

CS 380: Assignment 3

Fall 2006

Written Due before class on 2006-11-13

From R&N Chapter 7: Exercise 7.8

From R&N Chapter 8: Exercises 8.6 and 8.10

Please show all work.

Programming Part I Due before class on 2006-11-15

The programming portion of this assignment is broken into two distinct parts, each of equal value. The second part does not build upon the first; the first part is simply intended to introduce you to the Prolog programming language.

SWI-Prolog (a popular implementation of Prolog) is available on `tux.cs.drexel.edu` by running the command `pl`. To quit the interpreter at any time, press `CTRL-d`. To load a Prolog file (*i.e.*, your source code), run `consult('yourfilename')`. Note that the trailing period is required.

For this part of the programming portion, you are to solve Exercise 7.9 from the textbook in Prolog. This means you must encode all of the following as a Prolog knowledge base:

- If the unicorn is mythical then it is immortal;
- If the unicorn is not mythical then it is a mortal mammal;
- If the unicorn is either immortal or a mammal then it is horned;
- The unicorn is magical if it is horned.

It should then be possible to query your knowledge base (using Prolog) to determine whether or not the unicorn is mythical, magical, and/or horned. For this part of the assignment, you should submit (through E-mail) a file titled `YourfirstnameYourlastname.pl` and instructions on how to solve the problem using your code.

Programming Part II Due before class on 2006-11-20

You are to create a Prolog solver for a simplified (*i.e.*, 4×4) version of Sudoku.

A 4×4 matrix is said to have the “Sudoku Property” iff neither the rows, columns, nor the 2×2 sub-matrices have any duplicate elements. For example, the following matrix satisfies the Sudoku Property:

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

You are given a 4×4 matrix that is partially-complete; for example:

1		3	
			2
4			
	3		1

The object is to fill in the empty elements such that the resulting matrix satisfies the Sudoku Property. For this portion of the assignment, you need to write a Prolog solver for this simplified version of Sudoku. You may implement this within Prolog however you like. Here is an example of what your implementation should be able to do:

```
Welcome to SWI-Prolog (Multi-threaded, Version 5.6.17)
Copyright (c) 1990-2006 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
```

```
For help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- consult('EvanSultanikSudokuSolver.pl').
% EvanSultanikSudokuSolver.pl compiled 0.00 sec, 7,836 bytes
```

```
Yes
?- sudoku([[1,_,3,_],
           [_,_,_,2],
           [4,_,_,_],
           [_,3,_,1]]).
```

```
[1, 2, 3, 4]
[3, 4, 1, 2]
[4, 1, 2, 3]
[2, 3, 4, 1]
```

```
Yes
?-
```

Once again, you are to submit your Prolog code through E-mail. Please include any instructions necessary for running/testing your code.

EXTRA CREDIT: Extend your Sudoku solver to work with “regular” 9×9 Sudoku grids. Depending on your implementation and the nature of the given input, your 9×9 solver may take an inordinate amount of time to solve the problem; this is okay.