universität innsbruck

SAT and SMT Solving

Sarah Winkler

Computational Logic Group
Department of Computer Science
University of Innsbruck

lecture 1
SS 2019

Outline

- Introduction
  - Organisation
  - Why SAT and SMT?
  - Contents
- Propositional Logic
- DPLL
- Transformations to CNF
- Using SAT Solvers

Important Information

- LVA 703048 (PS 2)

Important Information

- LVA 703048 (PS 2)
- http://cl-informatik.uibk.ac.at/teaching/ss19/satsmt/

Important Information

- LVA 703048 (PS 2)
- http://cl-informatik.uibk.ac.at/teaching/ss19/satsmt/

Time and Place

PS	Friday	13:15 – 15:00	HSB9

Important Information

- LVA 703048 (PS 2)
- http://cl-informatik.uibk.ac.at/teaching/ss19/satsmt/

Time and Place

VO	Friday	13:15 – 14:00	HSB9
PS	Friday	14:15 – 15:00	HSB9

Important Information

- LVA 703048 (PS 2)
- `http://cl-informatik.uibk.ac.at/teaching/ss19/satsmt/`

Time and Place

VO	Friday	13:15 – 14:00	HSB9
PS	Friday	14:15 – 15:00	HSB9

Grading

- 70% weekly exercises, 30% proseminar test on June 28
- attendence required

Exercises

- 10 points per week
- indicate solved exercises before Friday 11:00 in OLAT, submit solutions

Important Information

- LVA 703048 (PS 2)
- `http://cl-informatik.uibk.ac.at/teaching/ss19/satsmt/`

Time and Place

VO	Friday	13:15 – 14:00	HSB9
PS	Friday	14:15 – 15:00	HSB9

Grading

- 70% weekly exercises, 30% proseminar test on June 28
- attendence required

Exercises

- 10 points per week
- indicate solved exercises before Friday 11:00 in OLAT, submit solutions

Important Information

- LVA 703048 (PS 2)
- http://cl-informatik.uibk.ac.at/teaching/ss19/satsmt/

Time and Place

VO	Friday	13:15 – 14:00	HSB9
PS	Friday	14:15 – 15:00	HSB9

Grading

- 70% weekly exercises, 30% proseminar test on June 28
- attendence required

Exercises

- 10 points per week
- indicate solved exercises before Friday 11:00 in OLAT, submit solutions

Consultation Hours

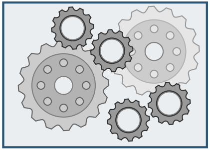
Sarah Winkler	3M03	Thursday 14:00 – 16:00

Outline

- Introduction
  - Organisation
  - Why SAT and SMT?
  - Contents
- Propositional Logic
- DPLL
- Transformations to CNF
- Using SAT Solvers

SAT Solving

input: propositional formula φ

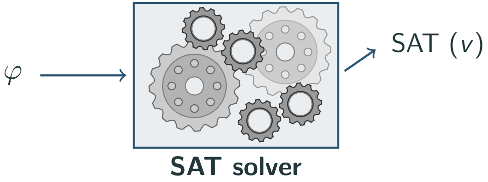
φ ⟶ 

SAT solver

SAT Solving

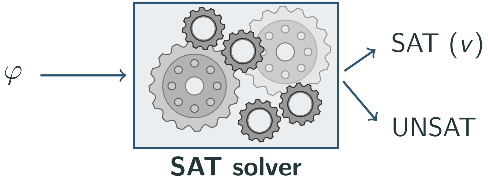
input: propositional formula φ

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

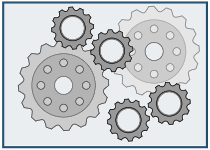


SAT Solving

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

φ $\longrightarrow$

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



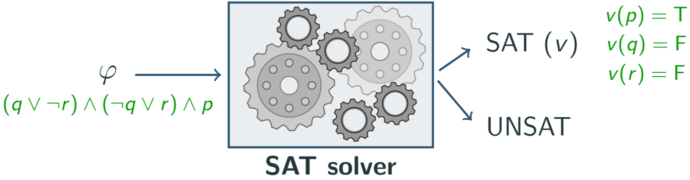
SAT solver

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

UNSAT

SAT Solving

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

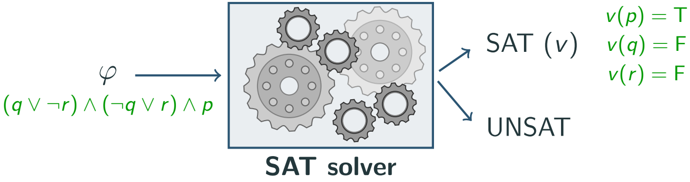


Terminology

- 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



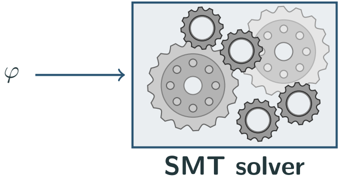
Terminology

- 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

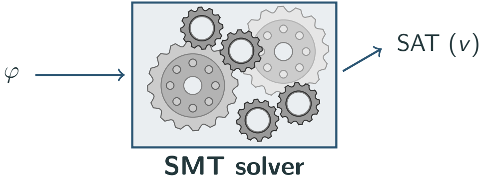
input: formula φ involving theory T



SMT Solving

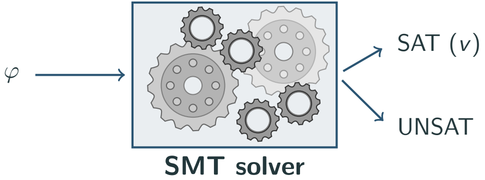
input: formula φ involving theory T

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

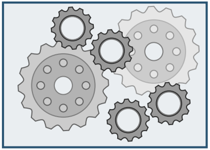
input: formula φ involving theory T

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

UNSAT

SMT Solving

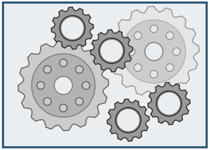
input: formula φ involving theory T

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

→ UNSAT

Example (Theories)

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

SMT Solving

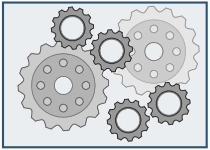
input: formula φ involving theory T

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

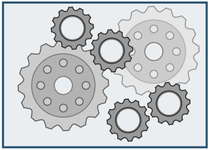
input: formula φ involving theory T

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

UNSAT otherwise

φ →

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



SMT solver

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

→ 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 $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$

SMT Solving

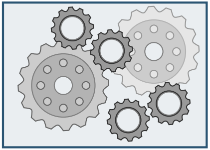
input: formula φ involving theory T

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

  answer to P is yes $\iff$ φ_P is satisfiable

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

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 $v(q_i) = \mathsf{T}$ if question i is included, $v(q_i) = \mathsf{F}$ 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

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

Result

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

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.

Problem

Can one co

$x^2 + y^2 =$

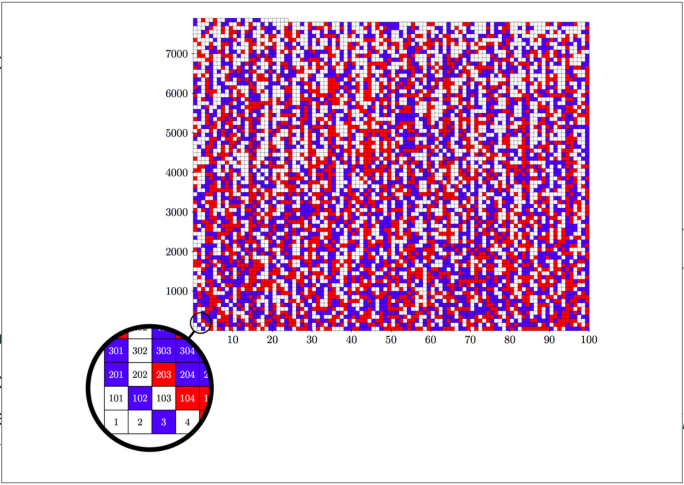
Example

(a) 1 … ✓

(b) 1 … ✗

SAT Enc

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

Der längste Mathe-Bew ×

www.spiegel.de/wissenschaft/mensch/

Search

Most Visited Getting Started OLAT

SPIEGEL ONLINE DER SPIEGEL SPIEGEL TV Anmelden

Menü | Politik Meinung Wirtschaft Panorama Sport Kultur Netzwelt Wissenschaft mehr▾

WISSENSCHAFT

Schlagzeilen | Wetter | DAX 12.200,59 | TV-Programm | Abo

Nachrichten › Wissenschaft › Mensch › Mathematik › Der längste Mathe-Beweis der Welt umfasst 200 Terabyte

Zahlenrätsel

Der längste Mathe-Beweis der Welt

Drei Mathematiker haben ein Zahlenrätsel geknackt - mithilfe eines Supercomputers. Der Beweis umfasst 200 Terabyte. Sie wollen wissen, worum es geht? Okay, versuchen wir es.

Von Holger Dambeck

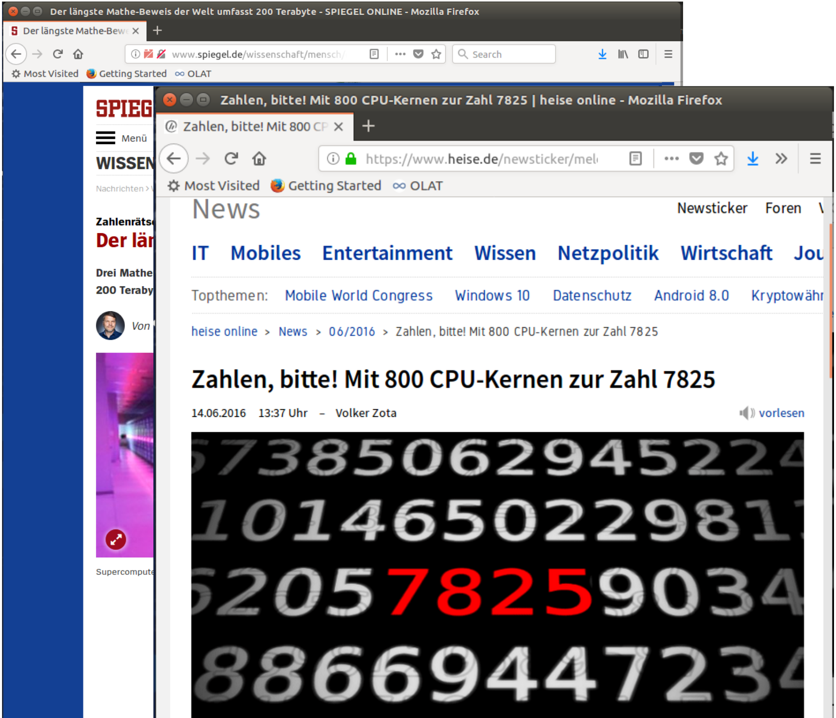
DPA

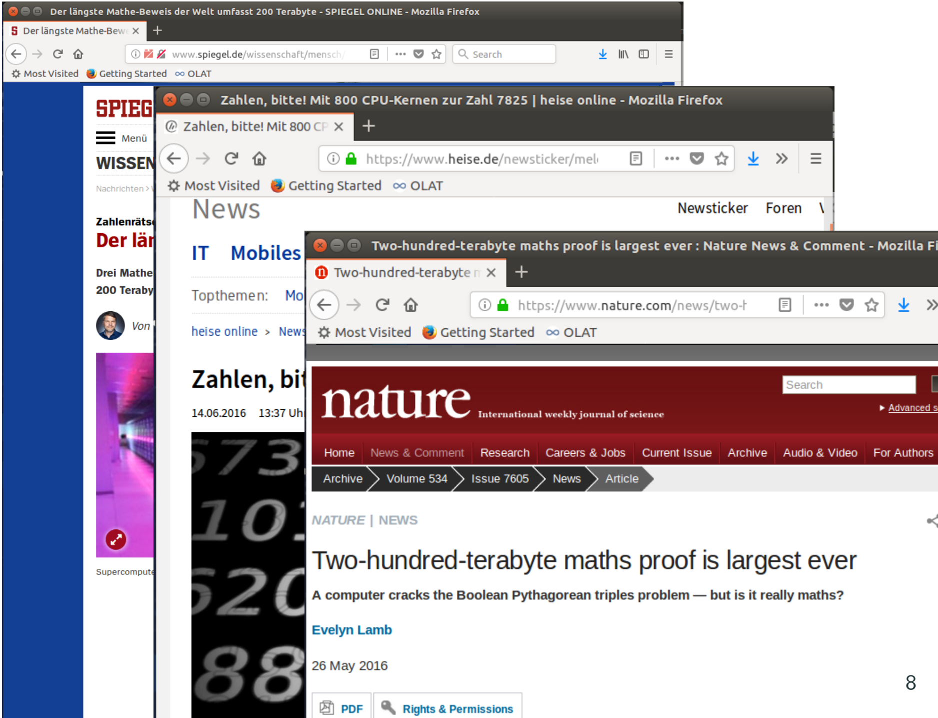
Supercomputer als Mathematiker

Teilen Twittern E-Mail

Montag, **30.05.2016** 18:41 Uhr Drucken Nutzungsrechte Feedback Kommentieren

Das Preisgeld war nur symbolisch: 100 Dollar hatte der US-Mathematiker Ronald Graham in den Achtzigerjahren demjenigen versprochen, der ein bis dahin ungelöstes Zahlenrätsel entschlüsselt. Es ging dabei um sogenannte


Der längste Mathe-Beweis der Welt umfasst 200 Terabyte - SPIEGEL ONLINE - Mozilla Firefox
Der längste Mathe-Bew
www.spiegel.de/wissenschaft/mensch/
Search
Most Visited
Getting Started
OLAT
SPIEG
Menü
WISSEN
Nachrichten
Zahlenräts
Der lär
Drei Mathe
200 Teraby
Von
Supercompute
Zahlen, bitte! Mit 800 CPU-Kernen zur Zahl 7825 | heise online - Mozilla Firefox
Zahlen, bitte! Mit 800 CP
https://www.heise.de/newsticker/mel
Most Visited
Getting Started
OLAT
News
Newsticker
Foren
IT
Mobiles
Entertainment
Wissen
Netzpolitik
Wirtschaft
Jou
Topthemen:
Mobile World Congress
Windows 10
Datenschutz
Android 8.0
Kryptowähr
heise online > News > 06/2016 > Zahlen, bitte! Mit 800 CPU-Kernen zur Zahl 7825
Zahlen, bitte! Mit 800 CPU-Kernen zur Zahl 7825
14.06.2016 13:37 Uhr – Volker Zota
vorlesen
7385062945222
1014650229811
2057825903
886694472 23


Der längste Mathe-Beweis der Welt umfasst 200 Terabyte - SPIEGEL ONLINE - Mozilla Firefox
www.spiegel.de/wissenschaft/mensch/
Most Visited
Getting Started
OLAT
Menü
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
14.06.2016 13:37 Uhr
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
Search
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, ...)

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

  $$\bigwedge_{i,p,r} \bigwedge_{j \neq i} \bigwedge_{k \neq i \wedge k \neq j} (x_{ijpr} \rightarrow \neg(x_{ikpr} \vee x_{kipr}))$$ 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

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

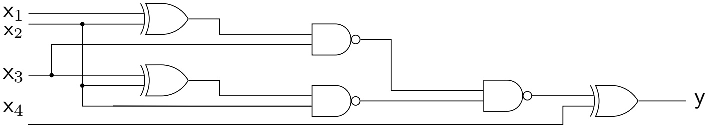
SAT scheduling is 100x faster than previous industrial scheduling tools

Application: Hardware Verification

Problem

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

Example (Formal Circuit Model)

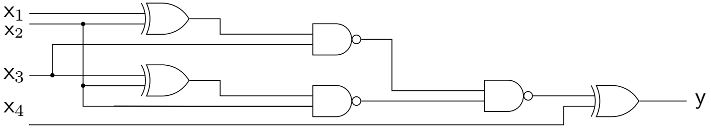


Application: Hardware Verification

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)

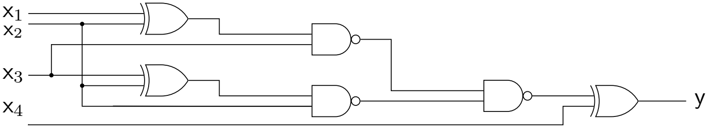


Application: Hardware Verification

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

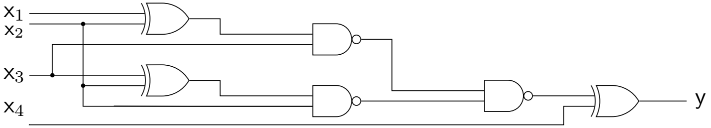
- variables for input and output
- SAT formulas for implemented behavior and expected behavior (specification)
- check for equivalence

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

Outline

- Introduction
  - Organisation
  - Why SAT and SMT?
  - Contents
- Propositional Logic
- DPLL
- Transformations to CNF
- Using SAT Solvers

Contents

Part 1: SAT

DPLL, conflict analysis, CDCL, heuristics, unsatisfiable cores, maxSAT, symmetry breaking

Part 2: SMT

DPLL(T), eager vs lazy, T-propagation, Nelson-Oppen combination, maxSMT

Part 3: Theory Solving

- equality with uninterpreted functions (congruence closure, conflict analysis)
- linear real arithmetic (simplex algorithm)
- arrays (reduction to EUF, lemmas on demand)
- bit vectors (bit blasting, preprocessing)

Practice

SAT solvers, SMT solvers, encoding, DIMACS, SMT-LIB, model checking

Contents

Part 1: SAT

DPLL, conflict analysis, CDCL, heuristics, unsatisfiable cores, maxSAT, symmetry breaking

Part 2: SMT

DPLL(T), eager vs lazy, T-propagation, Nelson-Oppen combination, maxSMT

Part 3: Theory Solving

- equality with uninterpreted functions (congruence closure, conflict analysis)
- linear real arithmetic (simplex algorithm)
- arrays (reduction to EUF, lemmas on demand)
- bit vectors (bit blasting, preprocessing)

Practice

SAT solvers, SMT solvers, encoding, DIMACS, SMT-LIB, model checking

Outline

- Introduction
- Propositional Logic
- DPLL
- Transformations to CNF
- Using SAT Solvers

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	$p \wedge q$	"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"

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	$p \rightarrow q$	"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 $\quad \neg \quad > \quad \wedge \quad > \quad \vee \quad > \quad \rightarrow, \leftrightarrow$

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

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

Definition (Propositional Logic: Semantics)

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

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

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

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:

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

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

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:

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

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

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

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 $v : \{p, q, r, \ldots\} \to \{\mathsf{F}, \mathsf{T}\}$ from atoms to truth values
- extension to formulas:

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

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

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

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

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

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

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

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 $\varphi_1, \ldots, \varphi_n \vDash \psi$
  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

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)

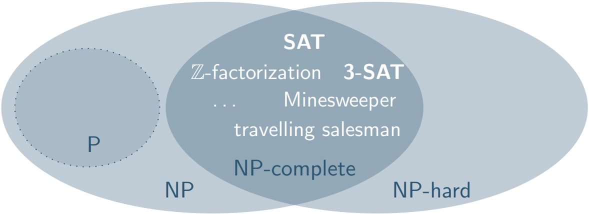
instance: propositional formula φ
question: is φ satisfiable?

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 **P** $\stackrel{?}{=}$ **NP**

Outline

- Introduction
- Propositional Logic
- DPLL
- Transformations to CNF
- Using SAT Solvers

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- DPLL is sound and complete backtracking-based search algorithm

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- DPLL is sound and complete backtracking-based search algorithm

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- DPLL is sound and complete backtracking-based search algorithm
- can be described abstractly by transition system
  (Nieuwenhuis, Oliveras, Tinelli 2006)

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- DPLL is sound and complete backtracking-based search algorithm
- can be described abstractly by transition system (Nieuwenhuis, Oliveras, Tinelli 2006)

Definition (Abstract DPLL)

- decision literal is annotated literal l^d

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- DPLL is sound and complete backtracking-based search algorithm
- can be described abstractly by transition system (Nieuwenhuis, Oliveras, Tinelli 2006)

Definition (Abstract DPLL)

- decision literal is annotated literal l^d
- state is pair $M \parallel F$ for
  - list M of (decision) literals — partial assignment
  - formula F in CNF

Approach

- most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- DPLL is sound and complete backtracking-based search algorithm
- can be described abstractly by transition system (Nieuwenhuis, Oliveras, Tinelli 2006)

Definition (Abstract DPLL)

- decision literal is annotated literal l^d
- state is pair $M \parallel F$ for
  - list M of (decision) literals — partial assignment
  - formula F in CNF
- transition rules

$$M \parallel F \quad \Longrightarrow \quad M' \parallel F' \quad \text{or} \quad \text{FailState}$$

Example

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$$

Example

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$\begin{array}{rll} & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \\ \Longrightarrow & 1^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{decide} \end{array}$$

Example

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{rrl}
 & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \\
\Longrightarrow & 1^d \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{decide} \\
\Longrightarrow & 1^d\ \overline{2} \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate}
\end{array}
$$

Example

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{llcl}
 & & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \\
\Longrightarrow & 1^d & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{decide} \\
\Longrightarrow & 1^d\ \overline{2} & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3 & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate}
\end{array}
$$

Example

$$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$$

$$\begin{array}{rrll}
 & \parallel & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \\
\Longrightarrow & 1^d \parallel & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{decide} \\
\Longrightarrow & 1^d\ \overline{2} \parallel & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3 \parallel & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3\ 4 \parallel & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate}
\end{array}$$

Example

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{rrll}
 & \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \\
\Longrightarrow & 1^d \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{decide} \\
\Longrightarrow & 1^d\ \overline{2} \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3 \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3\ 4 \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & \overline{1} \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{backtrack}
\end{array}
$$

Example

$$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$$

$$
\begin{array}{rrl}
 & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \\
\Longrightarrow & 1^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{decide} \\
\Longrightarrow & 1^d\ \overline{2} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & 1^d\ \overline{2}\ 3\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate} \\
\Longrightarrow & \overline{1} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{backtrack} \\
\Longrightarrow & \overline{1}\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4 & \text{unit propagate}
\end{array}
$$

Example

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

		$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	
$\Longrightarrow$	1^d	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	decide
$\Longrightarrow$	$1^d\ \overline{2}$	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$1^d\ \overline{2}\ 3$	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$1^d\ \overline{2}\ 3\ 4$	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$\overline{1}$	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	backtrack
$\Longrightarrow$	$\overline{1}\ 4$	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$\overline{1}\ 4\ 3^d$	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	decide

Example

$$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$$

	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$		
$\Longrightarrow$	$1^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	decide
$\Longrightarrow$	$1^d\ \overline{2} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$1^d\ \overline{2}\ 3 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$1^d\ \overline{2}\ 3\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$\overline{1} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	backtrack
$\Longrightarrow$	$\overline{1}\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate
$\Longrightarrow$	$\overline{1}\ 4\ 3^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	decide
$\Longrightarrow$	$\overline{1}\ 4\ 3^d\ 2 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ 2 \vee \overline{3} \vee \overline{4},\ 1 \vee 4$	unit propagate

Definition (DPLL Transition Rules)

- unit propagation $\quad M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\, l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M

Definition (DPLL Transition Rules)

- unit propagation $M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\, l \parallel F,\ C \vee l$
  if $M \vDash \neg C$ and l is undefined in M

- pure literal $M \parallel F \quad \Longrightarrow \quad M\, l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M

Definition (DPLL Transition Rules)

- unit propagation $\quad M \parallel F,\ C \vee l \implies M\, l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M
- pure literal $\quad M \parallel F \implies M\, l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M
- decide $\quad M \parallel F \implies M\, l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M

Definition (DPLL Transition Rules)

- unit propagation $M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M
- pure literal $M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M
- decide $M \parallel F \quad \Longrightarrow \quad M\ l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M
- backtrack $M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l^c \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and N contains no decision literals

Definition (DPLL Transition Rules)

- unit propagation $M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\, l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M

- pure literal $M \parallel F \quad \Longrightarrow \quad M\, l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M

- decide $M \parallel F \quad \Longrightarrow \quad M\, l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M

- backtrack $M\, l^d\, N \parallel F, C \quad \Longrightarrow \quad M\, l^c \parallel F, C$
  if $M\, l^d\, N \models \neg C$ and N contains no decision literals

- fail $M \parallel F, C \quad \Longrightarrow \quad$ FailState
  if $M \models \neg C$ and M contains no decision literals

Definition (DPLL Transition Rules)

- unit propagation $M \parallel F,\ C \vee l \implies M\, l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M
- pure literal $M \parallel F \implies M\, l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M
- decide $M \parallel F \implies M\, l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M
- backtrack $M\, l^d\, N \parallel F, C \implies M\, l^c \parallel F, C$
  if $M\, l^d\, N \models \neg C$ and N contains no decision literals
- fail $M \parallel F, C \implies$ FailState
  if $M \models \neg C$ and M contains no decision literals
- backjump $M\, l^d\, N \parallel F, C \implies M\, l' \parallel F, C$

Definition (DPLL Transition Rules)

- unit propagation $M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M
- pure literal $M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M
- decide $M \parallel F \quad \Longrightarrow \quad M\ l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M
- backtrack $M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l^c \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and N contains no decision literals
- fail $M \parallel F, C \quad \Longrightarrow \quad$ FailState
  if $M \models \neg C$ and M contains no decision literals
- backjump $M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that

Definition (DPLL Transition Rules)

- unit propagation $\quad M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M

- pure literal $\quad M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M

- decide $\quad M \parallel F \quad \Longrightarrow \quad M\ l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M

- backtrack $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l^c \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and N contains no decision literals

- fail $\quad M \parallel F, C \quad \Longrightarrow \quad$ FailState
  if $M \models \neg C$ and M contains no decision literals

- backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models C' \vee l'$ backjump clause

Definition (DPLL Transition Rules)

- unit propagation $\quad M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M

- pure literal $\quad M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M

- decide $\quad M \parallel F \quad \Longrightarrow \quad M\ l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M

- backtrack $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l^c \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and N contains no decision literals

- fail $\quad M \parallel F, C \quad \Longrightarrow \quad$ FailState
  if $M \models \neg C$ and M contains no decision literals

- backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models C' \vee l'$ — backjump clause
  - $M \models \neg C'$ and l' is undefined in M, and l' or l'^c occurs in F or in $M\ l^d\ N$

Definition (DPLL Transition Rules)

- unit propagation $\quad M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l$
  if $M \models \neg C$ and l is undefined in M

- pure literal $\quad M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if l occurs in F but l^c does not occur in F, and l is undefined in M

- decide $\quad M \parallel F \quad \Longrightarrow \quad M\ l^d \parallel F$
  if l or l^c occurs in F, and l is undefined in M

- backtrack $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l^c \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and N contains no decision literals

- fail $\quad M \parallel F, C \quad \Longrightarrow \quad$ FailState
  if $M \models \neg C$ and M contains no decision literals

- backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models C' \vee l'$ — backjump clause
  - $M \models \neg C'$ and l' is undefined in M, and l' or l'^c occurs in F or in $M\ l^d\ N$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

$$\| \ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

$$\begin{array}{lrll} & \| & \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \\ \Longrightarrow & 1^d \ \| & \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide} \end{array}$$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

$$\begin{array}{lrl}
 & \| \; \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \\
\Longrightarrow & 1^d \; \| \; \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide} \\
\Longrightarrow & 1^d\, 2 \; \| \; \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{unit propagate}
\end{array}$$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

$$
\begin{array}{lrcl}
 & \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \\
\Longrightarrow & 1^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide} \\
\Longrightarrow & 1^d\ 2 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{unit propagate} \\
\Longrightarrow & 1^d\ 2\ 3^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide}
\end{array}
$$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

$$\begin{array}{lrl}
 & \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \\
\Longrightarrow & 1^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide} \\
\Longrightarrow & 1^d\ 2 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{unit propagate} \\
\Longrightarrow & 1^d\ 2\ 3^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide} \\
\Longrightarrow & 1^d\ 2\ 3^d\ 4^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & \text{decide}
\end{array}$$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

	$\parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	
$\Longrightarrow$	$1^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2 \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d\ \overline{5} \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

$$
\begin{array}{llr r}
 & \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \\
\Longrightarrow & 1^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \text{decide} \\
\Longrightarrow & 1^d\ 2 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \text{unit propagate} \\
\Longrightarrow & 1^d\ 2\ 3^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \text{decide} \\
\Longrightarrow & 1^d\ 2\ 3^d\ 4^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \text{decide} \\
\Longrightarrow & 1^d\ 2\ 3^d\ 4^d\ \overline{5} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \text{unit propagate} \\
\Longrightarrow & 1^d\ 2\ 3^d\ \overline{4} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5 & & \text{backtrack}
\end{array}
$$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

	$\parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	
$\Longrightarrow$	$1^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2 \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d\ \overline{5} \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4} \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	backtrack
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4}\ 5 \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

	$\parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$		
$\Longrightarrow$	$1^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d\ \overline{5} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	backtrack
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4}\ 5 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$\ldots$	backtrack

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

	$\parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	
$\Longrightarrow$	$1^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2 \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d\ \overline{5} \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4} \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	backtrack
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4}\ 5 \parallel$	$\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$\ldots$		backtrack

Decisions 1 and 3 are incompatible: $\varphi \models \overline{1} \vee \overline{3}$

Example (Backjump)

$\varphi = (\overline{1} \lor 2) \land (\overline{1} \lor \overline{3} \lor 4 \lor 5) \land (\overline{2} \lor \overline{4} \lor \overline{5}) \land (4 \lor \overline{5}) \land (\overline{4} \lor 5)$

	$\|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	
$\Longrightarrow$	$1^d \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	decide
$\Longrightarrow$	$1^d\ 2 \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d\ \overline{5} \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4} \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	backtrack
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4}\ 5 \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	unit propagate
$\Longrightarrow$	...		backtrack
$\Longrightarrow$	$1^d\ 2\ \overline{3} \ \|\|$	$\overline{1} \lor 2,\ \overline{1} \lor \overline{3} \lor 4 \lor 5,\ \overline{2} \lor \overline{4} \lor \overline{5},\ 4 \lor \overline{5},\ \overline{4} \lor 5$	backjump

Decisions 1 and 3 are incompatible: $\varphi \models \overline{1} \lor \overline{3}$

Example (Backjump)

$\varphi = (\overline{1} \vee 2) \wedge (\overline{1} \vee \overline{3} \vee 4 \vee 5) \wedge (\overline{2} \vee \overline{4} \vee \overline{5}) \wedge (4 \vee \overline{5}) \wedge (\overline{4} \vee 5)$

	$\parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	
$\Longrightarrow$	$1^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	decide
$\Longrightarrow$	$1^d\ 2\ 3^d\ 4^d\ \overline{5} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	backtrack
$\Longrightarrow$	$1^d\ 2\ 3^d\ \overline{4}\ 5 \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	unit propagate
$\Longrightarrow$	$\ldots$	backtrack
$\Longrightarrow$	$1^d\ 2\ \overline{3} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	backjump
$\Longrightarrow^+$	$1^d\ 2\ \overline{3}\ \overline{4}\ \overline{5} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5$	

Decisions 1 and 3 are incompatible: $\varphi \models \overline{1} \vee \overline{3}$

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\parallel F \Longrightarrow_{\mathcal{B}}^{*} M \parallel F'$ then

- $F = F'$

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F'$ then

- $F = F'$
- M does not contain complementary literals

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\parallel F \Longrightarrow^*_{\mathcal{B}} M \parallel F'$ then

- $F = F'$
- M does not contain complementary literals
- literals in M are distinct

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\parallel F \Longrightarrow_{\mathcal{B}}^{*} M \parallel F'$ then

- $F = F'$
- M does not contain complementary literals
- literals in M are distinct
- length of M is bounded by number of atoms

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\parallel F \Longrightarrow^*_{\mathcal{B}} M \parallel F'$ then

- $F = F'$
- M does not contain complementary literals
- literals in M are distinct
- length of M is bounded by number of atoms

Lemma (Model Entailment)

Suppose $\parallel F \Longrightarrow^*_{\mathcal{B}} M \parallel F$ *such that*

- $M = M_0\ l_1^d\ M_1\ l_2^d\ M_2\ \dots\ l_k^d\ M_k$ *and*
- *there are no decision literals in* M_i.

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\| \; F \Longrightarrow_{\mathcal{B}}^{*} M \; \| \; F'$ then

- $F = F'$
- M does not contain complementary literals
- literals in M are distinct
- length of M is bounded by number of atoms

Lemma (Model Entailment)

Suppose $\| \; F \Longrightarrow_{\mathcal{B}}^{*} M \; \| \; F$ *such that*

- $M = M_0 \; l_1^d \; M_1 \; l_2^d \; M_2 \; \dots \; l_k^d \; M_k$ *and*
- *there are no decision literals in* M_i.

Then $F, \; l_1, \dots, l_i \vDash M_i$ *for all* $0 \leqslant i \leqslant k$.

Definition

basic DPLL $\mathcal{B}$ consists of unit propagation, decide, fail, and backjump

Properties

if $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F'$ then

- $F = F'$
- M does not contain complementary literals
- literals in M are distinct
- length of M is bounded by number of atoms

Lemma (Model Entailment)

Suppose $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F$ *such that*

- $M = M_0\ l_1^d\ M_1\ l_2^d\ M_2\ \ldots\ l_k^d\ M_k$ *and*
- *there are no decision literals in* M_i.

decisions imply
all other literals in M

Then $F,\ l_1, \ldots, l_i \vDash M_i$ *for all* $0 \leqslant i \leqslant k$.

Theorem (Termination)

for any formula F there are no infinite derivations

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots$$

Theorem (Termination)

for any formula F there are no infinite derivations

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots$$

Proof.

- for list of distinct literals M, define $a(M) = n - |M|$ where
  - n is total number of atoms
  - $|M|$ is length of M

Theorem (Termination)

for any formula F there are no infinite derivations

$$\| \; F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots$$

missing literals in M

Proof.

- for list of distinct literals M, define $a(M) = n - |M|$ where
  - n is total number of atoms
  - $|M|$ is length of M

Theorem (Termination)

for any formula F there are no infinite derivations

$$\| F \implies_{\mathcal{B}} S_1 \implies_{\mathcal{B}} S_2 \implies_{\mathcal{B}} \dots$$

missing literals in M

Proof.

- for list of distinct literals M, define $a(M) = n - |M|$ where
  - n is total number of atoms
  - $|M|$ is length of M
- measure state $M_0\ l_1^d\ M_1\ l_2^d\ M_2 \dots l_k^d\ M_k \parallel F$ by tuple

$$(a(M_0),\ a(M_1),\ \dots,\ a(M_k),\ \underbrace{\infty, \dots, \infty}_{n-k})$$

Theorem (Termination)

for any formula F there are no infinite derivations

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots$$

missing literals in M

Proof.

- for list of distinct literals M, define $a(M) = n - |M|$ where
  - n is total number of atoms
  - $|M|$ is length of M
- measure state $\quad M_0\ l_1^d\ M_1\ l_2^d\ M_2 \dots\ l_k^d\ M_k \parallel F \quad$ by tuple

$$(a(M_0),\ a(M_1),\ \dots,\ a(M_k),\ \underbrace{\infty, \dots, \infty}_{n-k})$$

- compare tuples lexicographically by extension of $>_{\mathbb{N}}$ with ∞ maximal

Theorem (Termination)

for any formula F there are no infinite derivations

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots$$

missing literals in M

Proof.

- for list of distinct literals M, define $a(M) = n - |M|$ where
  - n is total number of atoms
  - $|M|$ is length of M
- measure state $\quad M_0\ l_1^d\ M_1\ l_2^d\ M_2 \dots\ l_k^d\ M_k \parallel F \quad$ by tuple

$$(a(M_0),\ a(M_1),\ \dots,\ a(M_k),\ \underbrace{\infty, \dots, \infty}_{n-k})$$

- compare tuples lexicographically by extension of $>_{\mathbb{N}}$ with ∞ maximal
- every transition step decreases measure

∎

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$\| \; \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots \qquad\qquad (n, \infty, \ldots)$$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$\begin{array}{llll} & \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & & (n, \infty, \ldots) \\ \Longrightarrow & 1^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{decide} & (n, n, \infty, \ldots) \end{array}$$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{rrlrl}
 & & \| \; \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & & (n, \infty, \ldots) \\
\Longrightarrow & 1^d & \| \; \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{decide} & (n, n, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2} & \| \; \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-1, \infty, \ldots)
\end{array}
$$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{lrlll}
 & & \| \; \overline{1} \vee \overline{2},\; 2 \vee 3,\; \overline{1} \vee \overline{3} \vee 4,\; \ldots & & (n, \infty, \ldots) \\
\Longrightarrow & 1^d & \| \; \overline{1} \vee \overline{2},\; 2 \vee 3,\; \overline{1} \vee \overline{3} \vee 4,\; \ldots & \text{decide} & (n, n, \infty, \ldots) \\
\Longrightarrow & 1^d\, \overline{2} & \| \; \overline{1} \vee \overline{2},\; 2 \vee 3,\; \overline{1} \vee \overline{3} \vee 4,\; \ldots & \text{unit propagate} & (n, n-1, \infty, \ldots) \\
\Longrightarrow & 1^d\, \overline{2}\, 3 & \| \; \overline{1} \vee \overline{2},\; 2 \vee 3,\; \overline{1} \vee \overline{3} \vee 4,\; \ldots & \text{unit propagate} & (n, n-2, \infty, \ldots)
\end{array}
$$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

	$\| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$		$(n, \infty, \ldots)$
$\Longrightarrow$	$1^d \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	decide	$(n, n, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2} \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-1, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3 \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-2, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3\ 4 \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-3, \infty, \ldots)$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

	$\| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$		$(n, \infty, \ldots)$
$\Longrightarrow$	$1^d \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	decide	$(n, n, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2} \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-1, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3 \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-2, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3\ 4 \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-3, \infty, \ldots)$
$\Longrightarrow$	$\overline{1} \ \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	backtrack	$(n-1, \infty, \ldots)$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$		$(n, \infty, \ldots)$
$\Longrightarrow$	$1^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	decide	$(n, n, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-1, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-2, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-3, \infty, \ldots)$
$\Longrightarrow$	$\overline{1} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	backtrack	$(n-1, \infty, \ldots)$
$\Longrightarrow$	$\overline{1}\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n-2, \infty, \ldots)$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{rrlrl}
 & & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & & (n, \infty, \ldots) \\
\Longrightarrow & 1^d & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{decide} & (n, n, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2} & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-1, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2}\ 3 & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-2, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2}\ 3\ 4 & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-3, \infty, \ldots) \\
\Longrightarrow & \overline{1} & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{backtrack} & (n-1, \infty, \ldots) \\
\Longrightarrow & \overline{1}\ 4 & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n-2, \infty, \ldots) \\
\Longrightarrow & \overline{1}\ 4\ 3^d & \| \ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{decide} & (n-2, n, \infty, \ldots)
\end{array}
$$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

$$
\begin{array}{rrlrl}
 & \| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & & (n, \infty, \ldots) \\
\Longrightarrow & 1^d \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{decide} & (n, n, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2} \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-1, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2}\ 3 \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-2, \infty, \ldots) \\
\Longrightarrow & 1^d\ \overline{2}\ 3\ 4 \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n, n-3, \infty, \ldots) \\
\Longrightarrow & \overline{1} \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{backtrack} & (n-1, \infty, \ldots) \\
\Longrightarrow & \overline{1}\ 4 \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n-2, \infty, \ldots) \\
\Longrightarrow & \overline{1}\ 4\ 3^d \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{decide} & (n-2, n, \infty, \ldots) \\
\Longrightarrow & \overline{1}\ 4\ 3^d\ 2 \,\| & \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots & \text{unit propagate} & (n-2, n-1, \infty, \ldots)
\end{array}
$$

Example (Revisited for Termination)

$\varphi = (\overline{1} \vee \overline{2}) \wedge (2 \vee 3) \wedge (\overline{1} \vee \overline{3} \vee 4) \wedge (2 \vee \overline{3} \vee \overline{4}) \wedge (1 \vee 4)$

	$\parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$		$(n, \infty, \ldots)$
$\Longrightarrow$	$1^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	decide	$(n, n, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-1, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-2, \infty, \ldots)$
$\Longrightarrow$	$1^d\ \overline{2}\ 3\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n, n-3, \infty, \ldots)$
$\Longrightarrow$	$\overline{1} \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	backtrack	$(n-1, \infty, \ldots)$
$\Longrightarrow$	$\overline{1}\ 4 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n-2, \infty, \ldots)$
$\Longrightarrow$	$\overline{1}\ 4\ 3^d \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	decide	$(n-2, n, \infty, \ldots)$
$\Longrightarrow$	$\overline{1}\ 4\ 3^d\ 2 \parallel \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{1} \vee \overline{3} \vee 4,\ \ldots$	unit propagate	$(n-2, n-1, \infty, \ldots)$

Observation

- decide replaces ∞ by n
- unit propagate, backtrack, and backjump replace m by $m-1$

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \ldots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \mathrm{FailState}$ *then* F *is unsatisfiable*

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ then F is unsatisfiable

Proof.

- must have $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}} \text{FailState}$
  such that M contains no decision literals and $M \models \neg C$ for some C in F

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ then F is unsatisfiable

Proof.

- must have $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}} \text{FailState}$
  such that M contains no decision literals and $M \models \neg C$ for some C in F
- by Model Entailment Lemma $F \models M$, so $F \models \neg C$

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ *then* F *is unsatisfiable*

Proof.

- must have $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}} \text{FailState}$
  such that M contains no decision literals and $M \models \neg C$ for some C in F
- by Model Entailment Lemma $F \models M$, so $F \models \neg C$
- also have $F \models C$ because C is in F, so F is unsatisfiable ∎

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \ldots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ *then* F *is unsatisfiable*

Proof.

- must have $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}} \text{FailState}$
  such that M contains no decision literals and $M \models \neg C$ for some C in F
- by Model Entailment Lemma $F \models M$, so $F \models \neg C$
- also have $F \models C$ because C is in F, so F is unsatisfiable ∎

Theorem

if $S_n = M \| F'$ *then* F *is satisfiable and* $M \models F$

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ *then* F *is unsatisfiable*

Proof.

- must have $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}} \text{FailState}$
  such that M contains no decision literals and $M \vDash \neg C$ for some C in F
- by Model Entailment Lemma $F \vDash M$, so $F \vDash \neg C$
- also have $F \vDash C$ because C is in F, so F is unsatisfiable ∎

Theorem

if $S_n = M \| F'$ *then* F *is satisfiable and* $M \vDash F$

Proof.

- have $F = F'$

Consider maximal derivation with final state S_n:

$$\| F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ then F is unsatisfiable

Proof.

- must have $\| F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}} \text{FailState}$
  such that M contains no decision literals and $M \vDash \neg C$ for some C in F
- by Model Entailment Lemma $F \vDash M$, so $F \vDash \neg C$
- also have $F \vDash C$ because C is in F, so F is unsatisfiable ■

Theorem

if $S_n = M \| F'$ then F is satisfiable and $M \vDash F$

Proof.

- have $F = F'$
- S_n is final, so all literals of F are defined in M (otherwise decide applicable)

Consider maximal derivation with final state S_n:

$$\| \; F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n = \text{FailState}$ *then* F *is unsatisfiable*

Proof.

- must have $\| \; F \Longrightarrow_{\mathcal{B}}^{*} M \| F \Longrightarrow_{\text{fail}}$ FailState
  such that M contains no decision literals and $M \models \neg C$ for some C in F
- by Model Entailment Lemma $F \models M$, so $F \models \neg C$
- also have $F \models C$ because C is in F, so F is unsatisfiable ■

Theorem

if $S_n = M \| F'$ *then* F *is satisfiable and* $M \models F$

Proof.

- have $F = F'$
- S_n is final, so all literals of F are defined in M (otherwise decide applicable)
- $\nexists$ clause C in F such that $M \models \neg C$ (otherwise backjump or fail applicable)

Consider maximal derivation with final state S_n:

$$\| \; F \quad \Longrightarrow_{\mathcal{B}} \quad S_1 \quad \Longrightarrow_{\mathcal{B}} \quad S_2 \quad \Longrightarrow_{\mathcal{B}} \quad \dots \quad \Longrightarrow_{\mathcal{B}} \quad S_n$$

Theorem

if $S_n =$ FailState then F is unsatisfiable

Proof.

- must have $\| \; F \Longrightarrow_{\mathcal{B}}^{*} M \; \| \; F \Longrightarrow_{\text{fail}}$ FailState
  such that M contains no decision literals and $M \vDash \neg C$ for some C in F
- by Model Entailment Lemma $F \vDash M$, so $F \vDash \neg C$
- also have $F \vDash C$ because C is in F, so F is unsatisfiable ■

Theorem

if $S_n = M \; \| \; F'$ then F is satisfiable and $M \vDash F$

Proof.

- have $F = F'$
- S_n is final, so all literals of F are defined in M (otherwise decide applicable)
- $\nexists$ clause C in F such that $M \vDash \neg C$ (otherwise backjump or fail applicable)
- so M satisfies F ($M \vDash F$) ■

DPLL

Martin Davis and Hilary Putnam.
A Computing Procedure for Quantification Theory.
Journal of the ACM 7(3), pp. 201–215, 1960.

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli.
Solving SAT and SAT Modulo Theories: From an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).
Journal of the ACM 53(6), pp. 937–977, 2006.

Application Examples

Roope Kaivola *et al.*
Replacing Testing with Formal Verification in Intel CoreTM i7 Processor Execution Engine Validation.
Proc. 21st International Conference on Computer Aided Verification, pp. 414–429, 2009.

Andrei Horbach, Thomas Bartsch, and Dirk Briskorn.
Using a SAT solver to Schedule Sports Leagues.
Journal of Scheduling 15, pp. 117–125, 2012.

Marijn Heule, Oliver Kullmann, and Victor Marek.
Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer.
Proc. 16th International Conference on Theory and Applications of Satisfiability Testing, pp. 228–245, 2016.

Outline

- Introduction
- Propositional Logic
- DPLL
- Transformations to CNF
- Using SAT Solvers

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

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

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

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

Example

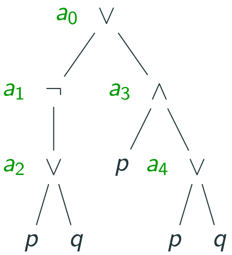
$$p \vee q \ \approx\ \top \qquad\qquad p \wedge \neg p \ \approx\ q \wedge \neg q \qquad\qquad p \wedge \neg p \ \not\approx\ p \wedge \neg q$$

Example (Tseitin's Transformation)

- $\varphi = \neg(p \lor q) \lor (p \land (p \lor q))$

Example (Tseitin's Transformation)

- $\varphi = \neg(p \lor q) \lor (p \land (p \lor q))$



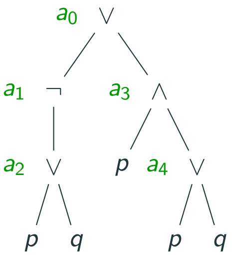
Example (Tseitin's Transformation)

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

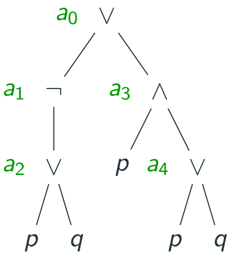
  $a_0 : \neg(p \vee q) \vee (p \wedge (p \vee q))$ $\quad a_1 : \neg(p \vee q)$

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

  $a_4 : p \vee q$



Example (Tseitin's Transformation)



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

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

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

  $a_4 \colon p \vee q$

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

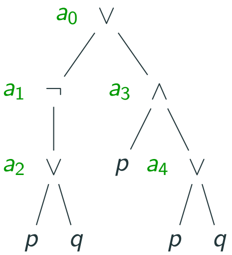
Example (Tseitin's Transformation)

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

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

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

  $a_4 : p \vee q$



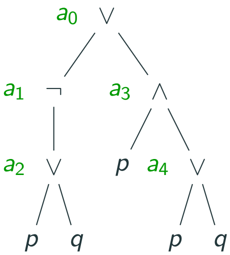
- $\varphi \approx \quad a_0 \wedge (a_0 \leftrightarrow a_1 \vee a_3) \wedge (a_1 \leftrightarrow \neg a_2) \wedge (a_2 \leftrightarrow p \vee q) \wedge (a_3 \leftrightarrow p \wedge a_4) \wedge (a_4 \leftrightarrow p \vee q)$
- every $\leftrightarrow$ subexpression can be replaced by at most three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

$$a \leftrightarrow b \wedge c \quad \equiv \quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

$$a \leftrightarrow b \vee c \quad \equiv \quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c)$$

$$a \leftrightarrow \neg b \quad \equiv \quad (\neg a \vee \neg b) \wedge (a \vee b)$$

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:

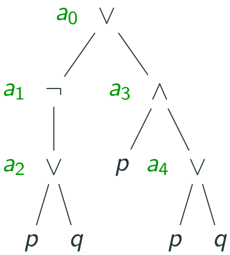
$$a \leftrightarrow b \wedge c \quad \equiv \quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

$$a \leftrightarrow b \vee c \quad \equiv \quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c)$$

$$a \leftrightarrow \neg b \quad \equiv \quad (\neg a \vee \neg b) \wedge (a \vee b)$$

- 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:

$$a \leftrightarrow b \wedge c \quad \equiv \quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$
$$a \leftrightarrow b \vee c \quad \equiv \quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c)$$
$$a \leftrightarrow \neg b \quad \equiv \quad (\neg a \vee \neg b) \wedge (a \vee b)$$

- common subexpressions can be shared

Observation

bi-implication $\leftrightarrow$ in Tseitin's transformation can be replaced by $\rightarrow$ or $\leftarrow$:
direction of implication $\rightarrow$ or $\leftarrow$ depends on polarity of subformula

Observation

bi-implication $\leftrightarrow$ in Tseitin's transformation can be replaced by $\rightarrow$ or $\leftarrow$:
direction of implication $\rightarrow$ or $\leftarrow$ depends on polarity of subformula

Observation

bi-implication $\leftrightarrow$ in Tseitin's transformation can be replaced by $\rightarrow$ or $\leftarrow$:
direction of implication $\rightarrow$ or $\leftarrow$ depends on polarity of subformula

Definition

for φ subformula occurrence of ψ

- let k be number of negations above φ in syntax tree of ψ

Observation

bi-implication $\leftrightarrow$ in Tseitin's transformation can be replaced by $\rightarrow$ or $\leftarrow$:
direction of implication $\rightarrow$ or $\leftarrow$ depends on polarity of subformula

Definition

for φ subformula occurrence of ψ

- let k be number of negations above φ in syntax tree of ψ
- polarity of φ is $+$ if k is even, and $-$ otherwise

Observation

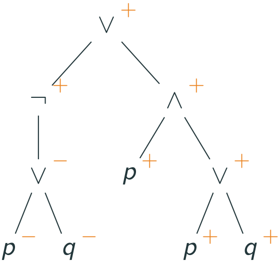
bi-implication $\leftrightarrow$ in Tseitin's transformation can be replaced by $\rightarrow$ or $\leftarrow$:
direction of implication $\rightarrow$ or $\leftarrow$ depends on polarity of subformula

Definition

for φ subformula occurrence of ψ

- let k be number of negations above φ in syntax tree of ψ
- polarity of φ is $+$ if k is even, and $-$ otherwise

Example



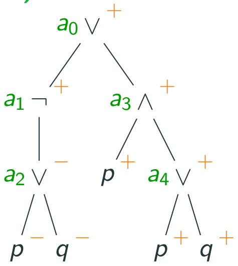
Example (Plaisted and Greenbaum's Transformation)

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

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

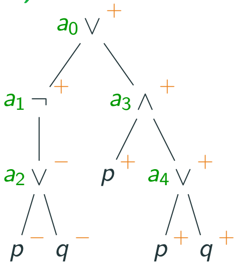
$a_2 : p \vee q$ $\quad a_3 : p \wedge (p \vee q)$

$a_4 : p \vee q$



Example (Plaisted and Greenbaum's Transformation)

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



- use fresh propositional variable for every connective

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

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

$a_4 : p \vee q$

- add $(a_i \rightarrow \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

$\varphi \approx \quad a_0$

Example (Plaisted and Greenbaum's Transformation)

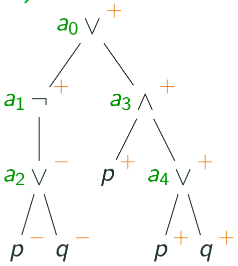
- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$

- use fresh propositional variable for every connective

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

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

  $a_4 : p \vee q$



- add $(a_i \rightarrow \ldots)$ if polarity of a_i is positive and $(a_i \leftarrow \ldots)$ if negative

  $\varphi \approx \quad a_0$

Example (Plaisted and Greenbaum's Transformation)

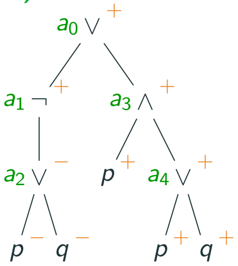
- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$

- use fresh propositional variable for every connective

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

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

  $a_4 : p \vee q$



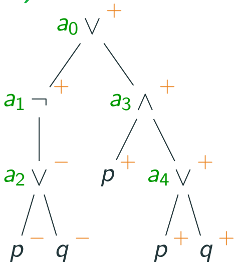
- add $(a_i \to \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

  $\varphi \approx \quad a_0 \wedge (a_0 \to a_1 \vee a_3)$

Example (Plaisted and Greenbaum's Transformation)

- $\varphi = \neg(p \lor q) \lor (p \land (p \lor q))$

- use fresh propositional variable for every connective



$a_0 : \neg(p \lor q) \lor (p \land (p \lor q))$ $\quad a_1 : \neg(p \lor q)$

$a_2 : p \lor q$ $\quad a_3 : p \land (p \lor q)$

$a_4 : p \lor q$

- add $(a_i \to \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

$\varphi \approx \quad a_0 \land (a_0 \to a_1 \lor a_3) \land (a_1 \to \neg a_2)$

Example (Plaisted and Greenbaum's Transformation)

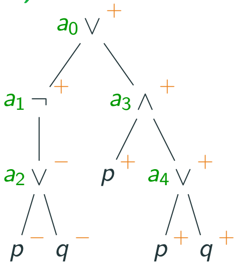
- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$

- use fresh propositional variable for every connective

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

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

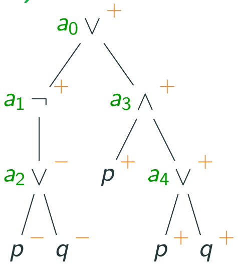
  $a_4 : p \vee q$



- add $(a_i \rightarrow \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

  $\varphi \approx \quad a_0 \wedge (a_0 \rightarrow a_1 \vee a_3) \wedge (a_1 \rightarrow \neg a_2) \wedge (a_2 \leftarrow p \vee q)$

Example (Plaisted and Greenbaum's Transformation)



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

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

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

  $a_4 \colon p \vee q$

- add $(a_i \rightarrow \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

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

Example (Plaisted and Greenbaum's Transformation)

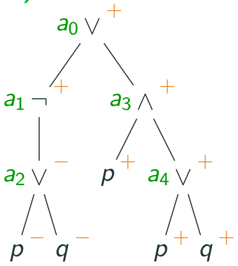
- $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$

- use fresh propositional variable for every connective

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

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

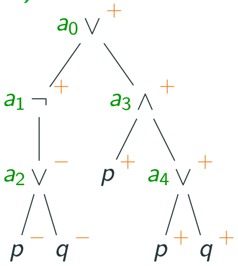
  $a_4 : p \vee q$



- add $(a_i \to \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

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

Example (Plaisted and Greenbaum's Transformation)



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

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

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

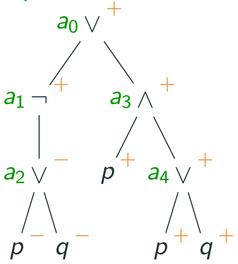
  $a_4 \colon p \vee q$

- add $(a_i \rightarrow \ldots)$ if polarity of a_i is positive and $(a_i \leftarrow \ldots)$ if negative

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

- every $\leftarrow$ and $\rightarrow$ subexpression can be replaced by at most two clauses:

Example (Plaisted and Greenbaum's Transformation)



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

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

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

  $a_4\colon p \vee q$

- add $(a_i \rightarrow \dots)$ if polarity of a_i is positive and $(a_i \leftarrow \dots)$ if negative

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

- every $\leftarrow$ and $\rightarrow$ subexpression can be replaced by at most two clauses:

  $$a \rightarrow b \wedge c \;\equiv\; (\neg a \vee b) \wedge (\neg a \vee c) \qquad a \leftarrow b \wedge c \;\equiv\; (a \vee \neg b \vee \neg c)$$

  $$a \rightarrow b \vee c \;\equiv\; (\neg a \vee b \vee c) \qquad a \leftarrow b \vee c \;\equiv\; (a \vee \neg b) \wedge (a \vee \neg c)$$

  $$a \rightarrow \neg b \;\equiv\; (\neg a \vee \neg b) \qquad a \leftarrow \neg b \;\equiv\; (a \vee b)$$

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 $x_1, \ldots, x_n$
- 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

Z3

common open source SAT/SMT solver

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False`   boolean constants

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False` — boolean constants
- `Bool(`*name*`)` — propositional variable named *name*
  (calling `Bool(`*name*`)` twice yields same variable)

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False` boolean constants
- `Bool(`*name*`)` propositional variable named *name*
  (calling `Bool(`*name*`)` twice yields same variable)
- `FreshBool(`*pre*`)` new propositional variable with name prefix *name*
  (calling `FreshBool(`*pre*`)` twice does not yield same variable!)

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False` boolean constants
- `Bool(`*name*`)` propositional variable named *name*
  (calling `Bool(`*name*`)` twice yields same variable)
- `FreshBool(`*pre*`)` new propositional variable with name prefix *name*
  (calling `FreshBool(`*pre*`)` twice does not yield same variable!)
- `And(`$a_1, \ldots, a_n$`)` conjunction with arbitrarily many arguments

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False` boolean constants
- `Bool(`*name*`)` propositional variable named *name*
  (calling `Bool(`*name*`)` twice yields same variable)
- `FreshBool(`*pre*`)` new propositional variable with name prefix *name*
  (calling `FreshBool(`*pre*`)` twice does not yield same variable!)
- `And`$(a_1, \ldots, a_n)$ conjunction with arbitrarily many arguments
- `Or`$(a_1, \ldots, a_n)$ disjunction with arbitrarily many arguments

Z3

common open source SAT/SMT solver

z3: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False` boolean constants
- `Bool(`*name*`)` propositional variable named *name*
  (calling `Bool(`*name*`)` twice yields same variable)
- `FreshBool(`*pre*`)` new propositional variable with name prefix *name*
  (calling `FreshBool(`*pre*`)` twice does not yield same variable!)
- `And(`$a_1, \ldots, a_n$`)` conjunction with arbitrarily many arguments
- `Or(`$a_1, \ldots, a_n$`)` disjunction with arbitrarily many arguments
- `Not(`a`)` negation

Z3

common open source SAT/SMT solver

`z3`: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

- `True`, `False` — boolean constants
- `Bool(`*name*`)` — propositional variable named *name*
  (calling `Bool(`*name*`)` twice yields same variable)
- `FreshBool(`*pre*`)` — new propositional variable with name prefix *name*
  (calling `FreshBool(`*pre*`)` twice does not yield same variable!)
- `And(`$a_1, \ldots, a_n$`)` — conjunction with arbitrarily many arguments
- `Or(`$a_1, \ldots, a_n$`)` — disjunction with arbitrarily many arguments
- `Not(`a`)` — negation
- `Implies(`a, b`)` — implication

Z3

common open source SAT/SMT solver

z3: Python interface to Z3

- from `https://github.com/Z3Prover/z3` or via `pip install z3`
- API: `https://z3prover.github.io/api/html/namespacez3py.html`

Building Formulas

`True`, `False`	boolean constants
`Bool(`*name*`)`	propositional variable named *name*
	(calling `Bool(`*name*`)` twice yields same variable)
`FreshBool(`*pre*`)`	new propositional variable with name prefix *name*
	(calling `FreshBool(`*pre*`)` twice does not yield same variable!)
`And(`$a_1, \ldots, a_n$`)`	conjunction with arbitrarily many arguments
`Or(`$a_1, \ldots, a_n$`)`	disjunction with arbitrarily many arguments
`Not(`a`)`	negation
`Implies(`a, b`)`	implication
`Xor(`a, b`)`	exclusive or

Solving Formulas

- `Solver()` create new solver object

Solving Formulas

- `Solver()` create new solver object
- `Solver.add(`$\varphi_1, \ldots, \varphi_n$`)` require constraints $\varphi_1, \ldots, \varphi_n$ to be true

Solving Formulas

- `Solver()` create new solver object
- `Solver.add(`$\varphi_1, \ldots, \varphi_n$`)` require constraints $\varphi_1, \ldots, \varphi_n$ to be true
- `Solver.check()` check for satisfiability

Solving Formulas

- `Solver()` create new solver object
- `Solver.add(`$\varphi_1, \ldots, \varphi_n$`)` require constraints $\varphi_1, \ldots, \varphi_n$ to be true
- `Solver.check()` check for satisfiability
- `Solver.model()` returns valuation (after successful call of `check`)

Solving Formulas

- `Solver()` create new solver object
- `Solver.add(`$\varphi_1, \ldots, \varphi_n$`)` require constraints $\varphi_1, \ldots, \varphi_n$ to be true
- `Solver.check()` check for satisfiability
- `Solver.model()` returns valuation (after successful call of `check`)

Moreover ...

- `simplify(`φ`)` simplifies formula φ

Solving Formulas

- `Solver()` create new solver object
- `Solver.add(`$\varphi_1, \ldots, \varphi_n$`)` require constraints $\varphi_1, \ldots, \varphi_n$ to be true
- `Solver.check()` check for satisfiability
- `Solver.model()` returns valuation (after successful call of `check`)

Moreover ...

- `simplify(`φ`)` simplifies formula φ
- `Solver.statistics()` is map of solving statistics

Example

```
from z3 import *

p = Bool('p') # create variable named 'p'
foo1 = FreshBool('foo') # create fresh variables prefixed 'foo'
foo2 = FreshBool('foo')

phi = Or(p, p, And(foo2, Xor(foo1, Not(foo1)), True), False)
print(phi) # Or(p, p, And(foo!1, Xor(foo!0, Not(foo!0)), True), False)
psi = simplify(phi)
print(psi) # Or(p, foo!1)

solver = Solver()
solver.add(psi) # assert that psi should be true
solver.add(Implies(foo1,p), Or(foo1, foo2)) # assert something else

print solver # [Or(p, foo!1), Implies(foo!0, p), Or(foo!0, foo!1)]
result = solver.check() # check for satisfiability

if result:
  model = solver.model() # get valuation
  print model[p], model[foo1], model[foo2] # False False True
```

Example (Minesweeper)

	3		1
	8		3
			2

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathrm{T}$ iff cell i has mine
- constraints for every hint (number in grid)

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathsf{T}$ iff cell i has mine
- constraints for every hint (number in grid)
  - 1 $(x_2 \vee x_5 \vee x_6)$

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathsf{T}$ iff cell i has mine
- constraints for every hint (number in grid)
  - **1** $(x_2 \vee x_5 \vee x_6) \wedge ((\neg x_2 \wedge \neg x_5) \vee (\neg x_2 \wedge \neg x_6) \vee (\neg x_5 \wedge \neg x_6))$

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathsf{T}$ iff cell i has mine
- constraints for every hint (number in grid)
  - **1** $(x_2 \vee x_5 \vee x_6) \wedge ((\neg x_2 \wedge \neg x_5) \vee (\neg x_2 \wedge \neg x_6) \vee (\neg x_5 \wedge \neg x_6))$
  - **8** $x_3 \wedge x_4 \wedge x_5 \wedge x_7 \wedge x_8 \wedge x_9 \wedge x_{10} \wedge x_{11}$

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathsf{T}$ iff cell i has mine
- constraints for every hint (number in grid)
  - **1** $(x_2 \vee x_5 \vee x_6) \wedge ((\neg x_2 \wedge \neg x_5) \vee (\neg x_2 \wedge \neg x_6) \vee (\neg x_5 \wedge \neg x_6))$
  - **8** $x_3 \wedge x_4 \wedge x_5 \wedge x_7 \wedge x_8 \wedge x_9 \wedge x_{10} \wedge x_{11}$
  - **3** $((x_5 \wedge x_6 \wedge x_8) \vee (x_5 \wedge x_6 \wedge x_{11}) \vee (x_5 \wedge x_8 \wedge x_{11}) \vee (x_6 \wedge x_8 \wedge x_{11})) \wedge (\neg x_5 \vee \neg x_6 \vee \neg x_8 \vee \neg x_{11})$

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathsf{T}$ iff cell i has mine
- constraints for every hint (number in grid)
  - **1** $(x_2 \vee x_5 \vee x_6) \wedge ((\neg x_2 \wedge \neg x_5) \vee (\neg x_2 \wedge \neg x_6) \vee (\neg x_5 \wedge \neg x_6))$
  - **8** $x_3 \wedge x_4 \wedge x_5 \wedge x_7 \wedge x_8 \wedge x_9 \wedge x_{10} \wedge x_{11}$
  - **3** $((x_5 \wedge x_6 \wedge x_8) \vee (x_5 \wedge x_6 \wedge x_{11}) \vee (x_5 \wedge x_8 \wedge x_{11}) \vee (x_6 \wedge x_8 \wedge x_{11})) \wedge (\neg x_5 \vee \neg x_6 \vee \neg x_8 \vee \neg x_{11})$
  - **2** $x_8 \wedge x_{11}$

Example (Minesweeper)

	3		1
	8		3
			2

x_1		x_2	
x_3	x_4	x_5	x_6
x_7		x_8	
x_9	x_{10}	x_{11}	

SAT Encoding

- variable x_i for each unknown cell i, $v(x_i) = \mathrm{T}$ iff cell i has mine
- constraints for every hint (number in grid)

  1 $(x_2 \vee x_5 \vee x_6) \wedge ((\neg x_2 \wedge \neg x_5) \vee (\neg x_2 \wedge \neg x_6) \vee (\neg x_5 \wedge \neg x_6))$

  8 $x_3 \wedge x_4 \wedge x_5 \wedge x_7 \wedge x_8 \wedge x_9 \wedge x_{10} \wedge x_{11}$

  3 $((x_5 \wedge x_6 \wedge x_8) \vee (x_5 \wedge x_6 \wedge x_{11}) \vee (x_5 \wedge x_8 \wedge x_{11}) \vee (x_6 \wedge x_8 \wedge x_{11})) \wedge (\neg x_5 \vee \neg x_6 \vee \neg x_8 \vee \neg x_{11})$

  2 $x_8 \wedge x_{11}$

  3 $\bigvee_{1 \leqslant i,j \leqslant 5, i \neq j} \neg x_i \wedge \neg x_j \qquad \bigvee_{1 \leqslant i,j,k \leqslant 5, i \neq j, i \neq k, j \neq k} x_i \wedge x_j \wedge x_k$