CSCI 235, Programming Languages, Prolog, Exercise 1

Deadline: Sunday 17.11.2019, 21.00

We will be using SWI-prolog, which is installed in lab 7.422. Type `prolog` to start the interpreter. It will show a command prompt `?-` The following commands may be useful:

- `consult('file.pl').` or `['file.pl'].` reads a file. In order to reload a file, use the same command. The interpreter will automatically forget the previous versions of the predicates in the file. So, SWI-Prolog is much nicer than Python. In case, the file name is unambiguous, you can omit the extension `pl`.
- If you want to leave the interpreter, type `halt.`
- If you want to list your program, you can type `listing( )` (lists everything), or `listing( pred )` (lists the given predicate).
- It is important to know that Prolog interpreters do not accept whitespace between a function (or predicate) name and the following '('. So, typing `fact ( 4, X )` will result in an unpleasant error message, while `fact( 4, X )` be accepted.

  It is also useful to know that every command always ends with a dot (`.`)
- If you use a variable only once in a predicate, the interpreter will produce a warning 'Singleton variables: ....'. It is intended as protection against typing errors. It can be safely ignored, if you are sure that there are no errors.
- Don't use the cut predicate `!`. It is sometimes useful, but not in this exercise.
- Try to avoid disjunction `;`. It is nearly always better to split up into different Horn clauses.

In the following task, we are going to solve the 8-queens problem. This is the problem of putting 8 queens on a chess board in such a way that no two queens are on the same place, or can strike each other.

Your solution has to include all of the predicates below. The predicates are graded separately.

1. Write a predicate `dist(X,Y,Z)` that succeeds if `Z` is the distance between `X` and `Y`. `Z` is never less than zero. You will need $\leq$. The syntax in Prolog is `=<`. Examples:

   ```
   ?- dist(1,4,Z).
      Z = 3 ;
   ?- dist(4,2,Z).
      Z = 2 ;
   ?- dist(1,1,Z).
      Z = 0 ;
   ```

2. Write a predicate `canstrike( q(X1,Y1), q(X2,Y2) )` that succeeds if a queen on position `(X1,Y1)` can strike a queen on position `(X2,Y2)`. This is the case if the queens are in the same row, the same column, or the distance between `X1` and `X2` equals the distance between `Y1` and `Y2`.

3. Write a predicate `betw( L, U, X )` that succeeds if `X` is between `L` and `U` (borders included). This predicate must be able to enumerate all `X` between `L` and `U`. This predicate must be called `betw` because the name `between` is already taken by a built-in predicate. Examples:

   ```
   ?- betw(2,1,X).
      false.
   ?- betw(2,5,X).
     X = 2 ;
     X = 3 ;
     X = 4 ;
     X = 5 ; (answer with ; to enumerate all solutions up to 5.)
   ```

   Make sure that the predicate always terminates. Hint for implementation: Use the fact that betw(L, U, X) means that either $L \leq U$ and $L = X$, or $L < U$ and betw$(L + 1, U, X)$.

4. Write a predicate `cannotstrike( L, q(X,Y) )` that succeeds if a queen on `(X,Y)` cannot strike any of the queens in `L`. Examples are:

   ```
   ?- cannotstrike( [], q(1,2) ).
   true.

   ?- cannotstrike( [ q(2,3) ], q(1,2)  ).
   false.

   ?- cannotstrike( [ q(4,1), q(5,3) ], q(1,2)  ).
   true.
   ```

5. Finally write a predicate `queens( I, N, P1, P2 )` that succeeds if `P1` and `P2` are lists of queen placements on an `N` times `N` board, and `P2` is an extension of `P1` containing `I` more queens.

The predicate must be able enumerate possible `P2` when `I,N,P1` are given.

Alternatively, one can say that `queens(I,N,P1,P2)` tries to put `I` additional queens into the placement `P1`.

The 8-queens problem can be solved by calling `queens( 8,8,[], Solution )`.

```
?- queens( 2, 8,[], S ).
S = [ q(2, 3), q(1, 1) ]
```

(There are many more solutions.)

```
?- queens( 2, 3,[], S ).
   S = [ q(2, 3), q(1, 1) ]
```

(Has 16 solutions.)

Hint: There are two cases: When $I = 0$, nothing needs needs to be done. When $I > 0$, first place $I - 1$ queens by using the predicate recursively. After that, use `betw` twice to guess an X- and a Y-coordinate for the I-th queen. Check if she cannot strike any of the other queens, and if she cannot, put her in front of the list of existing queens, using [|].