Department of Mathematics and Computer Science
University of Southern Denmark, Odense

November 13, 2008
KSL

DM509 Exam

Obligatory Assignment, part 1: PROLOG

Kim Skak Larsen
Fall 2008

Introduction

In this note, we describe one part of the exam project which must be solved in connection with the course DM509, Programming Languages, Fall 2008. It is important to read through the entire project description before starting the work on the project; also the sections on requirements and how to turn in your solution.

Deadline

Thursday, December 4, 2008 at noon.

Mazes and Minefields

You must solve two different, though very related, problems. First, you must design a maze solver, and then you must write a script to find the best possibly path, in a sense to be described.

A Maze Solver

You are given a description of a maze in a PROLOG format. Each unit of the maze is referred to using coordinates, and you are told where to start and where your goal is. In Fig. 1, you can see an example of this format. After the start and goal predicated, the file simply lists all *free* units. An illustration of this maze can be seen in Fig. 2.

You must define a predicate `shortest/1` such that the goal `shortest(L).` produces all shortest paths from the start to the goal. In the given example, there is only one path from the start to the goal which is therefore the only answer. The effect we want to see is listed in Fig. 3.

This example also defines the output format we want. The functor symbol `c` stands for "coordinates". Notice how the output specifies the path from the start to the goal in the maze. Also notice how you can start PROLOG up on your script file and then afterwards load a file defining the maze.

If we add `cell(5,5)` to the file, we get the file illustrated in Fig. 4. Here, the solution is still unique, because even though there are two different paths from start to goal, one is shorter than the other. We would expect the output given in Fig. 5.

However, if *instead* of adding `cell(5,5)`, we add `cell(1,1)`, and obtain the maze illustrated in Fig. 6, then there are suddenly two shortest paths, and we should see the result given in Fig. 7.

A Mine Solver

You are given a description of an area in a PROLOG format. Each unit of the area is referred to using coordinates, and a third value (the unit's *weight*) specifies how unpleasant that unit is (this is a generalized

minefield). You are also told where to start and where your goal is. In Fig. 8, you can see an example of this format. An illustration of this minefield can be seen in Fig. 9. For clarity, the indication of start and goal have been omitted from the illustration.

You must define a predicate `lightest/1` that finds all lightest paths from the start to the goal, where the weight of a path is the sum of all weights on it.

In this little example, the predicate must find the unique lightest path which goes through the unit of weight four and otherwise only ones. The output format should be the same as for the maze problem.

Execution Requirements

The programs must be written in GNU PROLOG. They must of course be able to run on the department's computers, i.e., the ones in the terminal room. Example computers are `logon`⟨digit⟩`.imada.sdu.dk`, where ⟨digit⟩ can be any of the digits $1, \ldots, 9$. These are the computers you can connect to using `ssh` from outside IMADA.

You are very welcome to develop your programs at home, but it is your responsibility. This includes technical problems at home, lack of access to relevant software, moving data to IMADA via modem, e-mail, ftp, floppy disks, USB keys, etc. and converting to the correct format, e.g., between Windows and Linux.

You must hand in two separate PROLOG scripts that must be called `maze.pl` and `mine.pl` (all lower case). The first should contain the definition of the predicate `shortest` and the second should contain the definition of the predicate `lightest`.

Turning In

You should produce a small report describing what you have done. A sufficient collection of tests must be carried, and you must verify and document in the report that the correct results are produced. The report must also contain a complete listing of programs.

Possible omissions, known errors, etc. should be described in the report. It is often a good idea to do this in a separate section instead of mixing it in with the rest of the report.

All the material described above must be turned in on paper at the secretaries' office and also electronically via Blackboard.

The paper version and the electronic version must of course be identical, except that you must date and sign the paper version. Also, for the purpose of the exam protocol, please clearly indicate your full name and birthday to make sure that we correctly identify you. It is also a good idea to write your IMADA login as well as your preferred e-mail address.

The secretaries' office may be closed for very short periods of time. If, for some unexpected reason, the office must be closed for longer periods of time close to the deadline, an announcement will be made outside the office, giving instructions as to where you turn in your report.

Exam Rules

An obligatory assignment is an exam project. Cooperation beyond what is explicitly permitted will be considered cheating and will be treated as such. You have a duty to keep your notes private and protect your files against reading and copying by others. Both parties involved in a possible plagiarism can be held responsible.

There will be given what we judge to be more than sufficient time for each assignment and you are strongly encouraged to plan your work such that you will finish some days before the deadline.

Assignments which are turned in after the deadline will not be accepted. Downtime on the system or the printers will not automatically result in an extension; not even if it is the last hours before the deadline. Neither will own or children's illness without a statement from your physician, etc.

Figures

```
start(0,0).
goal(6,6).
cell(0,0).
cell(0,1).
cell(0,2).
cell(0,3).
cell(0,4).
cell(0,5).
cell(0,6).
cell(1,4).
cell(1,6).
cell(2,0).
cell(2,1).
cell(2,2).
cell(2,3).
cell(2,4).
cell(2,6).
cell(3,4).
cell(4,0).
cell(4,2).
cell(4,3).
cell(4,4).
cell(4,5).
cell(4,6).
cell(5,0).
cell(5,2).
cell(6,0).
cell(6,1).
cell(6,2).
cell(6,3).
cell(6,4).
cell(6,5).
cell(6,6).
```

Figure 1: Example file `Maze7Unique.pl`.

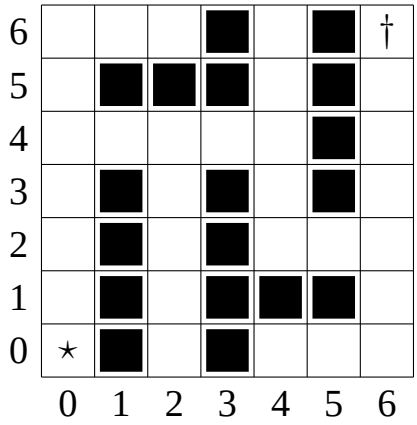

Figure 2: Illustration of example file `Maze7Unique.pl`.

```
(13:42) ~> gp maze.pl
GNU Prolog 1.3.0
By Daniel Diaz
Copyright (C) 1999-2007 Daniel Diaz
| ?- ['maze.pl'].
compiling /home/kslarsen/maze.pl for byte code...
/home/kslarsen/maze.pl compiled, 24 lines read - 6747 bytes written,
11 ms

yes
| ?- ['Maze7Unique.pl'].
compiling /home/kslarsen/Maze7Unique.pl for byte code...
/home/kslarsen/Maze7Unique.pl compiled, 33 lines read - 2951 bytes
written, 10 ms

yes
| ?- shortest(P).

P = [c(0,0),c(0,1),c(0,2),c(0,3),c(0,4),c(1,4),c(2,4),c(3,4),c(4,4),
c(4,3),c(4,2),c(5,2),c(6,2),c(6,3),c(6,4),c(6,5),c(6,6)] ? ;

(580 ms) no
| ?-
```

Figure 3: Running shortest on `Maze7Unique.pl`.

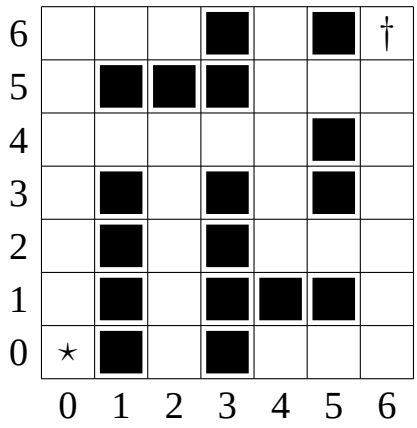

Figure 4: Illustration of example file `Maze7UniqueSolution.pl`.

```
| ?- ['Maze7UniqueSolution.pl'].
compiling /home/kslarsen/Maze7UniqueSolution.pl for byte code...
/home/kslarsen/Maze7UniqueSolution.pl compiled, 34 lines read - 3047
bytes written, 9 ms

yes
| ?- shortest(P).

P = [c(0,0),c(0,1),c(0,2),c(0,3),c(0,4),c(1,4),c(2,4),c(3,4),c(4,4),
c(4,5),c(5,5),c(6,5),c(6,6)] ? ;

(920 ms) no
| ?-
```

Figure 5: Running shortest on `Maze7UniqueSolution.pl`.

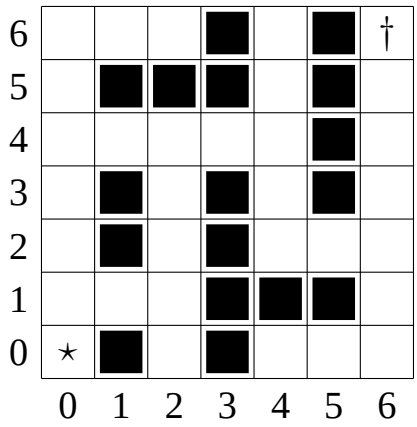

Figure 6: Illustration of example file `Maze7TwoSolutions.pl`.

```
| ?- ['Maze7TwoSolutions.pl'].
compiling /home/kslarsen/Maze7TwoSolutions.pl for byte code...
/home/kslarsen/Maze7TwoSolutions.pl compiled, 34 lines read - 3044
bytes written, 9 ms

yes
| ?- shortest(P).

P = [c(0,0),c(0,1),c(0,2),c(0,3),c(0,4),c(1,4),c(2,4),c(3,4),c(4,4),
c(4,3),c(4,2),c(5,2),c(6,2),c(6,3),c(6,4),c(6,5),c(6,6)] ? ;

P = [c(0,0),c(0,1),c(1,1),c(2,1),c(2,2),c(2,3),c(2,4),c(3,4),c(4,4),
c(4,3),c(4,2),c(5,2),c(6,2),c(6,3),c(6,4),c(6,5),c(6,6)] ? ;

(1816 ms) no
| ?-
```

Figure 7: Running shortest on `Maze7TwoSolution.pl`.

```
start(0,0).
goal(3,3).
cell(0,0,1).
cell(0,1,3).
cell(0,2,3).
cell(0,3,3).
cell(1,0,1).
cell(1,1,3).
cell(1,2,3).
cell(1,3,3).
cell(2,0,1).
cell(2,1,4).
cell(2,2,1).
cell(2,3,1).
cell(3,0,1).
cell(3,1,3).
cell(3,2,3).
cell(3,3,1).
```

Figure 8: Example file `Mine4.pl`.

3	3	3	1	1
2	3	3	1	3
1	3	3	4	3
0	1	1	1	1
	0	1	2	3

Figure 9: Illustration of example file `Mine4.pl`.