# DM22 Programming Languages

## Spring 2006

## Project 2

Department of Mathematics and Computer Science
University of Southern Denmark

May 3, 2006

The purpose of this project is to implement in Prolog a program finding solutions to the puzzle known by the name *Professorspillet*. The project is to be done in groups of two persons.

## Professorspillet

*Professorspillet*[1] is a 16-piece puzzle. Each piece is quadratic and has half a professor (torso or legs) at each of its four sides, and each professor has clothing in one of four colors. The objective of the game is to lay out the pieces in a 4x4 formation such that all professors formed along meeting edges have clothing of the same color in the torso and the legs parts. The pieces of the puzzle are depicted in Appendix A.

## Task

The task is to implement a Prolog program which can generate all solutions to *Professorspillet* one by one. More specifically, implement in Prolog predicates

[1] Produced by *Danspil*. See `http://www.danspil.dk/produkt.asp?ProductID=8483` for more info (note in particular the suggested player age of 11+ years and expected playing time of 30 minutes(!)).

```
profSolution(S)
noOfProfSolutions(N)
```

such that the first is true iff `S` is a solution to the game, and the second is true iff `N` is the total number of solutions (including symmetric ones) to the game.

Here, a solution should be a list of oriented pieces, where an orientation is between zero and three 90 degree clockwise turns of the piece compared to the position it has in Appendix A. The actual representation of an oriented piece is left to you.

The implementation must be based on the following recursive solution method: The pieces are at all times divided into a list constituting a partial solution and a list with the remaining pieces. Consider the slots of the 4x4 layout numbered as shown below.

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

Then a partial solution of length $k$ fills positions 0 to $k-1$ (with professors at edges matching according to the rules). At each step, the algorithm tries to extend the partial solution by trying to place a remaining piece in the next free position, $k$, and for each succesful attempt recurses with the new value of partial solution and list of remaining pieces. In pseudo-code, a step can be stated as

```
For each piece in remaining list
  For each orientation of the piece
    If fits in position k then
      recurse with updated solution and remaining list
```

A solution is found if the length of the partial solution reaches 16. For extracting each element of a list, the built-in predicate `select` may come in handy.

## Formalities

A printed report of two to five pages should be handed in. Prolog code and any test data should be given as appendices. The main aim of the report should be to describe the modeling and program design choices made during development, the reasoning behind these choices, and the structure of the final solution. Give two solutions found by your program, and state the total number of solutions according to your program.

A copy of the Prolog code should be handed in using the `aflever` command on the Imada system: Move to the directory containing your code and issue the command `aflever DM22`. This will copy the contents of the directory to a place accessible by the lecturer. Repeated use of the command is possible (later uses overwrites the contents from earlier uses).

In the directory, you must for identification purposes have an ASCII file named `names.txt` containing the names of the group members, with one name per line.

You must hand in the report and the code by

*Tuesday, May 23, 2006*

## A Pieces