DM534
Introduction to Computer Science
Study Groups on Satisfiability

Peter Schneider-Kamp

petersk@imada.sdu.dk

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

Task 1: 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(n,p)` that maps pairs (1,1), (1,2), ..., (3,3) to the integers 1 to 9 using the formula 3*(row-1)+column.

b) Write nested for-loops that go through 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.

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

1	2	3
4	5	6
7	8	9

Task 2: N-Towers

Generalize your Python program from Task 1 to generate the input for a SAT solver to solve the N-Towers problem:

a) Write a function `pair2int(n,p)` that takes a pair p = (i,j) of two integers and maps it to the integers 1 to n^2 using n*(i-1)+j.

b) Write nested for-loops that go through 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.

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

1	2	3
4	5	6
7	8	9

Task 3: N-Queens

Extend your Python program from Task 2 to generate the input for a SAT solver to solve the N-Queens problem:

a) Reuse your function `pair2int(n,p)` from Task 2.

b) Adapt your for-loops from Task 2 to produce also clauses for the diagonals.

c) Reuse the for-loop from Task 2 that produces clauses that specify that all rows contain a tower.

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

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

1	2	3
4	5	6
7	8	9