DM573
Introduction to Computer Science
Lecture on Satisfiability

Peter Schneider-Kamp

petersk@imada.sdu.dk

http://imada.sdu.dk/~petersk/

DM573
Introduction to Computer Science
Lecture on Satisfiability

Schneider-Kamp

Anton D. Lautrup
lautrup@imada.sdu.dk

http://[illegible]sk/

Satisfiability

- $X + 3 = Y$

Is this equation valid for any values of X and Y?

Eq. is satisfiable

Satisfiability

- X = X + 1

Satisfiable?

”Solution”:

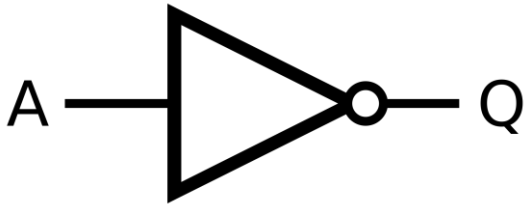
X can take boolean assignments.

Satisfiability is a semantic property

\+ can be the conjunction ∧

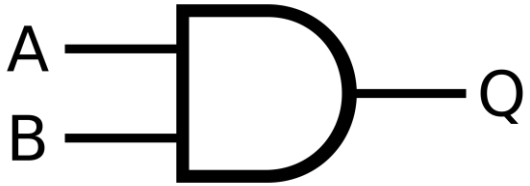
THE SAT PROBLEM

Logic gates



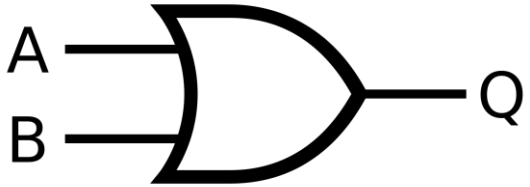
- ! / ¬ / -

A	Q
0	I
I	0



- && / ∧

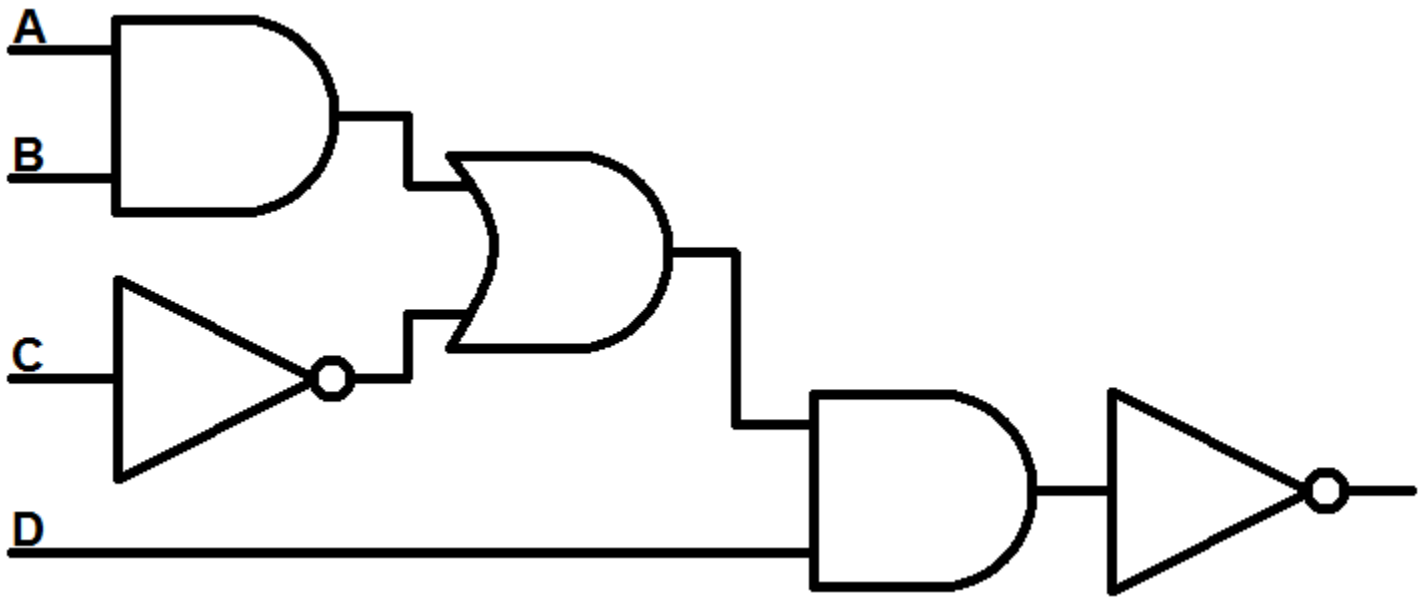
A	B	Q
0	0	0
0	I	0
I	0	0
I	I	I



- || / ∨

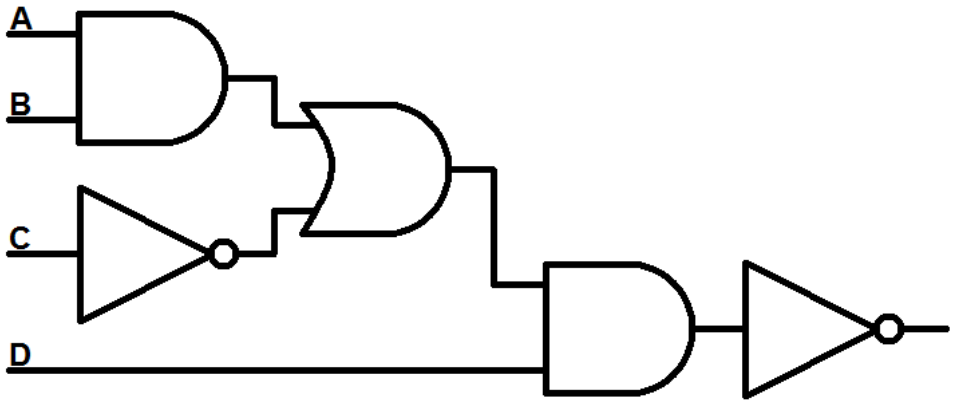
A	B	Q
0	0	0
0	I	I
I	0	I
I	I	I

Logic gates



$$\neg\Big(\big((\boldsymbol{A} \wedge \boldsymbol{B}) \vee \neg \boldsymbol{C}\big) \wedge \boldsymbol{D}\Big)$$

Logic gates



2: 4

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

V

1: 2

C	Q
0	1
1	0

→

3: 8

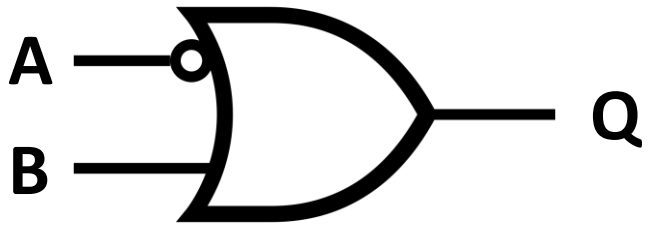
A	B	C	Q
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

etc...

New gate: Implication

(Material conditional)

- **A → B**



A	B	Q
0	0	I
0	I	I
I	0	0
I	I	I

A B

CLAIM: If it rains, I will bring an umbrella

It is a lie, only if it rains, and I bring no umbrella

Implication is a composite operator

A → B

A	B	Q
0	0	1
0	1	1
1	0	0
1	1	1

→

¬ A

A	Q
0	1
1	0

+

A ∨ B

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

I tell the truth:

- When it is NOT raining
- OR when I bring my umbrella

¬ A ∨ B

DM549: Propositional Variables

- Variable that can be either *false* or *true*
- Set P of **propositional variables**
- Example:

  $P = \{A,B,C,D,X, Y, Z, X_1, X_2, X_3, \ldots\}$

- A **variable assignment** is an assignment of the values *false* and *true* to all variables in P
- Example:

  X = *true*

  Y = *false*

  Z = *true*

DM549: Propositional Formulas

- **Propositional formulas**
  - If X in P, then X is a formula.
  - If F is a formula, then ¬F is a formula.
  - If F and G are formulas, then F ∧ G is a formula.
  - If F and G are formulas, then F ∨ G is a formula.
  - If F and G are formulas, then F ➔ G is a formula.
- Example: (X ➔ (Y ∧ ¬Z))

- Propositional variables or negated propositional variables are called **literals**
- Example: X, ¬X

Which formulas are satisfiable?

- X
- $\neg X$
- $X \wedge \neg X$
- $\neg X \wedge \neg X$
- $X \vee \neg X$

- $X_1 \rightarrow X_2$
- $\neg X_1 \vee X_2$
- ...

- True when X=T
- True when X=F
- Unsatisfiable
- True when X=F
- True when X=T or X=F

- True when X_1=T and X_2=T
- True when X_1=T and X_2=T

Satisfiability

- Variable assignment V **satisfies** formulas as follows:
  - V satisfies X in P iff V assigns X = *true*
  - V satisfies ¬F iff V does not satisfy F
  - V satisfies F∧ G iff V satisfies both F and G
  - V satisfies F∨ G iff V satisfies at least one of F and G
  - V satisfies F ➔ G iff V does not satisfy F or V satisfies G

- A propositional formula F is **satisfiable** iff there is a variable assignment V such that V satisfies F.

Satisfiability

The Satisfiability Problem of Propositional Logic (**SAT**):

- *Given a formula F, decide whether it is satisfiable.*

Modelling Satisfiability

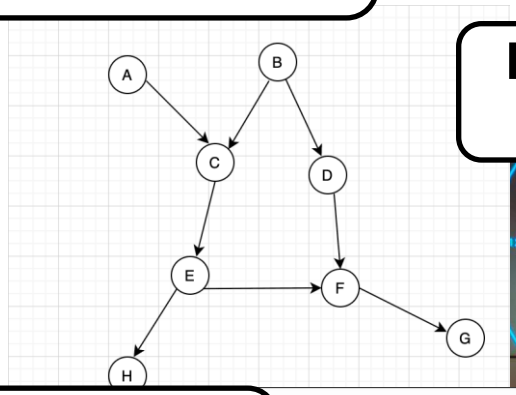
- propositional variables are basically bits
- model your problem by bits
- model the relation of the bits by a propositional formula
- solve the SAT problem to solve your problem

Your thoughts:
What problems can we model?

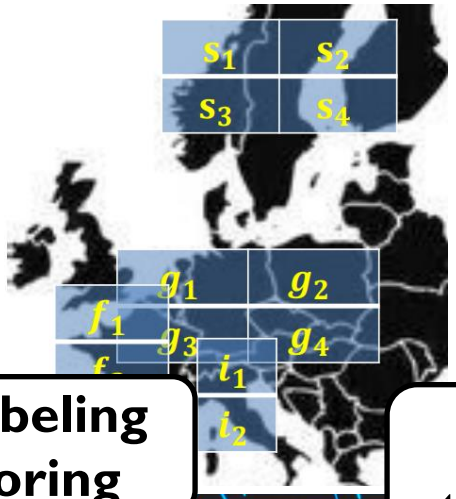
Go to www.menti.com and use the code 8502 6148

Problems to model using SAT

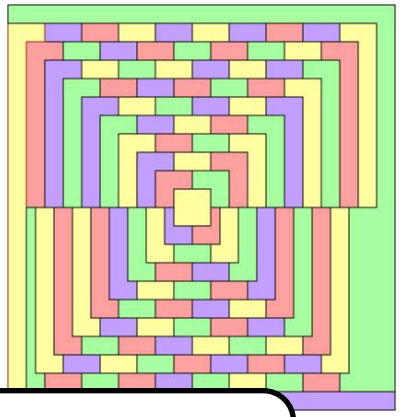
Checking dependencies

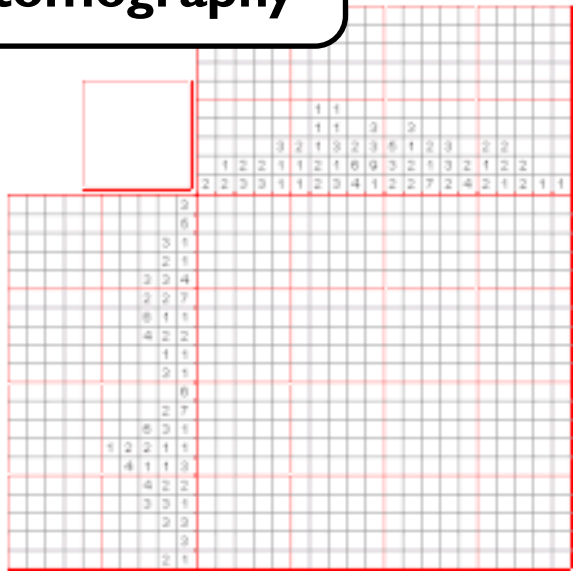


Map labeling & coloring



Discrete tomography





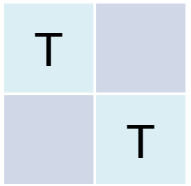
Configurable products

Modelling

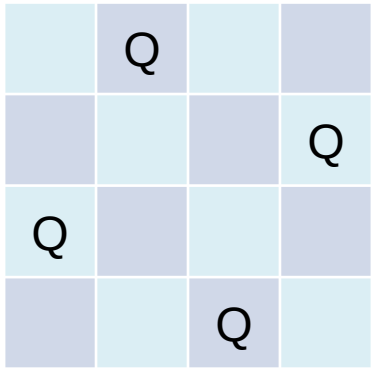
N-TOWERS & N-QUEENS

N-Towers & N-Queens

- N-Towers
  - How to place N towers on an NxN chessboard such that they do not attack each other?
  - (Towers attack horizontally and vertically.)



- N-Queens (restriction of N-Towers)
  - How to place N queens on an NxN chessboard such that they do not attack each other?
  - (Queens attack like towers + diagonally.)



Modeling by Propositional Variables

- Model NxN chessboard by NxN propositional variables $X_{i,j}$
- Semantics: $X_{i,j}$ is *true* iff there is a figure at row i, column j

- Example: 4x4 chessboard

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$
$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$
$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$
$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$

- Example solution:
  - $X_{1,2} = X_{2,4} = X_{3,1} = X_{4,3} =$ *true*
  - $X_{i,j} =$ *false* for all other $X_{i,j}$

Reducing the Problem to SAT

- Encode the properties of N-Towers to propositional formulas
- Example: 2-Towers

$X_{1,1} \rightarrow \neg X_{1,2}$	"Tower at (1,1) attacks to the right"
$X_{1,1} \rightarrow \neg X_{2,1}$	"Tower at (1,1) attacks downwards"
$X_{1,2} \rightarrow \neg X_{1,1}$	"Tower at (1,2) attacks to the left"
$X_{1,2} \rightarrow \neg X_{2,2}$	"Tower at (1,2) attacks downwards"
$X_{2,1} \rightarrow \neg X_{2,2}$	"Tower at (2,1) attacks to the right"
$X_{2,1} \rightarrow \neg X_{1,1}$	"Tower at (2,1) attacks upwards"
$X_{2,2} \rightarrow \neg X_{1,2}$	"Tower at (2,2) attacks to the left"
$X_{2,2} \rightarrow \neg X_{2,1}$	"Tower at (2,2) attacks upwards"
$X_{1,1} \vee X_{1,2}$	"Tower in first row"
$X_{2,1} \vee X_{2,2}$	"Tower in second row"

$X_{1,1}$	$X_{1,2}$
$X_{2,1}$	$X_{2,2}$

- Form a conjunction of all encoded properties:

$$(X_{1,1} \rightarrow \neg X_{1,2}) \wedge (X_{1,1} \rightarrow \neg X_{2,1}) \wedge (X_{1,2} \rightarrow \neg X_{1,1}) \wedge (X_{1,2} \rightarrow \neg X_{2,2}) \wedge (X_{2,1} \rightarrow \neg X_{1,1}) \wedge (X_{2,1} \rightarrow \neg X_{2,2}) \wedge (X_{2,2} \rightarrow \neg X_{1,2}) \wedge (X_{2,2} \rightarrow \neg X_{2,1}) \wedge (X_{1,1} \vee X_{1,2}) \wedge (X_{2,1} \vee X_{2,2})$$

Solving the Problem

- Determine satisfiability of

$(X_{1,1} \rightarrow \neg X_{1,2}) \wedge (X_{1,1} \rightarrow \neg X_{2,1}) \wedge (X_{1,2} \rightarrow \neg X_{1,1}) \wedge (X_{1,2} \rightarrow \neg X_{2,2}) \wedge (X_{2,1} \rightarrow \neg X_{1,1}) \wedge$
$(X_{2,1} \rightarrow \neg X_{2,2}) \wedge (X_{2,2} \rightarrow \neg X_{1,2}) \wedge (X_{2,2} \rightarrow \neg X_{2,1}) \wedge (X_{1,1} \vee X_{1,2}) \wedge (X_{2,1} \vee X_{2,2})$

- Satisfying variable assignment (others are possible):
  - $X_{1,1} = X_{2,2} = \mathit{true}$
  - $X_{1,2} = X_{2,1} = \mathit{false}$

$(\mathit{true} \rightarrow \neg \mathit{false}) \wedge (\mathit{true} \rightarrow \neg \mathit{false}) \wedge (\mathit{false} \rightarrow \neg \mathit{true}) \wedge (\mathit{false} \rightarrow \neg \mathit{true}) \wedge (\mathit{false} \rightarrow \neg \mathit{true}) \wedge$
$(\mathit{false} \rightarrow \neg \mathit{true}) \wedge (\mathit{true} \rightarrow \neg \mathit{false}) \wedge (\mathit{true} \rightarrow \neg \mathit{false}) \wedge (\mathit{true} \vee \mathit{false}) \wedge (\mathit{false} \vee \mathit{true})$

$(\mathit{true} \rightarrow \mathit{true}) \wedge (\mathit{true} \rightarrow \mathit{true}) \wedge (\mathit{false} \rightarrow \neg \mathit{true}) \wedge (\mathit{false} \rightarrow \neg \mathit{true}) \wedge (\mathit{false} \rightarrow \neg \mathit{true}) \wedge (\mathit{false}$
$\rightarrow \neg \mathit{true}) \wedge (\mathit{true} \rightarrow \mathit{true}) \wedge (\mathit{true} \rightarrow \mathit{true}) \wedge (\mathit{true} \vee \mathit{false}) \wedge (\mathit{false} \vee \mathit{true})$

$\mathit{true} \wedge \mathit{true} \wedge \mathit{true} \wedge \mathit{true} \wedge \mathit{true} \wedge \mathit{true} \wedge \mathit{true} \wedge \mathit{true} \wedge \mathit{true}$

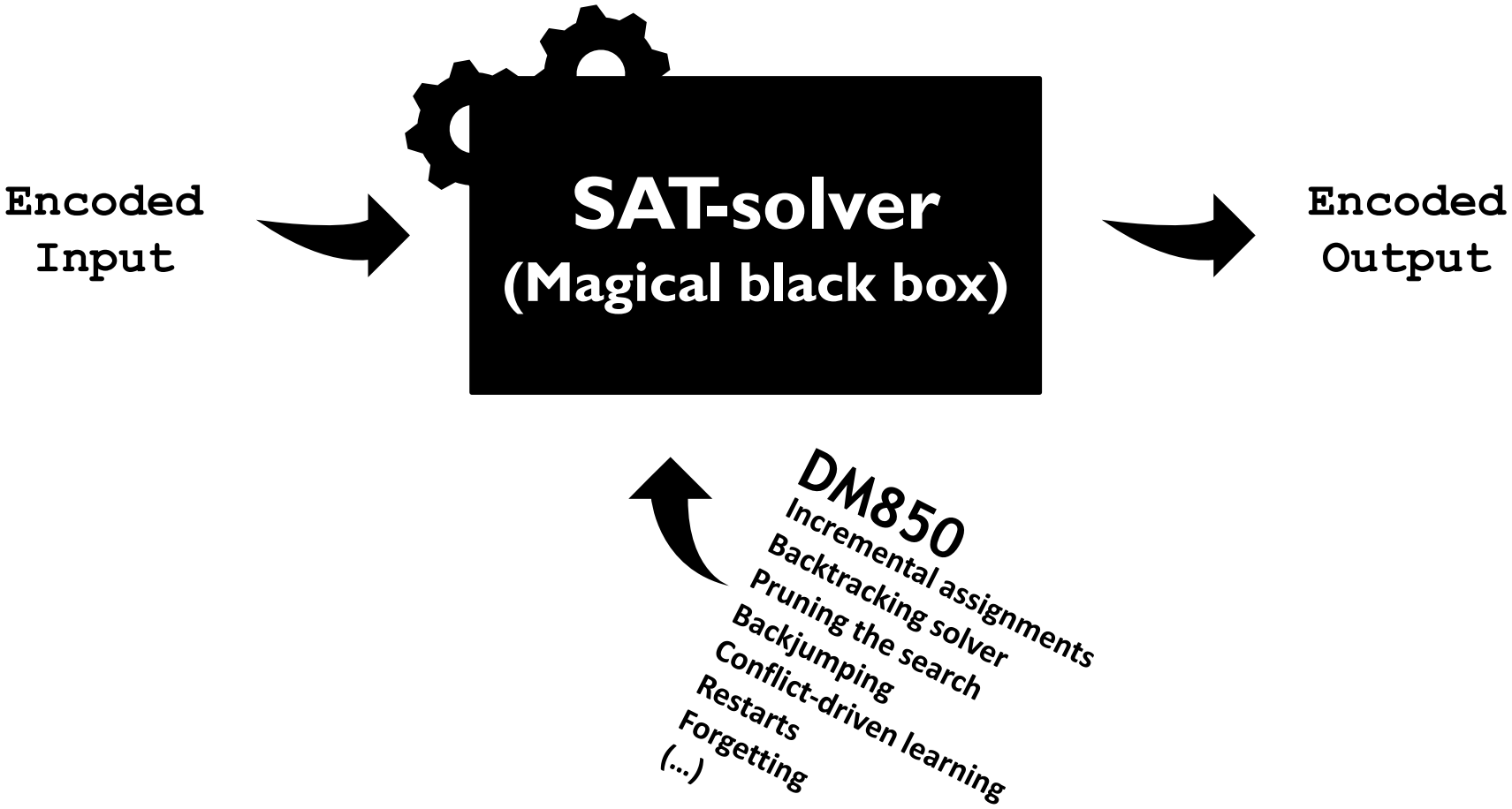
true

SAT Solving is Hard

- Given an assignment, it is easy to test whether it satisfies our formula
- BUT: there are many possible assignments!
- for m variables, there are 2^m possible assignments ☹
- SAT problem is a prototypical hard problem (NP-complete)

USING A SAT SOLVER

What is a SAT-solver?



SAT Solvers

- Plethora of SAT solvers available
  - For the best, visit http://www.satcompetition.org/
  - Different SAT solvers optimized for different problems

- One reasonable choice is the SAT solver **`lingeling`**
  - Very good overall performance at SAT Competition 2016
  - Parallelized versions available: `plingeling`, `treengeling`
  - Available from: http://fmv.jku.at/lingeling/

Conjunctive Normal Form (CNF)

- Nearly all SAT solvers require formulas in CNF
- CNF = **conjunction** of **disjunctions** of **literals**

- CNF transformations generally grow the number of clauses but translates a problem to a common language.

Conversion to CNF

1. Implications can be replaced by disjunction:
   - $F \rightarrow G$ converted to $\neg F \vee G$
2. DeMorgan's rules specify how to move negation "inwards":
   - $\neg(F \wedge G) = \neg F \vee \neg G$
   - $\neg(F \vee G) = \neg F \wedge \neg G$
3. Double negations can be eliminated:
   - $\neg(\neg F) = F$
4. Conjunction can be distributed over disjunction:
   - $F \vee (G \wedge H) = (F \vee G) \wedge (F \vee H)$

Example

- Example: 2-Towers

  $(X_{1,1} \rightarrow \neg X_{1,2}) \wedge (X_{1,1} \rightarrow \neg X_{2,1}) \wedge (X_{1,2} \rightarrow \neg X_{1,1}) \wedge (X_{1,2} \rightarrow \neg X_{2,2}) \wedge (X_{2,1} \rightarrow \neg X_{1,1}) \wedge$
  $(X_{2,1} \rightarrow \neg X_{2,2}) \wedge (X_{2,2} \rightarrow \neg X_{1,2}) \wedge (X_{2,2} \rightarrow \neg X_{2,1}) \wedge (X_{1,1} \vee X_{1,2}) \wedge (X_{2,1} \vee X_{2,2})$

- Conversion easy: $A \rightarrow B$ converted to $\neg A \vee B$

  $(\neg X_{1,1} \vee \neg X_{1,2}) \wedge (\neg X_{1,1} \vee \neg X_{2,1}) \wedge (\neg X_{1,2} \vee \neg X_{1,1}) \wedge (\neg X_{1,2} \vee \neg X_{2,2}) \wedge (\neg X_{2,1} \vee \neg X_{1,1}) \wedge$
  $(\neg X_{2,1} \vee \neg X_{2,2}) \wedge (\neg X_{2,2} \vee \neg X_{1,2}) \wedge (\neg X_{2,2} \vee \neg X_{2,1}) \wedge (X_{1,1} \vee X_{1,2}) \wedge (X_{2,1} \vee X_{2,2})$

- Write formulas in CNF as a list of clauses (= lists of literals)
- Example:

  $[[\neg X_{1,1}, \neg X_{1,2}],[\neg X_{1,1}, \neg X_{2,1}],[\neg X_{1,2}, \neg X_{1,1}],[\neg X_{1,2}, \neg X_{2,2}],[\neg X_{2,1}, \neg X_{1,1}],[\neg X_{2,1}, \neg X_{2,2}],$
  $[\neg X_{2,2}, \neg X_{1,2}],[\neg X_{2,2}, \neg X_{2,1}],[X_{1,1},X_{1,2}],[X_{2,1},X_{2,2}]]$

Write in conjunctive normal form

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

Go to www.menti.com and use the code 8502 6148

Option a

$A \wedge (\neg B \wedge \neg C)$

Option b

$A \wedge (\neg B \vee \neg C)$

Option c

$\neg A \vee (B \wedge C)$

Option d

$(A \vee B) \wedge (A \vee \neg C)$

Variable Enumeration

- SAT solvers expect variables to be identified with integers
- Starting from 1 and up to the number of variables used
- Necessary to map modeling variables to integer!
- Example: 4x4 chessboard
  - $X_{i,j}$ becomes 4*(i-1)+j

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$
$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$
$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$
$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(Simplified) DIMACS Format

- Description of DIMACS format for CNF (BB: dimacs.pdf)
- Simplified format (subset) implemented by most SAT solvers:
  - http://www.satcompetition.org/2016/format-benchmarks2016.html

- 2 types of lines for input
  - Starting with "`c `": **comment**
  - Starting with "`p `": **problem**
- 3 types of lines for output
  - Starting with "`c `": **comment**
  - Starting with "`s `": **solution**
  - Starting with "`v `": **variable assignment**

Input Format 1/2

- **Comments**
  - Anything in a line starting with "`c` " is ignored
  - Example:

```
c This file contains a SAT encoding of the 4-queens problem!
c The board is represented by 4x4 variables:
c          1  2  3  4
c          5  6  7  8
c          9 10 11 12
c         13 14 15 16
c
```

Input Format 2/2

- **Problem**
  - Starts with "`p cnf #variables #clauses`"
  - Then one clause per line where
    - Variables are numbered from 1 to #variables
    - Clauses/lines are terminated by 0
    - Positive literals are just numbers
    - Negative literals are negated numbers
  - Example:

```
p cnf 16 80
 -1  -2 0
...
-15 -16 0
 1  2  3  4 0
...
13 14 15 16 0
```

Output Format 1/2

- **Comments**
  - just like for the input format
  - Example:

```
c reading input file examples/4-queens.cnf
```

- **Solution**
  - Starts with "`s `"
  - Then either "`SATISFIABLE`" or "`UNSATISFIABLE`"
  - Example:

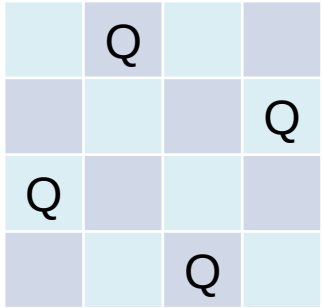
```
s SATISFIABLE
```

Output Format 2/2

- **Variable assignment**
  - Starts with "`v `"
  - Then list of literals that are assigned to true
    - "`1`" means variable 1 is assigned to true
    - "`-2`" means variable 2 is assigned to false
  - Terminated by "`0`"
  - Example:

```
v -1 2 -3 -4 -5 -6 -7 8 9 -10 -11 -12 -13 -14 15 -16 0
```

1 **2** 3 4	*false **true** false false*
5 6 7 **8**	*false false false **true***
9 10 11 12	***true** false false false*
13 14 **15** 16	*false false **true** false*



Running the SAT Solver

1. Save the comment and problem lines into .cnf file.
2. Invoke the SAT solver on this file.
3. Parse the standard output for the solution line.
4. If the solution is “s SATISFIABLE”, find variable assignment.

Practical Part

WRITING A SAT SOLVER

Brute-Force Solver

- iterate through all possible variable assignments
- for each assignment
  - if the assignment satisfies the formula
    - output SAT and the assignment
- if no assignment is found, output UNSAT

Empirical Evaluation

- For n variables, there are 2^n possible variable assignments

- **Example:**
- 2^{16} = 65,536 assignments for 4-queens (1 second)
- 2^{25} = 33,554,432 assignments for 5-queens (7 minutes)
- 2^{36} = 68,719,476,736 assignments for 6-queens (2 weeks)
- 2^{49} = 562949953421312 assignments for 7-queens (400 years)
- 2^{64} assignments for 8-queens (age of the universe)
- 2^{81} assignments for 9-queens (ahem … no!)

Fast Forwarding 60+ Years

- Incremental assignments
- Backtracking solver
- Pruning the search

Empirical Evaluation

- For n variables, there are 2^n possible variable assignments

- Example:
  - 2^{100} assignments for 10-queens (1.77 seconds)
  - 2^{121} assignments for 11-queens (1.29 seconds)
  - 2^{144} assignments for 12-queens (9.15 seconds)
  - 2^{169} assignments for 13-queens (5.21 seconds)
  - 2^{196} assignments for 14-queens (136.91 seconds)
  - ...

Fast Forwarding 60+ Years

- Incremental assignments
- Backtracking solver
- Pruning the search
- Backjumping
- Conflict-driven learning
- Restarts
- Forgetting

Empirical Evaluation

- For n variables, there are 2^n possible variable assignments

- Example:
  - 2^{256} assignments for 16-queens (0.02 seconds)
  - 2^{1024} assignments for 32-queens (0.10 seconds)
  - 2^{4096} assignments for 64-queens (1.08 seconds)
  - 2^{16384} assignments for 128-queens (17.92 seconds)
  - 2^{65536} assignments for 256-queens (366.05 seconds)
  - ...

Efficient SAT Solving

- in many cases, SAT problems can be solved efficiently
- state-of-the-art SAT solvers can be used as black boxes
- success of SAT solvers based on
  - relatively simple but highly-optimized algorithms
  - innovative and very pragmatic data structures
- used extensively for scheduling, hardware and software verification, mathematical proofs, ...

Take Home Slide

- SAT Problem = satisfiability of propositional logic formulas
- SAT used to successfully model hard (combinatorial) problems
- solving the SAT problem is hard in the general case
- advanced SAT solvers work fine (most of the time)

QUESTIONS?

Logic puzzles