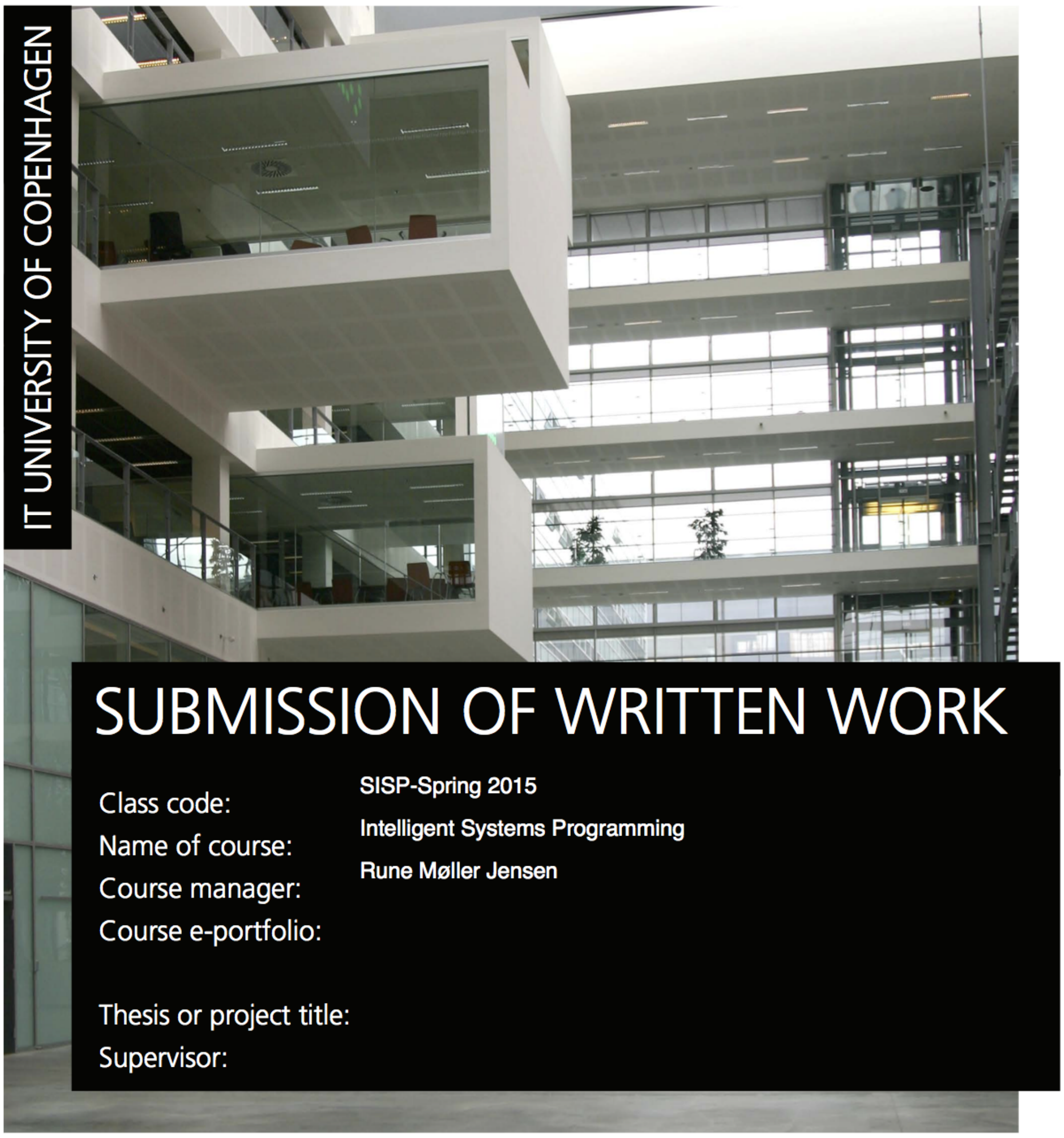

SUBMISSION OF WRITTEN WORK

Class code:	SISP-Spring 2015
Name of course:	Intelligent Systems Programming
Course manager:	Rune Møller Jensen
Course e-portfolio:	
Thesis or project title:	
Supervisor:	

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Theodor Lars Nyholm Ommen	27/05-1960	thlo@itu.dk
2.	Karina Abramova	14/09-1988	kaab@itu.dk
3.	Mikkel Bernt Buchvardt	27/11-1982	mbbu@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

The most important aspects of our implementation are:

When the solve() method is called (by user pressing the 'Solve' button in the GUI) before doing anything else all the domains are updated.

After this the provided INITIAL_FC() is run. If there is not consistency the program terminates with the message "Sudoku cannot be solved".

If there is consistency FC() method is called, this returns the first complete assignment that it finds.

2 Rules

The rules for assignments are simple - the numbers form 1 - 9 has to be somewhere in each row and each column and also in each defined square of 3 by 3 cells.

Provided methods called GetColumn(), GetRow() and GetVariable() are used to traverse for consistency.

3 Code

```
/* 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.