Automated Reasoning

Week 1. Course introduction and **SAT** basics

Cynthia Kop

Fall 2024

Automated Reasoning, IMC009

Automated Reasoning, IMC009

dr Cynthia Kop
c.kop@cs.ru.nl

dr Sebastian Junges
sebastian.junges@ru.nl

Automated Reasoning, IMC009

dr Cynthia Kop
c.kop@cs.ru.nl

dr Sebastian Junges
sebastian.junges@ru.nl

RU, Maria Montessori Building, MM02.610 (Wednesday mornings)

Organization

Organization

- Course + examination

Organization

- Course + examination
- Practical assignment

Organization

- Course + examination
- Practical assignment
- Each 50%; each has to be at least 5.

Organization

- Course $+$ examination
- Practical assignment
- Each 50%; each has to be at least 5.
- Practical assignment to be done in groups of at most 2.

Organization

- Course $+$ examination
- Practical assignment
- Each 50%; each has to be at least 5.
- Practical assignment to be done in groups of at most 2.
- Deadline for the first part of the practical assignment: 27 October, 2024

Organization

- Course + examination
- Practical assignment
- Each 50%; each has to be at least 5.
- Practical assignment to be done in groups of at most 2.
- Deadline for the first part of the practical assignment: 27 October, 2024
- Deadline for the second part of the practical assignment: 20 December, 2024

An open question

$$P = NP?$$

An open question

$$\text{P} = \text{NP?}$$

- P $\approx$ problems whose solution you can find easily

An open question

$$P = NP?$$

- P $\approx$ problems whose solution you can find easily
- NP $\approx$ problems whose solution you can verify easily

A well-known game: Sudoku

2				8	3	4	7	6
1			9					
6					4			
5		6					9	
8								4
	7					5		1
			4					9
					8			2
9	1	4	5	2				3

2	5	9	1	8	3	4	7	6
1	4	7	9	6	5	3	2	8
6	8	3	2	7	4	9	1	5
5	3	6	8	4	1	2	9	7
8	9	1	7	5	2	6	3	4
4	7	2	6	3	9	5	8	1
3	2	8	4	1	6	7	5	9
7	6	5	3	9	8	1	4	2
9	1	4	5	2	7	8	6	3

A thought experiment

What if: finding a solution is easy when you can check it?

A thought experiment

What if: finding a solution is easy when you can check it?

Consequences for:

- Encryption
- Mathematics
- Debugging
- Scheduling
- Any kind of (automatic) problem solving

A thought experiment

What if: finding a solution is easy when you can check it?

Consequences for:

- Encryption
- Mathematics
- Debugging
- Scheduling
- Any kind of (automatic) problem solving

A thought experiment

What if: finding a solution is easy when you can check it?

Consequences for:

- Encryption
- Mathematics
- Debugging
- Scheduling
- Any kind of (automatic) problem solving

A thought experiment

What if: finding a solution is easy when you can check it?

Consequences for:

- Encryption
- Mathematics
- Debugging
- Scheduling
- Any kind of (automatic) problem solving

A thought experiment

What if: finding a solution is easy when you can check it?

Consequences for:

- Encryption
- Mathematics
- Debugging
- Scheduling
- Any kind of (automatic) problem solving

A thought experiment

What if: finding a solution is easy when you can check it?

Consequences for:

- Encryption
- Mathematics
- Debugging
- Scheduling
- Any kind of (automatic) problem solving

New programming languages!

New programming languages!

2				8	3	4	7	6
1			9					
6					4			
5		6					9	
8								4
	7					5		1
			4					9
					8			2
9	1	4	5	2				3

New programming languages!

```
function clues {
  1,1 ⇒ 2 ; 1,2 ⇒ 1 ; 1,3 ⇒ 6 ;
  5,1 ⇒ 8 ; 6,1 ⇒ 3 ; 4,2 ⇒ 9 ; 6,3 ⇒ 4 ;
  7,1 ⇒ 4 ; 8,1 ⇒ 7 ; 9,1 ⇒ 6 ;
  1,4 ⇒ 5 ; 3,4 ⇒ 6 ; 1,5 ⇒ 8 ; 2,6 ⇒ 7 ;
  8,4 ⇒ 9 ; 9,5 ⇒ 4 ; 7,6 ⇒ 5 ; 9,6 ⇒ 1 ;
  1,9 ⇒ 9 ; 2,9 ⇒ 1 ; 3,9 ⇒ 4 ;
  4,7 ⇒ 4 ; 6,8 ⇒ 8 ; 4,9 ⇒ 5 ; 5,9 ⇒ 2 ;
  9,7 ⇒ 9 ; 9,8 ⇒ 2 ; 9,9 ⇒ 3 ;
  _ ⇒ 0
}

===============

declare field[x,y] :: {1..9} for x ∈ {1..9}, y ∈ {1..9}

# every number occurs once in each row
∀ i ∈ {1..9}.∀ y ∈ {1..9}.∃ x ∈ {1..9}. field[x,y] = i
# every number occurs once in each column
∀ i ∈ {1..9}.∀ x ∈ {1..9}.∃ y ∈ {1..9}. field[x,y] = i
# every number occurs once in each block
∀ i ∈ {1..9}.∀ x1 ∈ {0..2}.∀ y1 ∈ {0..2}.∃x2 ∈ {1..3}.∃y2 ∈ {1..3}.
  field[3*x1+x2,3*y1+y2] = i

# the clues are satisfied
∀ x ∈ {1..9}.∀ y ∈ {1..9} with clues[x,y] != 0. field[x,y] = clues[x,y]

===============

for y := 1 to 9 do {
  for x := 1 to 9 do {
    print(field[x,y], ' ')
    if x % 3 = 0 ∧ x != 9 then print('| ')
  }
  println()
  if y % 3 = 0 ∧ y != 9 then println('------+-------+------')
}
```

New programming languages! (Or libraries)

```python
from z3 import *

# 9x9 matrix of integer variables
X = [ [ Int("x_%s_%s" % (i+1, j+1)) for j in range(9) ] for i in range(9) ]

# each cell contains a value in {1, ..., 9}
cells_c  = [ And(1 <= X[i][j], X[i][j] <= 9) for i in range(9) for j in range(9) ]

# each row contains a digit at most once
rows_c   = [ Distinct(X[i]) for i in range(9) ]

# each column contains a digit at most once
cols_c   = [ Distinct([ X[i][j] for i in range(9) ]) for j in range(9) ]

# each 3x3 square contains a digit at most once
sq_c     = [ Distinct([ X[3*i0 + i][3*j0 + j] for i in range(3) for j in range(3) ])
             for i0 in range(3) for j0 in range(3) ]

sudoku_c = cells_c + rows_c + cols_c + sq_c

# sudoku instance, we use '0' for empty cells
instance = ((0,0,0,0,9,4,0,3,0),
            (0,0,0,5,1,0,0,0,7),
            (0,8,9,0,0,0,0,4,0),
            (0,0,0,0,0,0,2,0,8),
            (0,6,0,2,0,1,0,5,0),
            (1,0,2,0,0,0,0,0,0),
            (0,7,0,0,0,0,5,2,0),
            (9,0,0,0,6,5,0,0,0),
            (0,4,0,9,7,0,0,0,0))

instance_c = [ If(instance[i][j] == 0, True, X[i][j] == instance[i][j])
               for i in range(9) for j in range(9) ]

s = Solver()
s.add(sudoku_c + instance_c)
if s.check() == sat:
    m = s.model()
    r = [ [ m.evaluate(X[i][j]) for j in range(9) ] for i in range(9) ]
    print_matrix(r)
else:
    print ("failed to solve")
```

Topics

Topics

- How to use automatic solvers

Topics

- How to use automatic solvers

- How do these automatic solvers work

Topics

- How to use automatic solvers 
- How do these automatic solvers work 
- Automatic reasoning for predicate/equational logic 

Topics

- How to use automatic solvers

- How do these automatic solvers work

- Automatic reasoning for predicate/equational logic

- Boolean functions, and having fun with SAT/SMT

Example: (free after Lewis Carroll)

Example: (free after Lewis Carroll)

1. Good-natured tenured professors are dynamic.
2. Grumpy student advisors play slot machines.
3. Smokers wearing a cap are phlegmatic.
4. Comical student advisors are professors.
5. Smoking untenured members are nervous.
6. Phlegmatic tenured members wearing caps are comical.
7. Student advisors who are not stock market players are scholars.
8. Relaxed student advisors are creative.
9. Creative scholars who do not play slot machines wear caps.
10. Nervous smokers play slot machines.
11. Student advisors who play slot machines do not smoke.
12. Creative good-natured stock market players wear caps.

Example: (free after Lewis Carroll)

1. Good-natured tenured professors are dynamic.
2. Grumpy student advisors play slot machines.
3. Smokers wearing a cap are phlegmatic.
4. Comical student advisors are professors.
5. Smoking untenured members are nervous.
6. Phlegmatic tenured members wearing caps are comical.
7. Student advisors who are not stock market players are scholars.
8. Relaxed student advisors are creative.
9. Creative scholars who do not play slot machines wear caps.
10. Nervous smokers play slot machines.
11. Student advisors who play slot machines do not smoke.
12. Creative good-natured stock market players wear caps.

13. Therefore no student advisor is smoking.

Automating logic

Idea:

Automating logic

Idea:

- Make statements formal

Automating logic

Idea:

- Make statements formal
- Give formal statements to computer

Automating logic

Idea:

- Make statements formal
- Give formal statements to computer

Steps:

Automating logic

Idea:

- Make statements formal
- Give formal statements to computer

Steps:

- name every “basic” notion

Automating logic

Idea:

- Make statements formal
- Give formal statements to computer

Steps:

- name every “basic” notion
- transform claims to formulas over these names

Naming basic notions

name	meaning	opposite
A	good-natured	grumpy
B	tenured	
C	professor	
D	dynamic	phlegmatic
E	wearing a cap	
F	smoke	
G	comical	
H	relaxed	nervous
I	play stock market	
J	scholar	
K	creative	
L	plays slot machine	
M	student advisor	

Transforming *claims* to *propositions*

name	meaning	opposite
A	good-natured	grumpy
B	tenured	
C	professor	
D	dynamic	phlegmatic
	$\ldots$	

good-natured tenured professors are dynamic

Transforming *claims* to *propositions*

name	meaning	opposite
A	good-natured	grumpy
B	tenured	
C	professor	
D	dynamic	phlegmatic
	...	

good-natured tenured professors are dynamic

$$(A \wedge B \wedge C) \rightarrow D$$

Transforming *claims* to *propositions*

name	meaning	opposite
A	good-natured	grumpy
L	plays slot machine	
M	student advisor	
	$\ldots$	

grumpy student advisors play slot machines.

Transforming *claims* to *propositions*

name	meaning	opposite
A	good-natured	grumpy
L	plays slot machine	
M	student advisor	
	$\dots$	

grumpy student advisors play slot machines.

$$(\neg A \land M) \rightarrow L$$

Transforming *claims* to *propositions*

name	meaning	opposite
B	tenured	
F	smoke	
H	relaxed	nervous
	...	

smoking untenured members are nervous.

Transforming *claims* to *propositions*

name	meaning	opposite
B	tenured	
F	smoke	
H	relaxed	nervous
	...	

smoking untenured members are nervous.

$$(F \wedge \neg B) \rightarrow \neg H$$

Translating the full puzzle

1. $(A \wedge B \wedge C) \rightarrow D$
2. $(\neg A \wedge M) \rightarrow L$
3. $(F \wedge E) \rightarrow \neg D$
4. $(G \wedge M) \rightarrow C$
5. $(F \wedge \neg B) \rightarrow \neg H$
6. $(\neg D \wedge B \wedge E) \rightarrow G$
7. $(\neg I \wedge M) \rightarrow J$
8. $(H \wedge M) \rightarrow K$
9. $(K \wedge J \wedge \neg L) \rightarrow E$
10. $(\neg H \wedge F) \rightarrow L$
11. $(L \wedge M) \rightarrow \neg F$
12. $(K \wedge A \wedge I) \rightarrow E$

Translating the full puzzle

1. $(A \wedge B \wedge C) \rightarrow D$
2. $(\neg A \wedge M) \rightarrow L$
3. $(F \wedge E) \rightarrow \neg D$
4. $(G \wedge M) \rightarrow C$
5. $(F \wedge \neg B) \rightarrow \neg H$
6. $(\neg D \wedge B \wedge E) \rightarrow G$
7. $(\neg I \wedge M) \rightarrow J$
8. $(H \wedge M) \rightarrow K$
9. $(K \wedge J \wedge \neg L) \rightarrow E$
10. $(\neg H \wedge F) \rightarrow L$
11. $(L \wedge M) \rightarrow \neg F$
12. $(K \wedge A \wedge I) \rightarrow E$
13. Then $\neg(M \wedge F)$

Translating the full puzzle

$$
\begin{aligned}
&(((A \land B \land C) \to D) \;\land \\
&((\neg A \land M) \to L) \;\land \\
&((F \land E) \to \neg D) \;\land \\
&((G \land M) \to C) \;\land \\
&((F \land \neg B) \to \neg H) \;\land \\
&((\neg D \land B \land E) \to G) \;\land \\
&((\neg I \land M) \to J) \;\land \\
&((H \land M) \to K) \;\land \\
&((K \land J \land \neg L) \to E) \;\land \\
&((\neg H \land F) \to L) \;\land \\
&((L \land M) \to \neg F) \;\land \\
&((K \land A \land I) \to E)\;) \quad \to \quad \neg(M \land F)
\end{aligned}
$$

Translating the full puzzle

$$
\begin{aligned}
&(((A \wedge B \wedge C) \rightarrow D) \ \wedge \\
&((\neg A \wedge M) \rightarrow L) \ \wedge \\
&((F \wedge E) \rightarrow \neg D) \ \wedge \\
&((G \wedge M) \rightarrow C) \ \wedge \\
&((F \wedge \neg B) \rightarrow \neg H) \ \wedge \\
&((\neg D \wedge B \wedge E) \rightarrow G) \ \wedge \\
&((\neg I \wedge M) \rightarrow J) \ \wedge \\
&((H \wedge M) \rightarrow K) \ \wedge \\
&((K \wedge J \wedge \neg L) \rightarrow E) \ \wedge \\
&((\neg H \wedge F) \rightarrow L) \ \wedge \\
&((L \wedge M) \rightarrow \neg F) \ \wedge \\
&((K \wedge A \wedge I) \rightarrow E)\ ) \quad \rightarrow \quad \neg(M \wedge F)
\end{aligned}
$$

Goal: is this a tautology?

Truth tables

Truth tables

Idea: compute all possible valuations

Truth tables

Idea: compute all possible valuations

A	B	C	D	E	F	G	H	I	J	K	L	M	result
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1

. . .

Truth tables

Idea: compute all possible valuations

A	B	C	D	E	F	G	H	I	J	K	L	M	result
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1

. . .

Number of rows:

Truth tables

Idea: compute all possible valuations

A	B	C	D	E	F	G	H	I	J	K	L	M	result
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1

. . .

Number of rows: 2^n

Truth tables

Idea: compute all possible valuations

A	B	C	D	E	F	G	H	I	J	K	L	M	result
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1

. . .

Number of rows: 2^n

Observation: if φ always evaluates to *true*, then $\neg\varphi$ always evaluates to *false*.

Truth tables

Idea: compute all possible valuations

A	B	C	D	E	F	G	H	I	J	K	L	M	result
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1

. . .

Number of rows: 2^n

Observation: if φ always evaluates to *true*, then $\neg\varphi$ always evaluates to *false*.

Definition

A formula is **satisfiable** if it yields *true* for some valuations.

SAT

Propositional formula: a formula made from boolean variables and $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$.

SAT

Propositional formula: a formula made from boolean variables and $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$.

SAT(isfiability): the problem whether or not a given propositional formula is satisfiable.

SAT

Propositional formula: a formula made from boolean variables and $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$.

SAT(isfiability): the problem whether or not a given propositional formula is satisfiable.

Practical problems: often thousands of variables.

SAT

Propositional formula: a formula made from boolean variables and $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$.

SAT(isfiability): the problem whether or not a given propositional formula is satisfiable.

Practical problems: often thousands of variables.

$\Longrightarrow$ we need to do better than truth tables.

Complexity of algorithms

- (time) complexity: the number of steps required for executing an algorithm

Complexity of algorithms

- (time) complexity: the number of steps required for executing an algorithm
- space complexity: the amount of memory required for executing an algorithm

Complexity of algorithms

- (time) complexity: the number of steps required for executing an algorithm
- space complexity: the amount of memory required for executing an algorithm

Basic observation: (time) complexity $\geq$ space complexity

Complexity of algorithms

- (time) complexity: the number of steps required for executing an algorithm
- space complexity: the amount of memory required for executing an algorithm

Basic observation: (time) complexity $\geq$ space complexity

Big-O notation: $f(n) = O(g(n))$.

Complexity of algorithms

- (time) complexity: the number of steps required for executing an algorithm
- space complexity: the amount of memory required for executing an algorithm

Basic observation: (time) complexity $\geq$ space complexity

Big-O notation: $f(n) = O(g(n))$.

Big-Omega notation: $f(n) = \Omega(g(n))$:

Typical use of big-O notation

- $g(n) = n$ (**linear**)
- $g(n) = n^2$ (**quadratic**)
- $g(n) = n^k$ for some k (**polynomial**)
- $g(n) = a^n$ for some $a > 1$ (**exponential**)

No polynomial algorithm is known for SAT.

No polynomial algorithm is known for SAT.

That is: there is no algorithm:

No polynomial algorithm is known for SAT.

That is: there is no algorithm:

- given: arbitrary φ

No polynomial algorithm is known for SAT.

That is: there is no algorithm:

- given: arbitrary φ
- returns: whether or not φ is satisfiable

No polynomial algorithm is known for SAT.

That is: there is no algorithm:

- given: arbitrary φ
- returns: whether or not φ is satisfiable
- in time $\leq c * |\varphi|^k$ for some fixed c

No polynomial algorithm is known for SAT.

That is: there is no algorithm:

- given: arbitrary φ
- returns: whether or not φ is satisfiable
- in time $\leq c * |\varphi|^k$ for some fixed c

This can mean two different things!

P versus NP

P: the class of decision problems admitting a polynomial algorithm.

P versus NP

P: the class of decision problems admitting a polynomial algorithm.

NP: the class of decision problem which can be verified through a polynomial algorithm.

P versus NP

P: the class of decision problems admitting a polynomial algorithm.

NP: the class of decision problem which can be verified through a polynomial algorithm.

Conjecture: SAT is not in **P**.

P versus NP

P: the class of decision problems admitting a polynomial algorithm.

NP: the class of decision problem which can be verified through a polynomial algorithm.

Conjecture: SAT is not in **P**.

Conjecture: **P** $\neq$ **NP**

NP-Completeness

NP-Completeness

NP-complete: a decision problem that is "maximally hard".

NP-Completeness

NP-complete: a decision problem that is “maximally hard”.

SAT is NP-complete.

NP-Completeness

NP-complete: a decision problem that is “maximally hard”.

SAT is NP-complete.

Other NP-complete problems:

NP-Completeness

NP-complete: a decision problem that is “maximally hard”.

SAT is NP-complete.

Other NP-complete problems:

- Traveling Salesman

NP-Completeness

NP-complete: a decision problem that is “maximally hard”.

SAT is NP-complete.

Other NP-complete problems:

- Traveling Salesman
- Hamiltonian Circuit

NP-Completeness

NP-complete: a decision problem that is “maximally hard”.

SAT is NP-complete.

Other NP-complete problems:

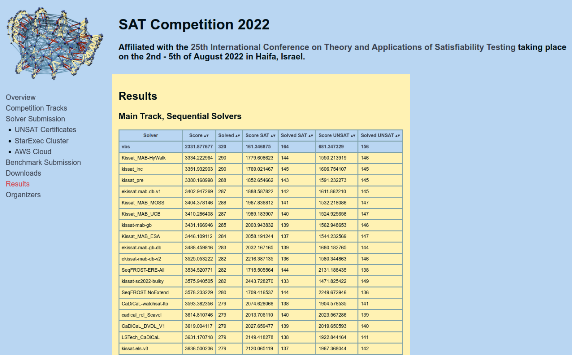
- Traveling Salesman
- Hamiltonian Circuit
- Knapsack

SAT solvers

Despite the inherent difficulty, many good **SAT solvers** exist!

SAT solvers

Despite the inherent difficulty, many good **SAT solvers** exist!



SAT Competition 2022

Affiliated with the 25th International Conference on Theory and Applications of Satisfiability Testing taking place on the 2nd - 5th of August 2022 in Haifa, Israel.

- Overview
- Competition Tracks
- Solver Submission
  - UNSAT Certificates
  - StarExec Cluster
  - AWS Cloud
- Benchmark Submission
- Downloads
- Results
- Organizers

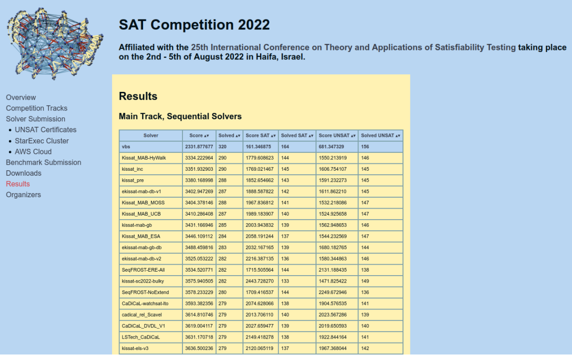
Results

Main Track, Sequential Solvers

Solver	Score ▴▾	Solved ▴▾	Score SAT ▴▾	Solved SAT ▴▾	Score UNSAT ▴▾	Solved UNSAT ▴▾
vbs	2331.877677	320	161.346875	164	681.347329	156
Kissat_MAB-HyWalk	3334.222964	290	1779.608623	144	1550.213919	146
kissat_inc	3351.932903	290	1769.021467	145	1606.754107	145
kissat_pre	3380.168998	288	1852.654662	143	1591.232273	145
ekissat-mab-db-v1	3402.947269	287	1888.587822	142	1611.862210	145
Kissat_MAB_MOSS	3404.378146	288	1967.836812	141	1532.218086	147
Kissat_MAB_UCB	3410.286408	287	1989.183907	140	1524.925658	147
kissat-mab-gb	3431.166946	285	2003.943832	139	1562.948653	146
Kissat_MAB_ESA	3446.109112	284	2058.191244	137	1544.232569	147
ekissat-mab-gb-db	3488.459816	283	2032.167165	139	1680.182765	144
ekissat-mab-db-v2	3525.053222	282	2216.387135	136	1580.344863	146
SeqFROST-ERE-All	3534.520771	282	1715.505564	144	2131.188435	138
kissat-sc2022-bulky	3575.940505	282	2443.728270	133	1471.825422	149
SeqFROST-NoExtend	3578.233229	280	1709.416537	144	2249.672946	136
CaDiCaL-watchsat-lto	3593.382356	279	2074.628066	138	1904.576535	141
cadical_rel_Scavel	3614.810746	279	2013.706110	140	2023.567286	139
CaDiCaL_DVDL_V1	3619.004117	279	2027.659477	139	2019.650593	140
LSTech_CaDiCaL	3631.170718	279	2149.418278	138	1922.844164	141
kissat-els-v3	3636.500236	279	2120.065119	137	1967.368044	142

SAT solvers

Despite the inherent difficulty, many good **SAT solvers** exist!



SAT Competition 2022

Affiliated with the 25th International Conference on Theory and Applications of Satisfiability Testing taking place on the the 2nd - 5th of August 2022 in Haifa, Israel.

Overview
Competition Tracks
Solver Submission
- UNSAT Certificates
- StarExec Cluster
- AWS Cloud

Benchmark Submission
Downloads
Results
Organizers

Results

Main Track, Sequential Solvers

Solver	Score ▴▾	Solved ▴▾	Score SAT ▴▾	Solved SAT ▴▾	Score UNSAT ▴▾	Solved UNSAT ▴▾
vbs	2331.877677	320	161.346875	164	681.347329	156
Kissat_MAB-HyWalk	3334.222964	290	1779.608623	144	1550.213919	146
kissat_inc	3351.932903	290	1769.021467	145	1606.754107	145
kissat_pre	3380.168998	288	1852.654662	143	1591.232273	145
ekissat-mab-db-v1	3402.947269	287	1888.587822	142	1611.862210	145
Kissat_MAB_MOSS	3404.378146	288	1967.836812	141	1532.218086	147
Kissat_MAB_UCB	3410.286408	287	1989.183907	140	1524.925658	147
kissat-mab-gb	3431.166946	285	2003.943832	139	1562.948653	146
Kissat_MAB_ESA	3446.109112	284	2058.191244	137	1544.232569	147
ekissat-mab-gb-db	3488.459816	283	2032.167165	139	1680.182765	144
ekissat-mab-db-v2	3525.053222	282	2216.387135	136	1580.344863	146
SeqFROST-ERE-All	3534.520771	282	1715.505564	144	2131.188435	138
kissat-sc2022-bulky	3575.940505	282	2443.728270	133	1471.825422	149
SeqFROST-NoExtend	3578.233229	280	1709.416537	144	2249.672946	136
CaDiCaL-watchsat-lto	3593.382356	279	2074.628066	138	1904.576535	141
cadical_rel_Scavel	3614.810746	279	2013.706110	140	2023.567286	139
CaDiCaL_DVDL_V1	3619.004117	279	2027.659477	139	2019.650593	140
LSTech_CaDiCaL	3631.170718	279	2149.418278	138	1922.844164	141
kissat-els-v3	3636.500236	279	2120.065119	137	1967.368044	142

Typical benchmarks have thousands of variables.

Many industrial problems are **within reach**.

Topics in the first month

- How to use automatic solvers

- How do these automatic solvers work

Topics in the first month

- Week 1 and 2: how to use SAT solvers

- Week 3 and 4: how do SAT solvers actually work

SAT solver input format

SAT solver input format

A **conjunctive normal form (CNF)** is a conjunction of clauses.
A **clause** is a disjunction of literals.
A **literal** is either a variable or the negation of a variable.

SAT solver input format

A **conjunctive normal form (CNF)** is a conjunction of clauses.
A **clause** is a disjunction of literals.
A **literal** is either a variable or the negation of a variable.

Hence a CNF is of the shape

$$\bigwedge_i (\bigvee_j \ell_{ij})$$

where ℓ_{ij} are literals.

SAT solver input format

A **conjunctive normal form (CNF)** is a conjunction of clauses.
A **clause** is a disjunction of literals.
A **literal** is either a variable or the negation of a variable.

Hence a CNF is of the shape

$$\bigwedge_i (\bigvee_j \ell_{ij})$$

where ℓ_{ij} are literals.

Here $\bigwedge_{i=1}^{n} A_i$ is an abbreviation for $A_1 \wedge A_2 \wedge A_3 \wedge \cdots \wedge A_n$

SAT solver input format

A **conjunctive normal form (CNF)** is a conjunction of clauses.
A **clause** is a disjunction of literals.
A **literal** is either a variable or the negation of a variable.

Hence a CNF is of the shape

$$\bigwedge_i \left(\bigvee_j \ell_{ij} \right)$$

where ℓ_{ij} are literals.

Here $\bigwedge_{i=1}^{n} A_i$ is an abbreviation for $A_1 \wedge A_2 \wedge A_3 \wedge \cdots \wedge A_n$

Proposition formulas can be transformed into CNF format with **linear** overhead.

Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

- 42437 literals (x_1 to x_{42437})

Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

- 42437 literals (x_1 to x_{42437})
- 150699 clauses

Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

- 42437 literals (x_1 to x_{42437})
- 150699 clauses
- all lines end with 0

Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

- 42437 literals (x_1 to x_{42437})
- 150699 clauses
- all lines end with 0
- minus is used for negated literals

Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

- 42437 literals (x_1 to x_{42437})
- 150699 clauses
- all lines end with 0
- minus is used for negated literals
- first clause: $\neg x_{4902} \vee x_{4903}$

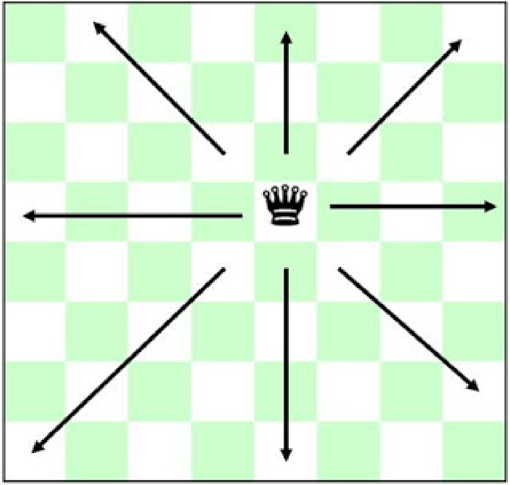
Example input file

```
p cnf 42437 150699
-4902 4903 0
-4904 4905 0
-4908 4909 0
-11 4911 0
-11 -12 4910 0
...
```

- 42437 literals (x_1 to x_{42437})
- 150699 clauses
- all lines end with 0
- minus is used for negated literals
- first clause: $\neg x_{4902} \vee x_{4903}$
- fifth clause: $\neg x_{11} \vee \neg x_{12} \vee x_{4910}$

Example: Eight Queens Problem

Example: Eight Queens Problem



Solving the Eight Queens Problem

Don't think about how to solve it, but only **specify** the problem.

Solving the Eight Queens Problem

Don't think about how to solve it, but only **specify** the problem.

For every position (y, x) on the board: boolean variable p_{yx} expresses whether there is a queen or not.

Solving the Eight Queens Problem

Don't think about how to solve it, but only **specify** the problem.

For every position (y, x) on the board: boolean variable p_{yx} expresses whether there is a queen or not.

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Eight queens requirements

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Eight queens requirements

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Note: If p_{yx} and $p_{y'x'}$ are in the same row: $y = y'$

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

At **least** one queen on row y:

$$p_{y1} \vee p_{y2} \vee p_{y3} \vee p_{y4} \vee p_{y5} \vee p_{y6} \vee p_{y7} \vee p_{y8}$$

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

At **least** one queen on row y:

$$p_{y1} \vee p_{y2} \vee p_{y3} \vee p_{y4} \vee p_{y5} \vee p_{y6} \vee p_{y7} \vee p_{y8}$$

Shorthand:

$$\bigvee_{j=1}^{8} p_{ij}$$

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

At **most** one queen on row y?

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

At **most** one queen on row y?

That is: for every $i < j$ not both p_{yi} and p_{yj} are true.

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

At **most** one queen on row y?

That is: for every $i < j$ not both p_{yi} and p_{yj} are true.

So $\neg p_{yi} \vee \neg p_{yj}$ for all $i < j$.

Eight queens requirements

Row y:

$$p_{y1}, p_{y2}, p_{y3}, p_{y4}, p_{y5}, p_{y6}, p_{y7}, p_{y8}$$

At **most** one queen on row y?

That is: for every $i < j$ not both p_{yi} and p_{yj} are true.

So $\neg p_{yi} \vee \neg p_{yj}$ for all $i < j$.

$$\bigwedge_{0<i<j\leq 8} (\neg p_{yi} \vee \neg p_{yj})$$

Eight queens requirements

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Eight queens requirements

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Column requirements for p_{yx}: the same as for rows with y, x swapped.

Eight queens requirements

Requirements until now:

Eight queens requirements

Requirements until now:

At least one queen on every row:

$$\bigwedge_{y=1}^{8} \bigvee_{x=1}^{8} p_{yx}$$

Eight queens requirements

Requirements until now:

At least one queen on every row:

$$\bigwedge_{y=1}^{8} \bigvee_{x=1}^{8} p_{yx}$$

At most one queen on every row:

$$\bigwedge_{y=1}^{8} \bigwedge_{0<i<j\leq 8} (\neg p_{yi} \vee \neg p_{yj})$$

Eight queens requirements

And similar for the columns:

Eight queens requirements

And similar for the columns:

At least one queen on every column:

$$\bigwedge_{x=1}^{8} \bigvee_{y=1}^{8} p_{yx}$$

Eight queens requirements

And similar for the columns:

At least one queen on every column:

$$\bigwedge_{x=1}^{8} \bigvee_{y=1}^{8} p_{yx}$$

At most one queen on every column:

$$\bigwedge_{x=1}^{8} \bigwedge_{0<i<j\leq 8} (\neg p_{ix} \vee \neg p_{jx})$$

Eight queens requirements

There is at most one queen on every diagonal:

Eight queens requirements

There is at most one queen on every diagonal:

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Eight queens requirements

There is at most one queen on every diagonal:

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

p_{yx} and $p_{y'x'}$ on such a diagonal

Eight queens requirements

There is at most one queen on every diagonal:

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

p_{yx} and $p_{y'x'}$ on such a diagonal

$$\Longleftrightarrow$$

$$y + x = y' + x'$$

Eight queens requirements

The diagonal in the other direction:

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

Eight queens requirements

The diagonal in the other direction:

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

p_{yx} and $p_{y'x'}$ on such a diagonal

Eight queens requirements

The diagonal in the other direction:

p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	p_{26}	p_{27}	p_{28}
p_{31}	p_{32}	p_{33}	p_{34}	p_{35}	p_{36}	p_{37}	p_{38}
p_{41}	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}
p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}
p_{61}	p_{62}	p_{63}	p_{64}	p_{65}	p_{66}	p_{67}	p_{68}
p_{71}	p_{72}	p_{73}	p_{74}	p_{75}	p_{76}	p_{77}	p_{78}
p_{81}	p_{82}	p_{83}	p_{84}	p_{85}	p_{86}	p_{87}	p_{88}

p_{yx} and $p_{y'x'}$ on such a diagonal

$$\iff$$

$$y - x = y' - x'$$

Eight queens requirements

So for all y, x, y', x' with $(y, x) \neq (y', x')$ satisfying $y + x = y' + x'$ or $y - x = y' - x'$:

$$\neg p_{yx} \vee \neg p_{y'x'}$$

Eight queens requirements

So for all y, x, y', x' with $(y, x) \neq (y', x')$ satisfying $y + x = y' + x'$ or $y - x = y' - x'$:

$$\neg p_{yx} \vee \neg p_{y'x'}$$

We may restrict to $y < y'$, yielding

Eight queens requirements

So for all y, x, y', x' with $(y, x) \neq (y', x')$ satisfying $y + x = y' + x'$ or $y - x = y' - x'$:

$$\neg p_{yx} \vee \neg p_{y'x'}$$

We may restrict to $y < y'$, yielding

$$\bigwedge_{0<y<y'\leq 8} \Bigl(\bigwedge_{x,x':y+x=y'+x' \vee y-x=y'-x'} \neg p_{yx} \vee \neg p_{y'x'} \Bigr)$$

Complete formula

$$\bigwedge_{y=1}^{8} \bigvee_{x=1}^{8} p_{yx} \wedge$$

$$\bigwedge_{y=1}^{8} \bigwedge_{0<i<j\leq 8} (\neg p_{yi} \vee \neg p_{yj}) \wedge$$

$$\bigwedge_{x=1}^{8} \bigvee_{y=1}^{8} p_{yx} \wedge$$

$$\bigwedge_{x=1}^{8} \bigwedge_{0<i<j\leq 8} (\neg p_{ix} \vee \neg p_{jx}) \wedge$$

$$\bigwedge_{0<y<y'\leq 8} \Big(\bigwedge_{x,x' \text{ with } y+x=y'+x' \vee y-x=y'-x'} \neg p_{yx} \vee \neg p_{y'x'} \Big)$$

Conclusion

The resulting formula

Conclusion

The resulting formula

- is easy to generate

Conclusion

The resulting formula

- is easy to generate
- consists of 740 clauses

Conclusion

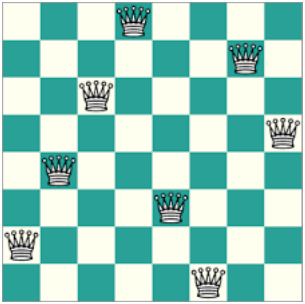
The resulting formula

- is easy to generate
- consists of 740 clauses
- is easily solved by a SAT solver

Conclusion

The resulting formula

- is easy to generate
- consists of 740 clauses
- is easily solved by a SAT solver



Generalising the Eight Queens problem

To find all 92 solutions, we use the following trick:

Generalising the Eight Queens problem

To find all 92 solutions, we use the following trick:

Find a solution, then add its negation to the formula. Repeat this until the formula is unsatisfiable.

Generalising the Eight Queens problem

To find all 92 solutions, we use the following trick:

Find a solution, then add its negation to the formula. Repeat this until the formula is unsatisfiable.

The problem can be generalised to n queens on $n \times n$ board.

Generalising the Eight Queens problem

To find all 92 solutions, we use the following trick:

Find a solution, then add its negation to the formula. Repeat this until the formula is unsatisfiable.

The problem can be generalised to n queens on $n \times n$ board.

For $n = 100$ Z3 finds a satisfying assignment of the 50Mb formula within 10 seconds.

Generalising the Eight Queens problem

To find all 92 solutions, we use the following trick:

Find a solution, then add its negation to the formula. Repeat this until the formula is unsatisfiable.

The problem can be generalised to n queens on $n \times n$ board.

For $n = 100$ Z3 finds a satisfying assignment of the 50Mb formula within 10 seconds.

The same approach is applicable to extending a partially filled board, which is an NP-complete problem.

Optimising our encoding?

Note that optimisations are possible!

Optimising our encoding?

Note that optimisations are possible!

- there is **at least** one queen in every row
- there is **at most** one queen in every row
- there is **at least** one queen in every column
- there is **at most** one queen in every column

Optimising our encoding?

Note that optimisations are possible!

- there is **at least** one queen in every row
- there is **at most** one queen in every row
- there is **at least** one queen in every column
- there is **at most** one queen in every column

We can choose any one of those to leave out!

Optimising our encoding?

Note that optimisations are possible!

- there is **at least** one queen in every row
- there is **at most** one queen in every row
- there is **at least** one queen in every column
- there is **at most** one queen in every column

We can choose any one of those to leave out!

In practice: this often makes no difference, or even makes things worse.

Optimising our encoding?

Note that optimisations are possible!

- there is **at least** one queen in every row
- there is **at most** one queen in every row
- there is **at least** one queen in every column
- there is **at most** one queen in every column

We can choose any one of those to leave out!

In practice: this often makes no difference, or even makes things worse.

(But: experimenting can be worthwhile.)

Eight Queens: conclusion

To conclude: we expressed a chess board problem in a pure SAT problem.

Eight Queens: conclusion

To conclude: we expressed a chess board problem in a pure SAT problem.

It is easy to generate this formula in the appropriate syntax by a small program.

Eight Queens: conclusion

To conclude: we expressed a chess board problem in a pure SAT problem.

It is easy to generate this formula in the appropriate syntax by a small program.

Then a SAT solver immediately finds a solution.

Class exercise: Sudoku

2				8	3	4	7	6
1			9					
6					4			
5		6					9	
8								4
	7					5		1
			4					9
					8			2
9	1	4	5	2				3

Class exercise: Sudoku

2				8	3	4	7	6
1			9					
6					4			
5		6					9	
8								4
	7					5		1
			4					9
					8			2
9	1	4	5	2				3

Despite apparent numbers, this can be done in pure SAT: only boolean variables!

Class exercise: Sudoku

2				8	3	4	7	6
1			9					
6					4			
5		6					9	
8								4
	7					5		1
			4					9
					8			2
9	1	4	5	2				3

Despite apparent numbers, this can be done in pure SAT: only boolean variables!

For every (y, x) and number k: boolean variable $s_{y,x,k}$.

Requirements

- each position has exactly one of the numbers 1–9
- in each row: every number in 1–9 occurs exactly once
- in each column: every number in 1–9 occurs exactly once
- in each block: every number in 1–9 occurs exactly once

Requirements

- each position has exactly one of the numbers 1–9
- in each row: every number in 1–9 occurs exactly once
- in each column: every number in 1–9 occurs exactly once
- in each block: every number in 1–9 occurs exactly once

Requirements

- each position has at most one of the numbers 1–9
- in each row: every number in 1–9 occurs at least once
- in each column: every number in 1–9 occurs at least once
- in each block: every number in 1–9 occurs at least once

Requirements

- each position has at most one of the numbers 1–9

$$\bigwedge_{y=1}^{9} \bigwedge_{x=1}^{9} \bigwedge_{1 \leq i < j \leq 9} \neg s_{y,x,i} \vee \neg s_{y,x,j}$$

- in each row: every number in 1–9 occurs at least once

- in each column: every number in 1–9 occurs at least once

- in each block: every number in 1–9 occurs at least once

Requirements

- each position has at most one of the numbers 1–9

$$\bigwedge_{y=1}^{9} \bigwedge_{x=1}^{9} \bigwedge_{1 \leq i < j \leq 9} \neg s_{y,x,i} \vee \neg s_{y,x,j}$$

- in each row: every number in 1–9 occurs at least once

$$\bigwedge_{y=1}^{9} \bigwedge_{i=1}^{9} \bigvee_{x=1}^{9} s_{y,x,i}$$

- in each column: every number in 1–9 occurs at least once

- in each block: every number in 1–9 occurs at least once

Requirements

- each position has at most one of the numbers 1–9

$$\bigwedge_{y=1}^{9} \bigwedge_{x=1}^{9} \bigwedge_{1 \leq i < j \leq 9} \neg s_{y,x,i} \vee \neg s_{y,x,j}$$

- in each row: every number in 1–9 occurs at least once

$$\bigwedge_{y=1}^{9} \bigwedge_{i=1}^{9} \bigvee_{x=1}^{9} s_{y,x,i}$$

- in each column: every number in 1–9 occurs at least once

$$\bigwedge_{x=1}^{9} \bigwedge_{i=1}^{9} \bigvee_{y=1}^{9} s_{y,x,i}$$

- in each block: every number in 1–9 occurs at least once

Requirements

- each position has at most one of the numbers 1–9

$$\bigwedge_{y=1}^{9} \bigwedge_{x=1}^{9} \bigwedge_{1 \leq i < j \leq 9} \neg s_{y,x,i} \vee \neg s_{y,x,j}$$

- in each row: every number in 1–9 occurs at least once

$$\bigwedge_{y=1}^{9} \bigwedge_{i=1}^{9} \bigvee_{x=1}^{9} s_{y,x,i}$$

- in each column: every number in 1–9 occurs at least once
- in each block: every number in 1–9 occurs at least once

$$\bigwedge_{y_1=0}^{2} \bigwedge_{x_1=0}^{2} \bigwedge_{i=1}^{9} \bigvee_{y_2=1}^{3} \bigvee_{x_2=1}^{3} s_{y_1*3+y_2,x_1*3+x_2,i}$$

An amazing example from academia

An amazing example from academia

Prove termination of the system of three rules

$$aa \to bc, \quad bb \to ac, \quad cc \to ab$$

An amazing example from academia

Prove termination of the system of three rules

$$aa \to bc, \quad bb \to ac, \quad cc \to ab$$

Solution: matrix interpretations.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} >^{ij}$$

$$\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} * \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{pmatrix}$$

An amazing example from academia

Prove termination of the system of three rules

$$aa \to bc, \quad bb \to ac, \quad cc \to ab$$

Solution: matrix interpretations.

Approach: encode matrix components as binary numbers, and send to a SAT solver.

An amazing example from academia

Prove termination of the system of three rules

$$aa \to bc, \quad bb \to ac, \quad cc \to ab$$

Solution: matrix interpretations.

Approach: encode matrix components as binary numbers, and send to a SAT solver.

No human intuition is available.

An example from education

An example from education

In the course **Software Development Entrepreneurship** students work together in groups of 4–5.

An example from education

In the course **Software Development Entrepreneurship** students work together in groups of 4–5.

- Group members share a 4-week available slot each week.

An example from education

In the course **Software Development Entrepreneurship** students work together in groups of 4–5.

- Group members share a 4-week available slot each week.
- Business students per group $\in \{1, 2\}$, confident programmers ≥ 2.

An example from education

In the course **Software Development Entrepreneurship** students work together in groups of 4–5.

- Group members share a 4-week available slot each week.
- Business students per group $\in \{1, 2\}$, confident programmers ≥ 2.
- At least 1 idea per group, at most 2.

An example from education

In the course **Software Development Entrepreneurship** students work together in groups of 4–5.

- Group members share a 4-week available slot each week.
- Business students per group $\in \{1, 2\}$, confident programmers ≥ 2.
- At least 1 idea per group, at most 2.
- Partner preferences (and anti-preference) should be considered.

An example from education

In the course **Software Development Entrepreneurship** students work together in groups of 4–5.

- Group members share a 4-week available slot each week.
- Business students per group $\in \{1, 2\}$, confident programmers ≥ 2.
- At least 1 idea per group, at most 2.
- Partner preferences (and anti-preference) should be considered.
- Individual wishes like gender balance.

An example from sports:

An example from sports:

We wish to make **fight assignments** for a martial arts tournament.

An example from sports:

We wish to make **fight assignments** for a martial arts tournament.

- 9 participants, 4 matches pp

An example from sports:

We wish to make **fight assignments** for a martial arts tournament.

- 9 participants, 4 matches pp
- every opponent at most once

An example from sports:

We wish to make **fight assignments** for a martial arts tournament.

- 9 participants, 4 matches pp
- every opponent at most once
- at least two matches rest between fights

An example from sports:

We wish to make **fight assignments** for a martial arts tournament.

- 9 participants, 4 matches pp
- every opponent at most once
- at least two matches rest between fights
- Everyone has similar total difficulty.

An example from sports:

We wish to make **fight assignments** for a martial arts tournament.

- 9 participants, 4 matches pp
- every opponent at most once
- at least two matches rest between fights
- Everyone has similar total difficulty.
- if there are no surprises, the top 4 should be the best 4.

Example: verification of a microprocessor

Example: verification of a microprocessor

Goal:

$$\neg(\text{specified behavior} \leftrightarrow \text{actual behavior})$$

is not satisfiable.

Extensions

Note: propositional logic is not enough for everything.

Other topics in this course:

Extensions

Note: propositional logic is not enough for everything.
Other topics in this course:

- SMT: Satisfiability Modulo Theories.
  $3a + 4b - c \leq 17$

Extensions

Note: propositional logic is not enough for everything.
Other topics in this course:

- SMT: Satisfiability Modulo Theories.
  $3a + 4b - c \leq 17$
- Predicate logic: reasoning with $\forall$ and $\exists$
  all men are mortal
  Socrates is a man
  hence Socrates is mortal

Extensions

Note: propositional logic is not enough for everything.
Other topics in this course:

- SMT: Satisfiability Modulo Theories.
  $3a + 4b - c \leq 17$

- Predicate logic: reasoning with $\forall$ and $\exists$
  all men are mortal
  Socrates is a man
  hence Socrates is mortal

- Equational logic: reasoning with equalities
  given: $0 + x = x$ *and*
  $(x + 1) + y = (x + y) + 1$
  conclude: $(0 + 1 + 1) + (0 + 1 + 1) = 0 + 1 + 1 + 1 + 1.$

Practical assignment

Practical assignment

- All doable with plain SAT!

Practical assignment

- All doable with plain SAT!
- But: that doesn't mean it's the best idea. Full capacity of Z3 may be used.

Practical assignment

- All doable with plain SAT!
- But: that doesn't mean it's the best idea. Full capacity of Z3 may be used.
- Advice: do **not** try to find solutions in esoteric Z3 abilities.

Practical assignment

- All doable with plain SAT!
- But: that doesn't mean it's the best idea. Full capacity of Z3 may be used.
- Advice: do **not** try to find solutions in esoteric Z3 abilities.
- Do not write SAT input files by hand (other than as a **small** test). Generate them, or use a library.

Practical assignment

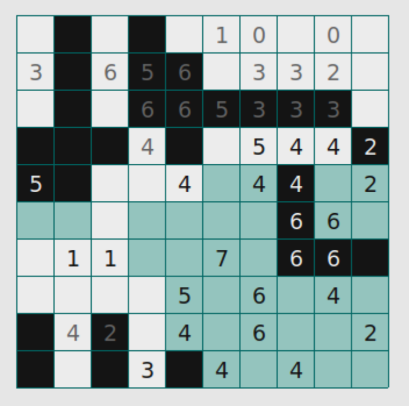
- All doable with plain SAT!
- But: that doesn't mean it's the best idea. Full capacity of Z3 may be used.
- Advice: do **not** try to find solutions in esoteric Z3 abilities.
- Do not write SAT input files by hand (other than as a **small** test). Generate them, or use a library.
- In the document you hand in, use clear notation, such as

$$\bigvee_{i=1}^{n} A_i \quad \text{for } A_1 \vee \cdots \vee A_n$$

and

$$\bigwedge_{P(i)} A_i \quad \text{for the conjunction of } A_i \text{ for all } i \text{ that satisfy } P(i)$$

Class exercise



Quiz

1. Indicate for each of the following statements whether it is **true**, **untrue** or **unknown**.
   1.1 SAT is in P.
   1.2 SAT is in NP.
   1.3 SAT is NP-complete.
   1.4 P $\neq$ NP.

2. How can you use a SAT solver to prove that the statement

$$\textit{if } A \to B \textit{ and } B \to C \textit{ then } A \to C$$

is a tautology?