DM573
Introduction to Computer Science
Exercises on Satisfiability

Peter Schneider-Kamp

petersk@imada.sdu.dk

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

PART I

Exercise I-1: Check Satisfiability

Which of the following formulas are satisfiable (give a satisfying assignment)? Which are not (give reasons)?

Note: in all exercises, logical negation (NOT) is denoted by "-".

a) $A \wedge B$
b) $A \vee B$
c) $A \rightarrow B$
d) $A \wedge -A$
e) $A \vee -A$

Exercise 1-2: Equivalent Formulas

Two formulas are equivalent, if the same assignments satisfy both of them.

Which of the following formulas are equivalent?

a) $\neg A \wedge B$

b) $\neg A \vee B$

c) $A \rightarrow B$

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

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

Exercise 1-3: Convert to CNF

Convert the following formulas into CNF:

a) $-A \wedge B$

b) $-A \vee B$

c) $A \rightarrow B$

d) $(A \rightarrow B) \wedge (-B \rightarrow A)$

Exercise 1-4: Breaking Symmetry

Solutions to N-Towers and N-Queens are symmetric:

	Q		
			Q
Q			
		Q	

and

		Q	
Q			
			Q
	Q		

Write two clauses that forbid solutions where there is a queen in the right half of the first row.

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

Exercise 1-5: Preparation

- Install **`lingeling`** or another compatible SAT solver
- Alternatively, use a Javascript SAT solver, e.g.:
  - https://www.msoos.org/2013/09/minisat-in-your-browser/
- Test it using the following input (also available on the course web page as the file `towers2x2.cnf`)

```
p cnf 4 6
-1 -2 0
-1 -3 0
-2 -4 0
-3 -4 0
1 2 0
3 4 0
```

Exercise 1-6: Removing Redundancies

The formula from page 11 on Peter's slides contains redundant information. For example, $X_{1,1} \rightarrow -X_{1,2}$ and $X_{1,2} \rightarrow -X_{1,1}$ are equivalent. Understand and remove these redundancies:

a) Why do these redundancies occur?
b) Identify all such redundancies!
c) Write down a simplified formula without redundancies!
d) Convert the simplified formula into CNF!
e) Write the formula in DIMACS format!
f) Run the `lingeling` or another SAT solver on it and interpret the result!

PART II

Exercise II-1: Check Satisfiability

Which of the following formulas are satisfiable (give a satisfying assignment)? Which are not (give reasons)?

a) $(A \rightarrow B) \land (B \rightarrow A)$

b) $(A \rightarrow B) \land (B \rightarrow A) \land A$

c) $(A \rightarrow B) \land (B \rightarrow A) \land -A$

d) $(A \rightarrow B) \land (B \rightarrow -A) \land (-A \rightarrow -B) \land (-B \rightarrow A)$

Exercise II-2: Equivalent Formulas

Two formulas are equivalent, if the same assignments satisfy both of them.

Which of the following formulas are equivalent?

a) $(A \rightarrow B) \land (-B \rightarrow A)$

b) $(A \rightarrow -B) \land (B \rightarrow A)$

c) $(-A \lor -B) \land (A \lor -B)$

d) $(B \lor A) \land (-A \lor B)$

Exercise II-3: Convert to CNF

Convert the following formulas into CNF:

a) (–A ➔ B) ∧ (–B ➔ –A)

b) A ➔ (– (B ∧ D))

c) A ➔ (– (B ∨ D))

d) A ➔ (– (B ➔ (C ∧ D)))

Exercise II-4: 3-Towers

Write a Python program that generates the input for a SAT solver to solve the 3-Towers problem:

a) Write a function `pair2int(r,c)` which maps (1,1), (1,2), ..., (3,3) to 1 to 9 using the formula 3*(r-1)+c.
b) Write nested for-loops that go through all positions on the board from (1,1) to (3,3) and produces clauses that represent attacks.
c) Write a for-loop that produces clauses that specify that all 3 rows contain a tower.
d) Using (a)–(c), write a DIMACS file and test it using lingeling or another SAT solver.

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

1	2	3
4	5	6
7	8	9

Exercise II-5: N-Towers

Generalize your Python program from Exercise II-4 to generate the input for a SAT solver to solve the N-Towers problem:

a) Write a function `pair2int(n,r,c)` which maps pairs (r,c) to the integers 1 to n^2 using the formula n*(r-1)+c.

b) Write nested for-loops that go through all positions on the board from (1,1) to (n,n) and produces clauses that represent attacks.

c) Write a for-loop that produces clauses that specify that all rows contain a tower.

d) Using (a)–(c), write a DIMACS file and test it using lingeling or another SAT solver.

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

1	2	3
4	5	6
7	8	9

Exercise II-6: N-Queens

Extend your Python program from Exercise II-5 to generate the input for a SAT solver to solve the N-Queens problem:

a) Reuse your function `pair2int(n,r,c)` from Exercise II-5.
b) Adapt your for-loops from Exercise II-5 to produce also clauses for the diagonals.
c) Reuse the for-loop from Exercise II-5 that produces clauses that specify that all rows contain a tower.
d) Using (a)–(c), write a DIMACS file and test it using lingeling or another SAT solver.

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

1	2	3
4	5	6
7	8	9