AIT Semantic and Declarative Technologies course, Fall 2015

Assignment 2: Solving a Skyscraper Sudoku puzzle

Deadline: 11.59pm CET, 20th November, 2015

Description

This task is related to the well-known Sudoku puzzle.

A **Sudoku** grid is a square matrix consisting of m rows and m columns where m itself is a square number, $m = k^2$. We refer to the elements of the matrix as the "fields" of the grid. The Sudoku grid subdivides to square sub-grids of k rows and k columns.

For the mathematically minded, the top left fields of sub-grids are in (row $i \cdot k+1$, column $j \cdot k+1$), for $i = 0, \ldots, k-1$ and $j = 0, \ldots, k-1$; assuming that the top left field of the whole grid is in row 1, column 1.

A Sudoku puzzle is given by providing a partially filled in Sudoku grid, where each field is either empty or contains a number between 1 and m. An example Sudoku puzzle for $k = 2$ is shown below; sub-grid borders are indicated by thick lines.

		4	1
4		2	
	2	1	4
	4		2

When solving a Sudoku puzzle, the empty fields have to be filled with numbers between 1 and m in such a way that no number appears twice in a row, column, or sub-grid in the filled-in Sudoku grid.

In the **Skyscraper Sudoku** puzzle variant, additional clues may be provided around the grid. Here, a number in a Sudoku field is viewed as the height (in storeys) of a building – skyscraper – placed in the given field of the grid. The additional clues provide information on how many buildings are visible in a row or a column, when viewed from a given direction. A building is visible if and only if only lower buildings stand in front of it.

A skyscraper clue can be placed to the left (or to the right) of a row, specifying the number of buildings visible from the left (right). Similarly, a skyscraper clue can be supplied above (or below) a column, describing the number of buildings visible from above (below). Not all such clues are necessarily supplied. An example Skyscraper Sudoku puzzle for $k = 2$ is shown below:

			3	
3				
			2	2
2				
		1		

The task

Using `library(clpfd)`, write a Prolog predicate `skysudoku/2` which takes a Skyscraper Sudoku puzzle `SP` as its first, input argument, and returns a solution `SS` of `SP` in the second argument. The predicate should enumerate all solutions of `SP` on backtracking. The order in which the solutions are enumerated does not matter. If `SP` has no solutions, then `skysudoku/2` should fail. Make your program efficient using the learned methods.

A Skyscraper Sudoku puzzle is specified as a Prolog compound term `ss(K,Clues)`, where `K` is the sub-grid size (referred to above as k), and `Clues` is a list of clues. There are two kinds of clues:

- The term `g(N,R,C)` represents a "given number" clue: number `N` is known to occupy the field in row `R` column `C`.
- The term `v(V,Dir,RC)` represents a "visible count" clue. Here, `Dir` specifies the direction from which the visible buildings are counted. `Dir` is one of the atoms `n`, `e`, `s`, `w` (standing for north, east, south, west). `RC` is a row or column number to which the clue applies. `V` is the count of the buildings visible from the direction `Dir` in row or column `RC`.

Rows and columns are counted from 1, i.e., the top left field is row 1 column 1. Accordingly, `R`, `C` and `RC` can take values `1..m`, where `m = K*K`. Numbers `N` (present in given number clues) and visible counts `V` also take values `1..m`. You don't have to check these preconditions, your program will be run only on test cases that satisfy them.

The above Skyscraper Sudoku puzzle is represented by the following Prolog term:

```
ss(2,[g(2,2,3),v(3,n,4),v(1,s,2),v(2,e,2),v(3,w,1),v(2,w,3)])
```

The solution of a Skyscraper Sudoku puzzle should be returned as a full Sudoku grid, represented in Prolog as a list of rows, each row being a list of numbers `1..m`. The only solution of the above sample puzzle should be returned as:

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

The data types used by the predicate `skysudoku/2` are specified below using Mercury-style type declarations:

```
% :- type spuzzle ---> ss(size,list(clue))
% :- type size == int.
% :- type clue ---> g(num,row,col) ; v(vcount,dir,rowcol).
% :- type row == int.
% :- type col == int.
% :- type rowcol == int.
% :- type num == int.
% :- type vcount == int.
% :- type dir ---> n ; e ; s ; w.

% :- type ssol == list(list(num)).

% :- pred skysudoku(spuzzle::in, ssol::out).
% skysudoku(SP, SS): SS is a solution of the Skyscraper Sudoku puzzle SP.
```

Sample runs

```
| ?- skysudoku(ss(1,[]), SS).
SS = [[1]] ? ;
no
| ?- skysudoku(ss(2,[v(4,n,1),v(4,s,4)]), SS).
SS = [[1,3,2,4],[2,4,1,3],[3,1,4,2],[4,2,3,1]] ? ;
no
| ?- skysudoku(ss(2,[v(4,n,1),g(4,1,4)]), SS).
SS = [[1,3,2,4],[2,4,1,3],[3,1,4,2],[4,2,3,1]] ? ;
SS = [[1,3,2,4],[2,4,1,3],[3,2,4,1],[4,1,3,2]] ? ;
SS = [[1,3,2,4],[2,4,3,1],[3,1,4,2],[4,2,1,3]] ? ;
no
| ?- skysudoku(ss(2,[g(4,1,1),v(2,n,1)]), SS).
no
| ?- skysudoku(ss(3,[v(2,n,1),v(6,n,2),v(3,n,3),v(3,n,4),v(1,n,5),
                    v(3,n,6),v(4,n,7),v(3,n,8),v(2,n,9),v(2,w,1),
                    v(1,w,2),v(3,w,3),v(3,w,4),v(3,w,5),v(2,w,6),
                    v(4,w,7),v(3,w,8),v(2,w,9),v(2,s,1),v(2,s,2),
                    v(1,s,3),v(4,s,4),v(3,s,5),v(3,s,6),v(2,s,7),
                    v(3,s,8),v(3,s,9),v(2,e,1),v(5,e,2),v(1,e,3),
                    v(4,e,4),v(2,e,5),v(4,e,6),v(2,e,7),v(3,e,8),
                    v(4,e,9),g(6,1,4),g(8,2,6),g(2,3,8),g(6,4,3),
                    g(1,5,5),g(3,6,7),g(5,7,2),g(5,8,4),g(4,9,6)]), SS),
     (   member(Row, SS), write(Row), nl, fail
     ;   nl, fail
     ).
[7,2,5,6,9,1,4,3,8]
[9,3,1,4,2,8,7,6,5]
[6,4,8,7,5,3,1,2,9]
[3,7,6,9,4,2,8,5,1]
[5,8,4,3,1,7,2,9,6]
[1,9,2,8,6,5,3,7,4]
[2,5,3,1,8,9,6,4,7]
[4,1,7,5,3,6,9,8,2]
[8,6,9,2,7,4,5,1,3]

no
```

The last, 9x9 puzzle is taken from here: `http://www.garethmoore.co.uk/2009/05/skyscraper-sudoku/`. You may want to print this puzzle and try to solve it also manually.