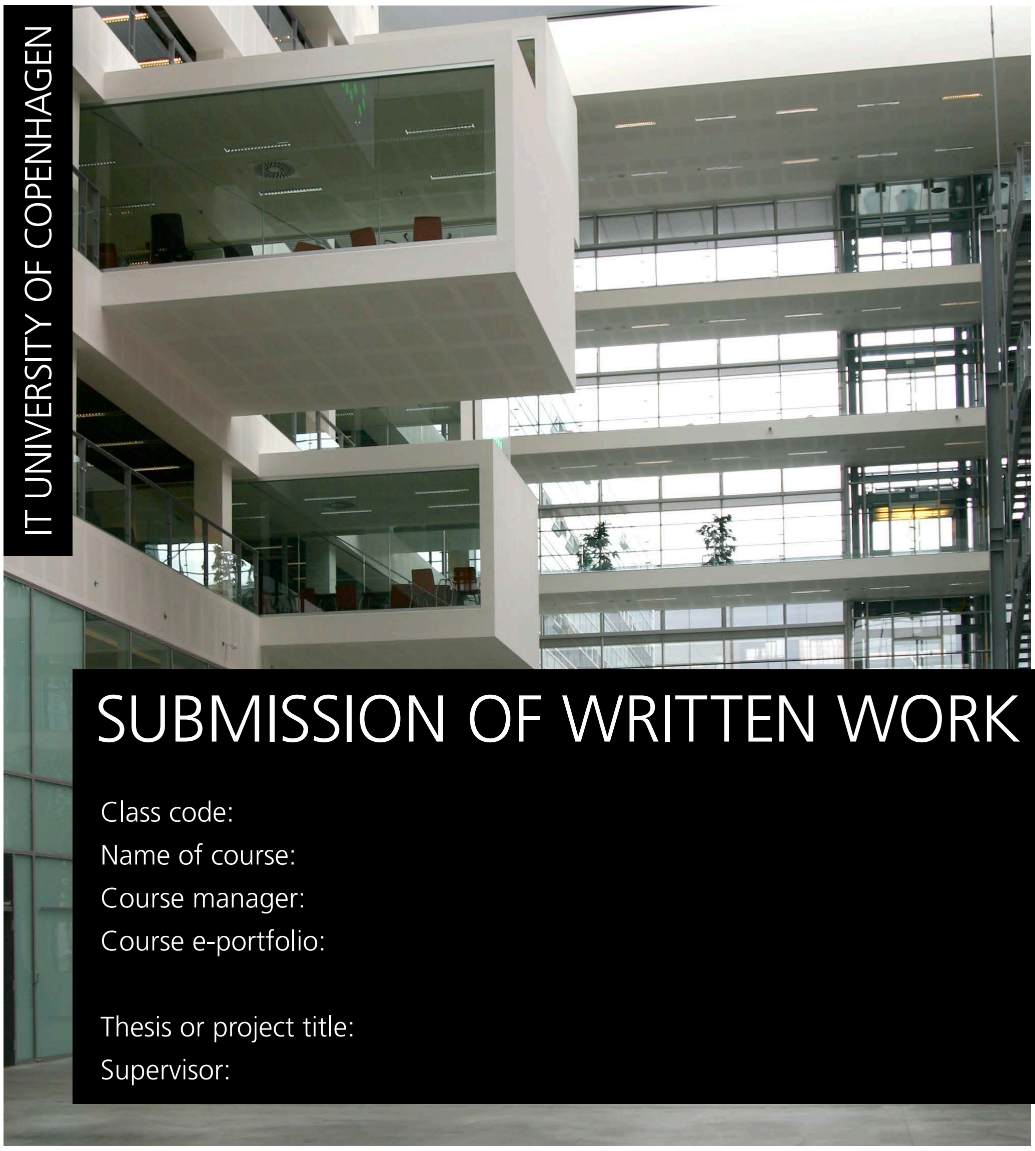

SUBMISSION OF WRITTEN WORK

Class code:
Name of course:
Course manager:
Course e-portfolio:

Thesis or project title:
Supervisor:

Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.		@itu.dk
2.		@itu.dk
3.		@itu.dk
4.		@itu.dk
5.		@itu.dk
6.		@itu.dk
7.		@itu.dk

Report: Sudoku

by Karina Abramova,
Mikkel Bernt Buchvardt
and
Theodor Lars Nyholm Ommen

1 Introduction

[4, 0, 0, 2, 0, 8, 0, 0, 3, 0, 0, 1, 0, 0, 0, 9, 0, 0, 0, 7, 0, 0, 9, 0, 0, 2, 0, 6, 0, 0, 3, 0, 0, 0, 0, 8, 0, 0, 5, 0, 0, 0, 1, 0, 0, 8, 0, 0, 0, 0, 9, 0, 0, 5, 0, 5, 0, 0, 3, 0, 0, 9, 0, 0, 0, 7, 0, 0, 0, 3, 0, 0, 2, 0, 0, 9, 0, 4, 0, 0, 6]

2 Rules

As described in the assignment there are two rules to be observed:

- On an N by N size board N queens has to be placed
- All queens are to be considered able to of different color

We had to find a way to translate these two rules into boolean functions.

There has to be exactly one queen in each row and each column In each diagonal there can also be only one queen.

Another way of describing the exactly one queen in each row is by saying that if there is a queen in the first cell, there cannot be one in the second and there cannot be one in the third and so on. These rules have to be entered in the BDD and only then can you start placing queens on the board (and in the BDD).

3 Code

For setting expression on where a queen can be placed we set up the BDD with nand expressions for all fields depending on each other (if the first then not the second and not the third and so on).

For instance the code for the expression that only one cell in a column can contain a queen is done as follows(from line 121 in QueensLogic.java):

```
/* No one in the same column */
for (l = 0; l < N; l++) {
if (l != j) { // if it is not the same variable
BDD u = X[i][l].apply(X[i][j], BDDFactory.nand);
// creating tmp BDD by applying the binary operator opr NOT AND to the
   two BDDs.
// ie. (0 and not 5 and not 10 and not 15 and not 20 ) for the first
   column
//    (1 and not 6 and not 11 and not 16 and not 21) for the seconnd
   row ... (for N=5)
a.andWith(u);
}
```

The other rules are entered into the BDD in the same way and they are finally and'ed to each other in one BDD called queen.

'queen' is now a BDD representing all the rules an is ready for restrictions (the placing of queens in different cells).

In order to ensure that no queen placed on the board will violate any of the rules we looked at BDD.pathCount() method for the BDD 'queen' restricted to the placement of a queen (what actually happens to the BDD when a queen is placed at a specific field on the board?).

In details it is done by the insertQueen() method that inserts a queen by restricting the variable that corresponds to the cell to the BDD queen and then iterate through every cell on the board. By temporary restricting every cell to the resulting BDD of the placement and checking if there exists a path with this restriction it can be determined if the cell should be crossed or stay open.

4 User Interface

We used a print of the board to the console for debugging and used the GUI for the final version. We kept the console printouts to make it easier to understand what is actually going on.

We decided not to build in means of termination in to the program as the user will want to see the result of the configuration.