University of Arizona, Department of Computer Science

CSc 372 — Assignment 4 — Due noon, Tue Oct 18 — 5%

Christian Collberg
October 5, 2011

1 Introduction

The purpose of this assignment is to familiarize yourself with Prolog.

1. Every predicate should be commented. At the very least, the comments should state what the predicate does, which arguments it takes, and what result it produces.
2. You should make your predicates as simple and elegant as possible. In particular, stay away from abominations such as:
   (a) `foo(A,B) :- A = B.`
   Instead use:
   ```
   foo(A,A).
   ```
   (b) `foo(A) :- bar(A); zap(A).`
   Instead use:
   ```
   foo(A) :- bar(A).
   foo(A) :- zap(A).
   ```
   In fact, I do not want you to use *any* semi-colons in this assignment!
3. You will be graded primarily on **correctness**, **elegance**, **clarity**, and **documentation**.
4. This assignment is graded out of 100. It is worth 5% of your final grade. This is an individual assignment.

2 Magic Squares

Write a predicate `magic(B)` which takes a *board* (a 3×3 grid) and succeeds if all rows, all columns, and the diagonals sum to 15: [10 points]

```
| ?- magic(board(2,7,6,9,5,1,4,3,8)).
yes
| ?- magic(board(2,7,6,9,5,1,4,3,7)).
no
```

A board is represented by a structure like this:

```
board(A,B,C,      % First row
      D,E,F,      % Second row
      G,H,I)      % Third row
```

Your predicate should start like this:

```
magic(board(A,B,C,D,E,F,G,H,I)) :- ...
```

You don't need to check that all the entries on a board are unique.

3 Prolog Arithmetic

1. Write a predicate `abs(X,Y)` which takes a number `X` as input and computes the absolute value $|X|$ as output: [10 points]

   ```
   | ?- abs(0,0).
   yes
   | ?- abs(-1,1).
   yes
   | ?- abs(-1,Y).
   Y = 1
   | ?- abs(5,Y).
   Y = 5
   ```

2. Write a Prolog predicate `seq(First, Last, N)` that instantiates `N` to each integer value from `First` through `Last`, inclusive. `First` and/or `Last` may be negative. Assume that `First` and `Last` are integers. Examples: [10 points]

   ```
   | ?- seq(1, 3, X).
   X = 1 ? ;
   X = 2 ? ;
   X = 3 ? ;
   no

   | ?- seq(-3, 2, X).
   X = -3 ? ;
   X = -2 ? ;
   X = -1 ? ;
   X = 0 ? ;
   X = 1 ? ;
   X = 2 ? ;
   no

   | ?- seq(1,1,X).
   X = 1 ? ;
   no
   ```

```
| ?- seq(2,-2,X).
no
```

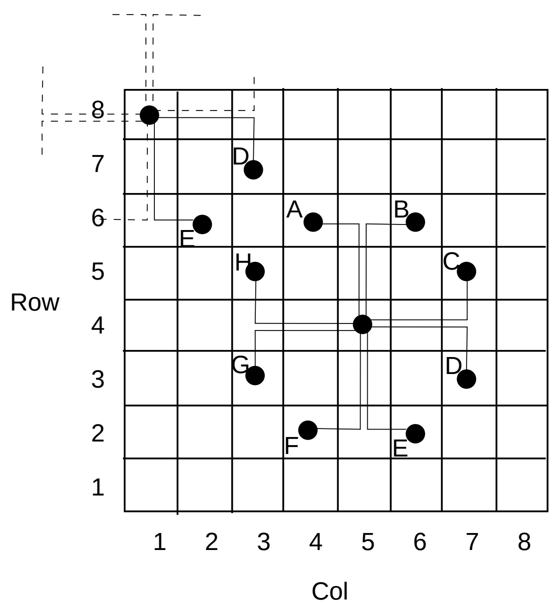
4 Chess

Write a predicate `knight(Row,Col,Row1,Col2)` that, given a position `(Row,Col)` on a chess board, will produce all the legal positions where a knight might move from that position. [15 points]

From Wikipedia (`http://en.wikipedia.org/wiki/Knight_(chess)`):

> The knight move is unusual among chess pieces. When it moves, it can move two squares horizontally and one square vertically, or two squares vertically and one square horizontally. The complete move therefore looks like the letter *L*. Unlike all other standard chess pieces, the knight can "jump over" all other pieces (of either color) to its destination square. It captures an enemy piece by moving into its square. The knight's ability to "jump over" other pieces means it is at its most powerful in closed positions, in contrast to that of a bishop. The move is one of the longest-surviving moves in chess, having remained unchanged since before the seventh century. Because of this it also appears in most chess-related national games. The knight moves alternately to white and black squares.

Consider this illustration:



From position (4,5) (we give position as (row,column) pairs) the knight can move to the positions A-H. From position (8,1), the knight can only move to D and E — it has to stay within the board!

Here are some examples:

```
| ?- knight(4,5,R,C).
C = 4
R = 6 ? ;
C = 6
R = 6 ? ;
C = 7
R = 5 ? ;
C = 7
R = 3 ? ;
C = 6
R = 2 ? ;
C = 4
R = 2 ? ;
C = 3
R = 3 ? ;
C = 3
R = 5
yes

| ?- knight(4,5,5,7).
true ? ;
no

| ?- knight(8,1,R,C).
C = 3
R = 7 ? ;
C = 2
R = 6 ? ;
no
```

NOTE: Generate the position exactly as in the above example, i.e. starting with position A and then going clockwise around.

NOTE: Your predicate *must* use backtracking!

.

HINT: You may want to start with a predicate `withinBoard(Row,Col)` that succeeds for positions within the board and fails otherwise. Then, write a predicate `move(R,C)` which gives the positions where the knight can move, relative to the current position.

5 Peano Axioms (Rerun!)

Let's once again represent integers using the Peano axioms: `0=zero, 1=succ(zero), 2=succ(succ(zero)), 3=succ(succ(succ(zero)))`, etc. Write a predicate `sum(A,B,C)` that adds `A` and `B` making `C`. Below are some examples. Note how the predicate can be invoked with different arguments being instantiated or variables, and how that affects its behavior. [10 points]

```
| ?- sum(zero,zero,R).
R = zero
yes
| ?- sum(zero,zero,zero).
yes
| ?- sum(zero,succ(zero),R).
R = succ(zero)
yes
| ?- sum(succ(zero),succ(zero),R).
R = succ(succ(zero))
yes
| ?- sum(succ(zero),succ(succ(zero)),R).
R = succ(succ(succ(zero)))
yes
| ?- sum(succ(zero),succ(succ(zero)),succ(succ(succ(zero)))).
yes
| ?- sum(A,B,succ(succ(succ(zero)))).
A = zero
B = succ(succ(succ(zero))) ? ;

A = succ(zero)
B = succ(succ(zero)) ? ;

A = succ(succ(zero))
B = succ(zero) ? ;

A = succ(succ(succ(zero)))
B = zero ? ;
no
| ?- sum(succ(zero),B,succ(succ(succ(zero)))).
B = succ(succ(zero))
yes
```

6 Parts List

Consider this database of facts, describing what something is made up of: [15 points]

```
has(bicycle,wheel,2).
has(bicycle,handlebar,1).
has(bicycle,break,2).
has(wheel,hub,1).
has(wheel,spoke,32).
has(bicycle,frame,1).

has(car,steering_wheel,1).
has(car,stereo,1).
has(car,tires,4).
```

Write a predicate `partof(X,Y)` that succeeds if `Y` is a part of `X`:

```
| ?- partof(wheel,spoke).
true
| ?- partof(bicycle,spoke).
true
| ?- partof(car,spoke).
no
```

`partof(X,Y)` can also be used to enumerate the parts that make up an object, or of which an object is a part:

```
| ?- partof(bicycle,X).
X = wheel ? ;
X = handlebar ? ;
X = break ? ;
X = frame ? ;
X = hub ? ;
X = spoke ? ;
no
| ?- partof(X,spoke).
X = wheel ? ;
X = bicycle ? ;
no
```

7 Dating Database

You are running a computerized dating service and maintain a database consisting of the following facts:

```
% name, gender, height(cm), age, education (hs,college,masters,phd))
person(lisa, female, 180, 30, phd).
person(jenny, female, 167, 25, hs).

person(bob, male, 180, 40, phd).
person(charles, male, 190, 30, masters).
person(arnold, male, 177, 29, hs).
```

You know from experience that a woman will only date a man if

1. he is at least as tall as she is,
2. his educational level is at least as high as hers,
3. he is not younger, and no more than 10 years[1] older than her.

For the purposes of this assignment we assume that women will date only men and men will date only women.

1. Add these facts

   ```
   edu_less(hs,college).
   edu_less(college,masters).
   edu_less(masters,phd).
   ```

   to your program and write a recursive rule `edu_le(A,B)` that succeeds if the educational level `A` is less than or equal to `B`: [15 points]

   ```
   | ?- edu_le(hs,hs).
   yes
   | ?- edu_le(hs,college).
   yes
   | ?- edu_le(hs,phd).
   yes
   | ?- edu_le(phd,masters).
   no
   ```

[1] Ha!

2. Write a Prolog predicate `dateable(Female,Male)` which encodes the dating rules above[2]: [15 points]

```
| ?- dateable(lisa,charles).
no
| ?- dateable(lisa, bob).
yes
| ?- dateable(jenny,arnold).
yes
```

8 Submission and Assessment

The deadline for this assignment is noon, Tue Oct 18. It is worth 5% of your final grade.

You should submit the assignment electronically using `d2l.arizona.edu`.

Name the file `ass4.pl`.

Don't show your code to anyone, don't read anyone else's code, don't discuss the details of your code with anyone. If you need help with the assignment see the instructor or the TA.

[2] Every assignment really should contain a Seinfeld quote, shouldn't it? So, here goes, from *The Wink*:

Jerry:	Elaine, what percentage of people would you say are good looking?
Elaine:	Twenty-five percent.
Jerry:	Twenty-five percent, you say? No way! It's like 4 to 6 percent. It's a twenty to one shot.
Elaine:	You're way off.
Jerry:	Way off? Have you been to the motor vehicle bureau? It's like a leper colony down there.
Elaine:	So what you are saying is that 90 to 95 percent of the population is undateable?
Jerry:	UNDATEABLE!
Elaine:	Then how are all these people getting together?
Jerry:	Alcohol.