Lucia Winter School 2014

B. Andres / T. Schaub

Assignment 1

Problem Description. The task of this project is to solve a Sudoku[1] puzzle using ASP. The goal of the game is to fill a 9x9 grid with digits so that each column, each row and each of the nine 3x3 sub-grids that compose the grid contains all numbers from 1 to 9. In other words, the grid has to be filled with numbers from 1 to 9 so that the same number does not appear twice in the same row, column or in any of the nine 3x3 subregions of the 9x9 playing board. Initially the grid is partially filled.

One example is shown in Figure 1. The left figure shows the initial configuration, and the right figure shows the same puzzle with solution numbers marked in red.

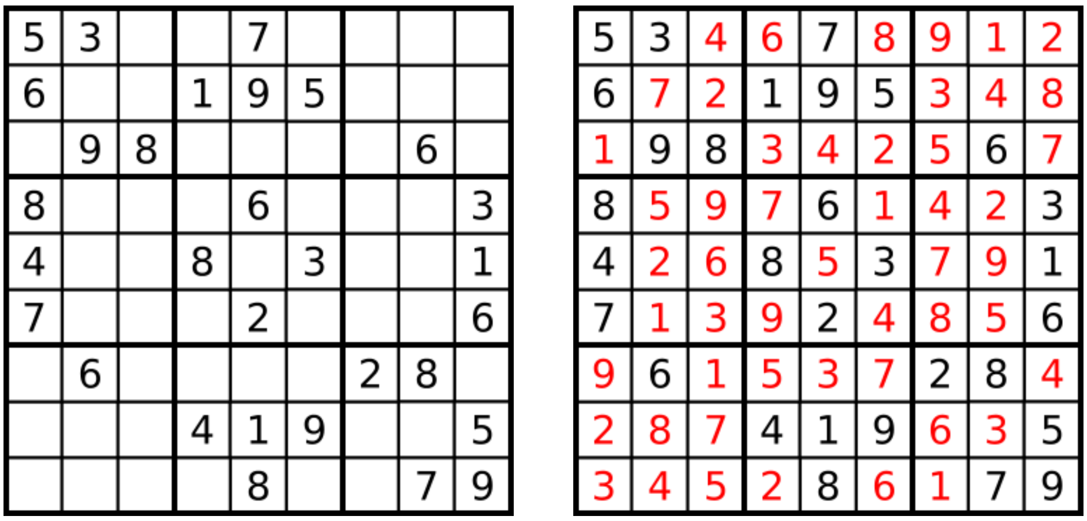

Figure 1: A sudoku and its solution.

Representation in ASP. The initial state of the grid is represented by facts of predicate `initial/3`:

```
initial(X,Y,N).   % initially cell [X,Y] contains number N
```

For instance, the example shown in Figure 1 is represented by the following facts:

```
initial(1,1,5). initial(1,2,3). initial(1,5,7).
initial(2,1,6). initial(2,4,1). initial(2,5,9). initial(2,6,5).
initial(3,2,9). initial(3,3,8). initial(3,8,6).
initial(4,1,8). initial(4,5,6). initial(4,9,3).
initial(5,1,4). initial(5,3,8). initial(5,6,3). initial(5,9,1).
initial(6,1,7). initial(6,5,2). initial(6,9,6).
initial(7,2,6). initial(7,7,2). initial(7,8,8).
initial(8,4,4). initial(8,5,1). initial(8,6,9). initial(8,9,5).
initial(9,5,8). initial(9,8,7). initial(9,9,9).
```

The solution is represented by facts of predicate `sudoku/3`:

```
sudoku(X,Y,N).   % the sudoku in cell [X,Y] contains number N
```

[1] `http://en.wikipedia.org/wiki/Sudoku`

For instance, the solution of Figure 1 consists of squarethe following atoms:

```
sudoku(1,1,5) sudoku(1,2,3) ... sudoku(1,8,1) sudoku(1,9,2)
...
sudoku(9,1,3) sudoku(9,2,4) ... sudoku(9,8,7) sudoku(9,9,9)
```

In order to be able to provide an encoding for solving sudoku puzzles of varying sizes the instance file provides the the size of the sub-grids through the fact `sub_size(N)`. You may use this fact to get the number of cells as well as the number of different values for a specific puzzle (both N^2). In addition, you are able to identify all sub-grids of the puzzle with the following rule:

```
subgrid(X,Y,G) :- cell(X), cell(Y), sub_size(S), G = (X-1)/S*S+(Y-1)/S+1.
```

Representing that the cell [X,Y] belongs to the sub-grid G.

Frame1work. You are given 5 different sudoku instance files (ex1.lp to ex5.lp) for the sizes 4x4 and 9x9 each. Write an ASP encoding `sudoku.lp` that will solve each one of the instance files. You can test your encoding by calling the ASP grounder+solver clingo together with both files, e.g.

```
clingo-4.4.0 sudoku.lp examples\4x4\ex1.lp
```

To call ex1.lp of the size 4x4 instances. Provided you are in the sudoku folder and have a link to your clingo-4.4.0 executable.

Tips:

- To begin with, it may be easier to work with the 4x4 sudoku and once this is done, try to handle the 9x9 cases.
- Try to generate first all possible assignments for all grid cells via **choice rules** and then reduce the possible assignments to valid one via **integrity constraints**.
- To reduce unnecessary output you may add the following line to your encoding:

  ```
  #show sudoku/3.
  ```

- Commands to find all stable models look as follows (there may be only one, though):

  ```
  $  clingo-4.4.0 sudoku.lp examples\4x4\ex1.lp 0
  ```