form (DNF)

Definition (Propositional Logic: Syntax)

propositional formulas are built form

- atoms $p,\ q,\ r,\ p_1,\ p_2,\ \ldots$

Definition (Propositional Logic: Syntax)

propositional formulas are built form

- atoms $p,\ q,\ r,\ p_1,\ p_2,\ \ldots$
- constants $\bot,\ \top$

Definition (Propositional Logic: Syntax)

propositional formulas are built form

- atoms $p,\ q,\ r,\ p_1,\ p_2,\ \ldots$
- constants $\bot,\ \top$
- negation $\neg p$ “not p”

Definition (Propositional Logic: Syntax)

propositional formulas are built form

- atoms $p, q, r, p_1, p_2, \ldots$
- constants $\bot, \top$
- negation $\neg p$ "not p"
- conjunction $p \wedge q$ "p and q"

Definition (Propositional Logic: Syntax)

propositional formulas are built form

▶ atoms	$p, q, r, p_1, p_2, \ldots$	
▶ constants	$\bot, \top$	
▶ negation	$\neg p$	"not p"
▶ conjunction	$p \wedge q$	"p and q"
▶ disjunction	$p \vee q$	"p or q"

Definition (Propositional Logic: Syntax)

propositional formulas are built form

▶ atoms	$p,\ q,\ r,\ p_1,\ p_2,\ \ldots$	
▶ constants	$\bot,\ \top$	
▶ negation	$\neg p$	"not p"
▶ conjunction	$p \wedge q$	"p and q"
▶ disjunction	$p \vee q$	"p or q"
▶ implication	$p \rightarrow q$	"if p then q holds"

Definition (Propositional Logic: Syntax)

propositional formulas are built form

▶ atoms	$p,\ q,\ r,\ p_1,\ p_2,\ \ldots$	
▶ constants	$\bot,\ \top$	
▶ negation	$\neg p$	"not p"
▶ conjunction	$p \land q$	"p and q"
▶ disjunction	$p \lor q$	"p or q"
▶ implication	$p \to q$	"if p then q holds"
▶ equivalence	$p \leftrightarrow q$	"p if and only if q"

Definition (Propositional Logic: Syntax)

propositional formulas are built form

▶ atoms	$p,\ q,\ r,\ p_1,\ p_2,\ \ldots$	
▶ constants	$\bot,\ \top$	
▶ negation	$\neg p$	"not p"
▶ conjunction	$p \wedge q$	"p and q"
▶ disjunction	$p \vee q$	"p or q"
▶ implication	$p \rightarrow q$	"if p then q holds"
▶ equivalence	$p \leftrightarrow q$	"p if and only if q"

according to the BNF grammar

$$\varphi ::= p \mid \bot \mid \top \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Definition (Propositional Logic: Syntax)

propositional formulas are built form

- atoms $p, \ q, \ r, \ p_1, \ p_2, \ \ldots$
- constants $\bot, \ \top$
- negation $\neg p$ "not p"
- conjunction $p \wedge q$ "p and q"
- disjunction $p \vee q$ "p or q"
- implication $p \to q$ "if p then q holds"
- equivalence $p \leftrightarrow q$ "p if and only if q"

according to the BNF grammar

$$\varphi ::= p \mid \bot \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Conventions

- binding precedence $\neg \quad > \quad \wedge \quad > \quad \vee \quad > \quad \to, \leftrightarrow$

Definition (Propositional Logic: Syntax)

propositional formulas are built form

▶ atoms	$p,\ q,\ r,\ p_1,\ p_2,\ \ldots$	
▶ constants	$\bot,\ \top$	
▶ negation	$\neg p$	"not p"
▶ conjunction	$p \wedge q$	"p and q"
▶ disjunction	$p \vee q$	"p or q"
▶ implication	$p \rightarrow q$	"if p then q holds"
▶ equivalence	$p \leftrightarrow q$	"p if and only if q"

according to the BNF grammar

$$\varphi ::= p \mid \bot \mid \top \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Conventions

- binding precedence $\quad \neg \quad > \quad \wedge \quad > \quad \vee \quad > \quad \rightarrow, \leftrightarrow$
- omit outer parantheses

Definition (Propositional Logic: Syntax)

propositional formulas are built form

- atoms $p,\ q,\ r,\ p_1,\ p_2,\ \ldots$
- constants $\bot,\ \top$
- negation $\neg p$ “not p”
- conjunction $p \wedge q$ “p and q”
- disjunction $p \vee q$ “p or q”
- implication $p \rightarrow q$ “if p then q holds”
- equivalence $p \leftrightarrow q$ “p if and only if q”

according to the BNF grammar

$$\varphi ::= p \mid \bot \mid \top \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Conventions

- binding precedence $\neg \quad > \quad \wedge \quad > \quad \vee \quad > \quad \rightarrow, \leftrightarrow$
- omit outer parantheses
- $\rightarrow$, $\wedge$, $\vee$ are right-associative: $p \rightarrow q \rightarrow r$ denotes $p \rightarrow (q \rightarrow r)$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \ldots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \ldots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F}$$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \ldots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F} \qquad\qquad v(\top) = \mathsf{T}$$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \dots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F} \qquad\qquad v(\top) = \mathsf{T}$$

$$v(\varphi \land \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \dots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F} \qquad v(\top) = \mathsf{T}$$

$$v(\varphi \wedge \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases} \qquad v(\neg\varphi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = \mathsf{F} \\ \mathsf{F} & \text{if } v(\varphi) = \mathsf{T} \end{cases}$$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \dots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F} \qquad\qquad v(\top) = \mathsf{T}$$

$$v(\varphi \wedge \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases} \qquad\qquad v(\neg\varphi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = \mathsf{F} \\ \mathsf{F} & \text{if } v(\varphi) = \mathsf{T} \end{cases}$$

$$v(\varphi \vee \psi) = \begin{cases} \mathsf{F} & \text{if } v(\varphi) = v(\psi) = \mathsf{F} \\ \mathsf{T} & \text{otherwise} \end{cases}$$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \ldots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F} \qquad v(\top) = \mathsf{T}$$

$$v(\varphi \wedge \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases} \qquad v(\neg\varphi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = \mathsf{F} \\ \mathsf{F} & \text{if } v(\varphi) = \mathsf{T} \end{cases}$$

$$v(\varphi \vee \psi) = \begin{cases} \mathsf{F} & \text{if } v(\varphi) = v(\psi) = \mathsf{F} \\ \mathsf{T} & \text{otherwise} \end{cases} \qquad v(\varphi \leftrightarrow \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Definition (Propositional Logic: Semantics)

- valuation (truth assignment) is mapping $v : \{p, q, r, \dots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

$$v(\bot) = \mathsf{F} \qquad\qquad v(\top) = \mathsf{T}$$

$$v(\varphi \wedge \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases} \qquad v(\neg\varphi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = \mathsf{F} \\ \mathsf{F} & \text{if } v(\varphi) = \mathsf{T} \end{cases}$$

$$v(\varphi \vee \psi) = \begin{cases} \mathsf{F} & \text{if } v(\varphi) = v(\psi) = \mathsf{F} \\ \mathsf{T} & \text{otherwise} \end{cases} \qquad v(\varphi \leftrightarrow \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) \\ \mathsf{F} & \text{otherwise} \end{cases}$$

$$v(\varphi \to \psi) = \begin{cases} \mathsf{F} & \text{if } v(\varphi) = \mathsf{T},\ v(\psi) = \mathsf{F} \\ \mathsf{T} & \text{otherwise} \end{cases}$$

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathrm{T}$ for some valuation v

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathsf{T}$ for some valuation v
- formula φ is valid if $v(\varphi) = \mathsf{T}$ for every valuation v

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathrm{T}$ for some valuation v
- formula φ is valid if $v(\varphi) = \mathrm{T}$ for every valuation v
- semantic entailment $\varphi_1, \ldots, \varphi_n \vDash \psi$
  if $v(\psi) = \mathrm{T}$ whenever $v(\varphi_1) = v(\varphi_2) = \cdots = v(\varphi_n) = \mathrm{T}$

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathrm{T}$ for some valuation v
- formula φ is valid if $v(\varphi) = \mathrm{T}$ for every valuation v
- semantic entailment $\varphi_1, \ldots, \varphi_n \vDash \psi$
  if $v(\psi) = \mathrm{T}$ whenever $v(\varphi_1) = v(\varphi_2) = \cdots = v(\varphi_n) = \mathrm{T}$
- formulas φ and ψ are equivalent ($\varphi \equiv \psi$) if $v(\varphi) = v(\psi)$ for every valuation v

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathrm{T}$ for some valuation v
- formula φ is valid if $v(\varphi) = \mathrm{T}$ for every valuation v
- semantic entailment $\varphi_1, \dots, \varphi_n \vDash \psi$
  if $v(\psi) = \mathrm{T}$ whenever $v(\varphi_1) = v(\varphi_2) = \cdots = v(\varphi_n) = \mathrm{T}$
- formulas φ and ψ are equivalent ($\varphi \equiv \psi$) if $v(\varphi) = v(\psi)$ for every valuation v
- formulas φ and ψ are equisatisfiable ($\varphi \approx \psi$) if

$$\varphi \text{ is satisfiable} \iff \psi \text{ is satisfiable}$$

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathrm{T}$ for some valuation v
- formula φ is valid if $v(\varphi) = \mathrm{T}$ for every valuation v
- semantic entailment $\varphi_1, \dots, \varphi_n \vDash \psi$
  if $v(\psi) = \mathrm{T}$ whenever $v(\varphi_1) = v(\varphi_2) = \cdots = v(\varphi_n) = \mathrm{T}$
- formulas φ and ψ are equivalent ($\varphi \equiv \psi$) if $v(\varphi) = v(\psi)$ for every valuation v
- formulas φ and ψ are equisatisfiable ($\varphi \approx \psi$) if

$$\varphi \text{ is satisfiable} \iff \psi \text{ is satisfiable}$$

Theorem

formula φ is unsatisfiable if and only if $\neg\varphi$ is valid

Definitions

- formula φ is satisfiable if $v(\varphi) = \mathsf{T}$ for some valuation v
- formula φ is valid if $v(\varphi) = \mathsf{T}$ for every valuation v
- semantic entailment $\varphi_1, \dots, \varphi_n \vDash \psi$
  if $v(\psi) = \mathsf{T}$ whenever $v(\varphi_1) = v(\varphi_2) = \cdots = v(\varphi_n) = \mathsf{T}$
- formulas φ and ψ are equivalent ($\varphi \equiv \psi$) if $v(\varphi) = v(\psi)$ for every valuation v
- formulas φ and ψ are equisatisfiable ($\varphi \approx \psi$) if

$$\varphi \text{ is satisfiable} \iff \psi \text{ is satisfiable}$$

Theorem

formula φ is unsatisfiable if and only if $\neg\varphi$ is valid

Theorem

satisfiability and validity are decidable

Definition (Literal)

- literal is atom p or negation of atom $\neg p$

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms
- conjunctive normal form (CNF) is conjunction of disjunctions

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms
- conjunctive normal form (CNF) is conjunction of disjunctions
- 3-CNF is conjunction of disjunctions with 3 literals: $\bigwedge_i (a_i \vee b_i \vee c_i)$

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms
- conjunctive normal form (CNF) is conjunction of disjunctions
- 3-CNF is conjunction of disjunctions with 3 literals: $\bigwedge_i (a_i \vee b_i \vee c_i)$
- disjunctive normal form (DNF) is disjunction of conjunctions

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms
- conjunctive normal form (CNF) is conjunction of disjunctions
- 3-CNF is conjunction of disjunctions with 3 literals: $\bigwedge_i (a_i \vee b_i \vee c_i)$
- disjunctive normal form (DNF) is disjunction of conjunctions

Theorem

for every formula φ there is CNF ψ, 3-CNF χ and DNF η such that $\varphi \equiv \psi \equiv \chi \equiv \eta$

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms
- conjunctive normal form (CNF) is conjunction of disjunctions
- 3-CNF is conjunction of disjunctions with 3 literals: $\bigwedge_i (a_i \vee b_i \vee c_i)$
- disjunctive normal form (DNF) is disjunction of conjunctions

Theorem

for every formula φ there is CNF ψ, 3-CNF χ and DNF η such that $\varphi \equiv \psi \equiv \chi \equiv \eta$

Remarks

- translation from formula to CNF can result in exponential blowup

Definition (Literal)

- literal is atom p or negation of atom $\neg p$
- literals l_1 and l_2 are complementary if $l_1 = \neg l_2$ or $l_2 = \neg l_1$

Definitions

- negation normal form (NNF) if formula with negation only applied to atoms
- conjunctive normal form (CNF) is conjunction of disjunctions
- 3-CNF is conjunction of disjunctions with 3 literals: $\bigwedge_i (a_i \vee b_i \vee c_i)$
- disjunctive normal form (DNF) is disjunction of conjunctions

Theorem

for every formula φ there is CNF ψ, 3-CNF χ and DNF η such that $\varphi \equiv \psi \equiv \chi \equiv \eta$

Remarks

- translation from formula to CNF can result in exponential blowup
- Tseitin's transformation is linear and produces equisatisfiable formula

Satisfiability (SAT)

instance:	propositional formula φ
question:	is φ satisfiable?

Satisfiability (SAT)

instance:	propositional formula φ
question:	is φ satisfiable?

3-Satisfiability (3-SAT)

instance:	propositional formula φ in 3-CNF
question:	is φ satisfiable?

Satisfiability (SAT)

instance:	propositional formula φ
question:	is φ satisfiable?

3-Satisfiability (3-SAT)

instance:	propositional formula φ in 3-CNF
question:	is φ satisfiable?

Theorem

SAT and 3-SAT are NP-complete problems

Satisfiability (SAT)

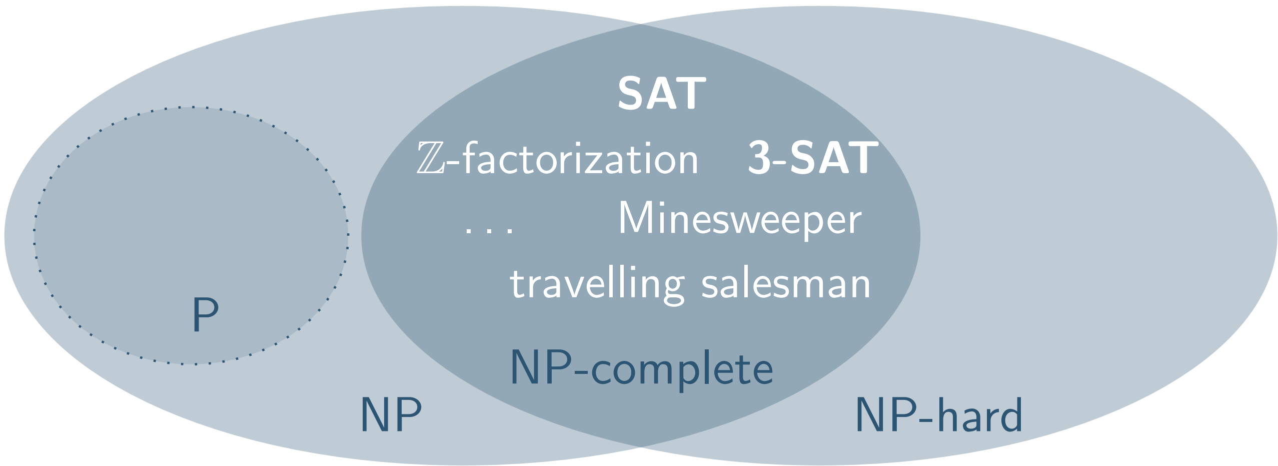
instance: propositional formula φ
question: is φ satisfiable?

3-Satisfiability (3-SAT)

instance: propositional formula φ in 3-CNF
question: is φ satisfiable?

Theorem

SAT and 3-SAT are NP-complete problems



- 1 million \$ prize money awarded for solution to $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

Fact

most SAT solvers require input to be in CNF

Fact

most SAT solvers require input to be in CNF

Remarks

- transforming formula to equivalent CNF can cause exponential blowup
- transforming formula into equisatisfiable CNF is possible in linear time

Fact

most SAT solvers require input to be in CNF

Remarks

- transforming formula to equivalent CNF can cause exponential blowup
- transforming formula into equisatisfiable CNF is possible in linear time

Definition

formulas φ and ψ are equisatisfiable ($\varphi \approx \psi$) if

$$\varphi \text{ is satisfiable} \qquad \iff \qquad \psi \text{ is satisfiable}$$

Fact

most SAT solvers require input to be in CNF

Remarks

- transforming formula to equivalent CNF can cause exponential blowup
- transforming formula into equisatisfiable CNF is possible in linear time

Definition

formulas φ and ψ are equisatisfiable ($\varphi \approx \psi$) if

$$\varphi \text{ is satisfiable} \qquad \Longleftrightarrow \qquad \psi \text{ is satisfiable}$$

Example

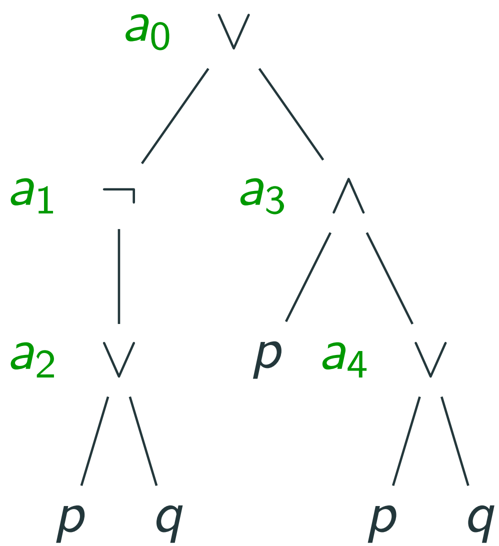
$$p \vee q \;\approx\; \top \qquad\qquad p \wedge \neg p \;\approx\; q \wedge \neg q \qquad\qquad p \wedge \neg p \;\not\approx\; p \wedge \neg q$$

Example (Tseitin's Transformation)

- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$

Example (Tseitin's Transformation)

- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$



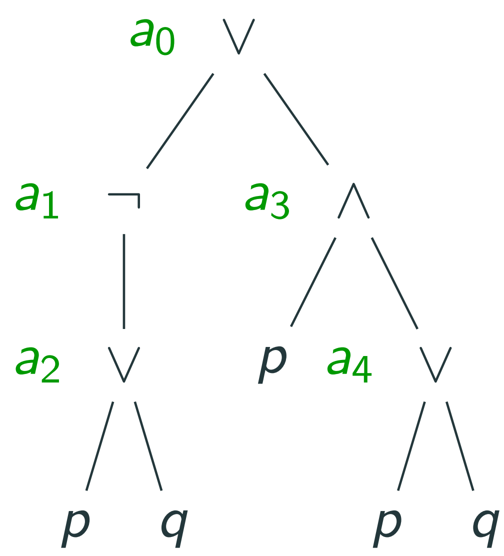
Example (Tseitin's Transformation)

- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$
- use fresh propositional variable for every connective

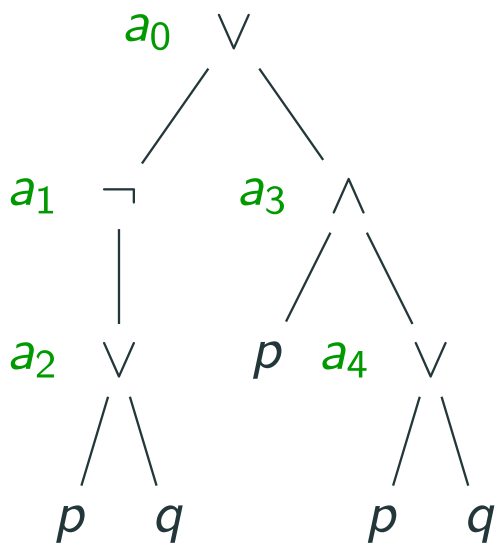
$a_0 : \neg(p \vee q) \vee (p \wedge (p \vee q))$ $\quad a_1 : \neg(p \vee q)$

$a_2 : p \vee q$ $\quad a_3 : p \wedge (p \vee q)$

$a_4 : p \vee q$



Example (Tseitin's Transformation)



- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$
- use fresh propositional variable for every connective

  $a_0 \colon \neg(p \vee q) \vee (p \wedge (p \vee q))$ $\quad a_1 \colon \neg(p \vee q)$

  $a_2 \colon p \vee q$ $\quad a_3 \colon p \wedge (p \vee q)$

  $a_4 \colon p \vee q$

- $\varphi \approx \quad a_0 \wedge (a_0 \leftrightarrow a_1 \vee a_3) \wedge (a_1 \leftrightarrow \neg a_2) \wedge (a_2 \leftrightarrow p \vee q) \wedge$
  $(a_3 \leftrightarrow p \wedge a_4) \wedge (a_4 \leftrightarrow p \vee q)$

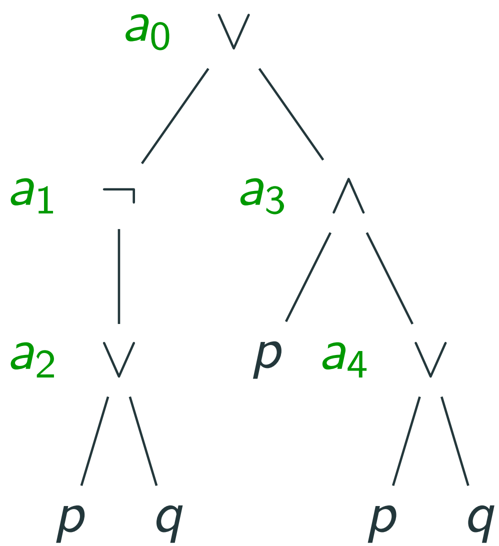
Example (Tseitin's Transformation)

- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$
- use fresh propositional variable for every connective

  $a_0 \colon \neg(p \vee q) \vee (p \wedge (p \vee q)) \quad a_1 \colon \neg(p \vee q)$

  $a_2 \colon p \vee q \quad a_3 \colon p \wedge (p \vee q)$

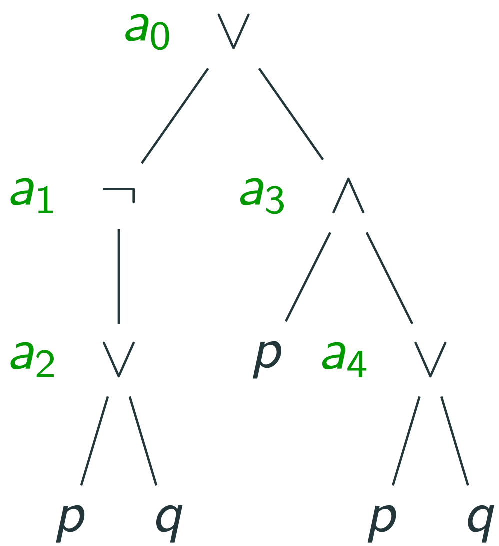
  $a_4 \colon p \vee q$



- $\varphi \approx \quad a_0 \wedge (a_0 \leftrightarrow a_1 \vee a_3) \wedge (a_1 \leftrightarrow \neg a_2) \wedge (a_2 \leftrightarrow p \vee q) \wedge (a_3 \leftrightarrow p \wedge a_4) \wedge (a_4 \leftrightarrow p \vee q)$
- every $\leftrightarrow$ subexpression can be replaced by at most three clauses:

$$
\begin{aligned}
a \leftrightarrow b \wedge c &\quad\equiv\quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c) \\
a \leftrightarrow b \vee c &\quad\equiv\quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c) \\
a \leftrightarrow \neg b &\quad\equiv\quad (\neg a \vee \neg b) \wedge (a \vee b)
\end{aligned}
$$

Example (Tseitin's Transformation)



- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$
- use fresh propositional variable for every connective

  $a_0 : \neg(p \vee q) \vee (p \wedge (p \vee q))$ $\quad$ $a_1 : \neg(p \vee q)$

  $a_2 : p \vee q$ $\quad$ $a_3 : p \wedge (p \vee q)$

  $a_4 : p \vee q$

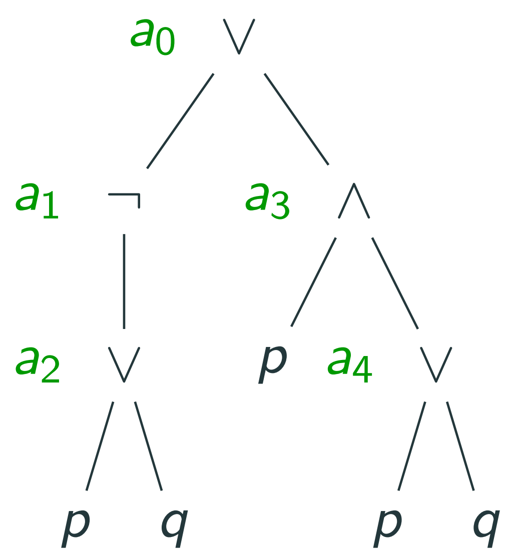
- $\varphi \approx \quad a_0 \wedge (a_0 \leftrightarrow a_1 \vee a_3) \wedge (a_1 \leftrightarrow \neg a_2) \wedge (a_2 \leftrightarrow p \vee q) \wedge (a_3 \leftrightarrow p \wedge a_4) \wedge (a_4 \leftrightarrow p \vee q)$

- every $\leftrightarrow$ subexpression can be replaced by at most three clauses:

$$
\begin{aligned}
a \leftrightarrow b \wedge c &\quad\equiv\quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c) \\
a \leftrightarrow b \vee c &\quad\equiv\quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c) \\
a \leftrightarrow \neg b &\quad\equiv\quad (\neg a \vee \neg b) \wedge (a \vee b)
\end{aligned}
$$

- common subexpressions can be shared

Example (Tseitin's Transformation)



- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$
- use fresh propositional variable for every connective

  $a_0 : \neg(p \vee q) \vee (p \wedge (p \vee q))$ $\quad a_1 : \neg(p \vee q)$

  $a_2 : p \vee q$ $\quad a_3 : p \wedge (p \vee q)$

  $a_4 : p \vee q$

- $\varphi \approx \quad a_0 \wedge (a_0 \leftrightarrow a_1 \vee a_3) \wedge (a_1 \leftrightarrow \neg a_2) \wedge (a_2 \leftrightarrow p \vee q) \wedge (a_3 \leftrightarrow p \wedge a_2)$
- every $\leftrightarrow$ subexpression can be replaced by at most three clauses:

$$
\begin{aligned}
a \leftrightarrow b \wedge c &\quad\equiv\quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c) \\
a \leftrightarrow b \vee c &\quad\equiv\quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c) \\
a \leftrightarrow \neg b &\quad\equiv\quad (\neg a \vee \neg b) \wedge (a \vee b)
\end{aligned}
$$

- common subexpressions can be shared

SAT Solvers

Minisat, Glucose, CaDiCaL, Glu_VC, Plingeling, MapleLRB LCM, MapleCOMPSPS, Riss, Lingeling, Treengeling, CryptoMiniSat, abcdSAT, Dimetheus, Kiel, MapleCOMSPS, Rsat, SWDiA5BY, BlackBox, SWDiA5BY, pprobSAT, glueSplit_clasp, BalancedZ, SApperloT, PeneLoPe, MXC, ROKKminisat, MiniSat_HACK_999ED, ZENN, CSHCrandMC, MiniGolf, march_rw, sattime2011, mphasesat64, sparrow2011, pmcSAT, CSHCpar8, gluebit_clasp, clasp, precosat, gNovelty, SATzilla, SatELite, Score2SAT, YalSAT, tch glucose3, . . .

SAT Solvers

Minisat, Glucose, CaDiCaL, Glu_VC, Plingeling, MapleLRB LCM, MapleCOMPSPS, Riss, Lingeling, Treengeling, CryptoMiniSat, abcdSAT, Dimetheus, Kiel, MapleCOMSPS, Rsat, SWDiA5BY, BlackBox, SWDiA5BY, pprobSAT, glueSplit_clasp, BalancedZ, SApperloT, PeneLoPe, MXC, ROKKminisat, MiniSat_HACK_999ED, ZENN, CSHCrandMC, MiniGolf, march_rw, sattime2011, mphasesat64, sparrow2011, pmcSAT, CSHCpar8, gluebit_clasp, clasp, precosat, gNovelty, SATzilla, SatELite, Score2SAT, YalSAT, tch glucose3, ...

SAT Competition

- annual competition for different tracks (main, parallel, no-limit, ...)
- increasing set of benchmarks from industry, mathematics, cryptography, ...
- standardized input format DIMACS and proof format DRAT

SAT Solvers

Minisat, Glucose, CaDiCaL, Glu_VC, Plingeling, MapleLRB LCM, MapleCOMPSPS, Riss, Lingeling, Treengeling, CryptoMiniSat, abcdSAT, Dimetheus, Kiel, MapleCOMSPS, Rsat, SWDiA5BY, BlackBox, SWDiA5BY, pprobSAT, glueSplit_clasp, BalancedZ, SApperloT, PeneLoPe, MXC, ROKKminisat, MiniSat_HACK_999ED, ZENN, CSHCrandMC, MiniGolf, march_rw, sattime2011, mphasesat64, sparrow2011, pmcSAT, CSHCpar8, gluebit_clasp, clasp, precosat, gNovelty, SATzilla, SatELite, Score2SAT, YalSAT, tch glucose3, ...

SAT Competition

- annual competition for different tracks (main, parallel, no-limit, ...)
- increasing set of benchmarks from industry, mathematics, cryptography, ...
- standardized input format DIMACS and proof format DRAT

http://www.satcompetition.org/

SAT Solvers

Minisat, Glucose, CaDiCaL, Glu_VC, Plingeling, MapleLRB LCM, MapleCOMPSPS, Riss, Lingeling, Treengeling, CryptoMiniSat, abcdSAT, Dimetheus, Kiel, MapleCOMSPS, Rsat, SWDiA5BY, BlackBox, SWDiA5BY, pprobSAT, glueSplit_clasp, BalancedZ, SApperloT, PeneLoPe, MXC, ROKKminisat, MiniSat_HACK_999ED, ZENN, CSHCrandMC, MiniGolf, march_rw, sattime2011, mphasesat64, sparrow2011, pmcSAT, CSHCpar8, gluebit_clasp, clasp, precosat, gNovelty, SATzilla, SatELite, Score2SAT, YalSAT, tch glucose3, . . .

SAT Competition

- annual competition for different tracks (main, parallel, no-limit, . . .)
- increasing set of benchmarks from industry, mathematics, cryptography, . . .
- standardized input format DIMACS and proof format DRAT

`http://www.satcompetition.org/`

Minisat

- minimalistic open source solver (`http://minisat.se/` or apt, yum,. . .)

```
$ minisat test.sat result.txt
```

- web interface

Example (DIMACS)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_4)$ can be expressed by

```
c a very simple example
p cnf 4 3
1 -3 0
2 3 -1 0
-1 2 4 0
```

Example (DIMACS)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_4)$ can be expressed by

```
c a very simple example
p cnf 4 3
1 -3 0
2 3 -1 0
-1 2 4 0
```

The DIMACS Format

- header `p cnf` n m specifies number of variables n and number of clauses m

Example (DIMACS)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_4)$ can be expressed by

```
c a very simple example
p cnf 4 3
1 -3 0
2 3 -1 0
-1 2 4 0
```

The DIMACS Format

- header `p cnf` n m specifies number of variables n and number of clauses m
- variables (atoms) are assumed to be $x_1, \ldots, x_n$

Example (DIMACS)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_4)$ can be expressed by

```
c a very simple example
p cnf 4 3
1 -3 0
2 3 -1 0
-1 2 4 0
```

The DIMACS Format

- header `p cnf` n m specifies number of variables n and number of clauses m
- variables (atoms) are assumed to be $x_1, \dots, x_n$
- literal x_i is denoted `i` and literal $\neg x_i$ is denoted `-i`

Example (DIMACS)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_4)$ can be expressed by

```
c a very simple example
p cnf 4 3
1 -3 0
2 3 -1 0
-1 2 4 0
```

The DIMACS Format

- header `p cnf` n m specifies number of variables n and number of clauses m
- variables (atoms) are assumed to be $x_1, \ldots, x_n$
- literal x_i is denoted `i` and literal $\neg x_i$ is denoted `-i`
- a clause is a list of literals terminated by `0`

Example (DIMACS)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_4)$ can be expressed by

```
c a very simple example
p cnf 4 3
1 -3 0
2 3 -1 0
-1 2 4 0
```

The DIMACS Format

- header `p cnf` n m specifies number of variables n and number of clauses m
- variables (atoms) are assumed to be $x_1, \ldots, x_n$
- literal x_i is denoted `i` and literal $\neg x_i$ is denoted `-i`
- a clause is a list of literals terminated by `0`
- lines starting with `c` are considered comments