Declarative Problem Solving

Prof. Torsten Schaub, Javier Romero Davila, Arne König

Universität Potsdam — Wintersemester 2013/2014

Project 1 (Weaving)

Problem Description. The task of this project is to solve a logic puzzle using ASP. The goal of the game is to color a board one row or column at a time such that the final result matches the predetermined colors.

At each step you color the squares in exactly one column or row, and the game is finished after each column and row has been colored exactly once. Coloring a column or row replaces the current color of each square with a color chosen by the player. To win the game each final color of a square has to match its predetermined color, and squares without a predetermined color can be colored arbitrarily.

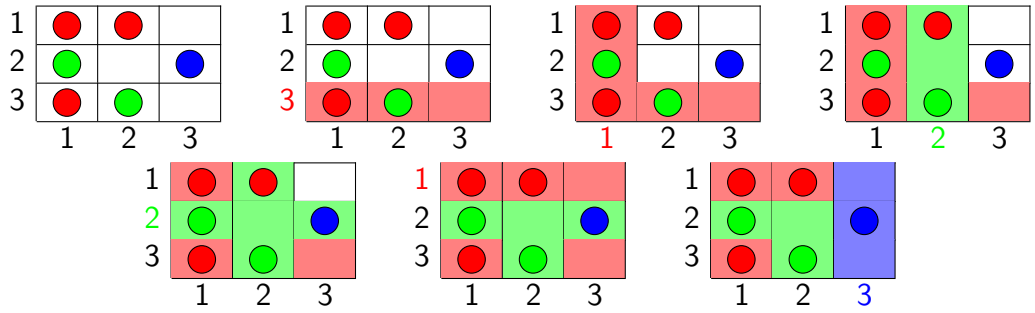

Figure 1: Example with a 3×3 board and step by step solution.

One example is shown in Figure 1. The top left table shows the initial configuration with six predetermined colors, and the following tables represent the steps in one possible solution. The painted column or row is marked with a colored number on the side or bottom of the table. The last table shows the final state with all square colors matching the predetermined ones.

Representation in ASP. The dimensions of the board, the number of steps, the colors used, and the predetermined colors are represented by the following facts:

```
dim(D).                 % Dimensions of the board
steps(S).               % Number of steps necessary
color(C).               % Available color
square_color(X,Y,C).    % Predetermined color
```

Additionally, the predicate `gets_painted/4` is used to predefine some of the painting steps. The first parameter defines whether a row (`y`) or column (`x`) is painted, the second defines the row or column number, the third its color and the last at which point in the game the action takes place (note that the first step has the number 1).

```
gets_painted(A,N,C,S). % Predetermined painting step
```

For instance, the example shown in Figure 1 (top left) is represented by the following facts:

```
dim(3). steps(6). color(red). color(green). color(blue).
square_color(1,1,red). square_color(2,1,red).
square_color(1,2,green). square_color(3,2,blue).
square_color(1,3,red). square_color(2,3,green).
gets_painted(y,3,red,1). gets_painted(x,1,red,2).
```

A solution – i.e. the ordered list of painted columns and rows – is represented as follows:

```
paint(A1,N1,C1,1)  paint(A2,N2,C2,2)  ...  paint(An,Nn,Cn,n)
```

The parameters of `paint/4` are identical to those of `gets_painted/4`, and the solution in Figure 1 is represented by the following atoms:

```
paint(y,3,red,1)    paint(x,1,red,2)  paint(x,2,green,3)
paint(y,2,green,4)  paint(y,1,red,5)  paint(x,3,blue,6)
```

Note that the first two steps are identical to the ones predefined by `gets_painted/4`.

Framework. In the `weaving.tar.gz` archive you will find the `weaving.lp` file and five example instances. The file `weaving.lp` has to be completed with your ASP encoding. When you submit your encoding, the following lines have to be included in the `weaving.lp` file:[1]

```
#hide.
#show paint/4.
```

The file containing your encoding has to be called `weaving.lp`. Apart from `paint/4` no additional predicates must be included in the output!

Formalities. You can work on the solution alone or in groups of two people. Different groups have to submit different solutions, in case of plagiarism all groups involved will fail the project. Please submit your encoding by **March 2, 2014** via YETI (All group members have to create a YETI account!). Be sure to submit your encoding in a file named `weaving.lp` containing only lowercase letters.

We will test your encoding with the five provided instances as well as additional instances. Your encoding has to find the correct solutions for all instances. This will be tested automatically by YETI after you uploaded the encoding (with a slight delay). If your solution is not correct then YETI will display an error message. Please correct any errors that occur on your own or contact us if you get stuck.

[1] These instructions hide all atoms that are not part of the solution representation.

Tips:

- Commands to find all stable models look as follows:

```
$  gringo weaving.lp levelX.lp | clasp 0
```

- If you are stuck you can contact us. We will do out best to answer all your questions. You can send us questions and remarks via Moodle (forum/wiki) or send them via mail to `asp@lists.cs.uni-potsdam.de`.
- Start as soon as possible to avoid running out of time. (However, if you still realize that you have problems making it before the deadline, please contact us instead of copying another solution).