Answer Set Solving in Practice

Prof. Torsten Schaub, Javier Romero

University of Potsdam — Winter Semester 2014/2015

Project 3 (Minotaur)

Problem Description. The task of this project is to solve the Minotaur puzzle using ASP. You are in a labyrinth and the goal of the game is to go from a start field to a goal field without being eaten by the Minotaur. The labyrinth has walls, and neither you nor the Minotaur can go through them. At every step you can move one field in any direction: up, down, left or right. The Minotaur is faster than you, but luckily he is a bit foolish. He can move two fields every step, but his movements are determined by his and our position: there is no choice for him. He will try to move right or left to become closer to you. If that is not possible, he will try to move up and down to become closer. If this is also not possible, the Minotaur will not move. In this way the Minotaur may move in a step first in one direction (e.g., up) and then in another direction (e.g., left). Typically, to solve a puzzle you have to trap the Minotaur between the walls of the labyrinth, so that you can run free to the goal. You can play the game at: `http://www.janko.at/Spiele/Minotaurus`. It is fun and it will help you to get used to the rules.

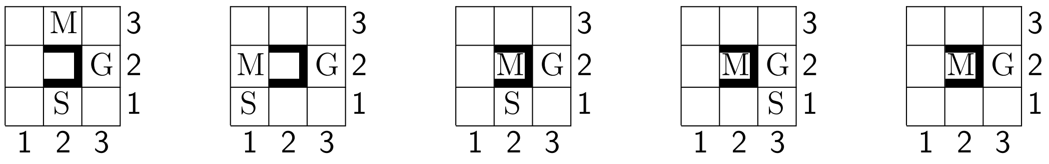

Figure 1: Minotaur Game.

Figure 1 represents a game with start field $(2,1)$, goal field $(3,2)$ and the Minotaur at $(2,3)$. You can scape from the Minotaur and reach the goal with these movements:

1. Go from the start field $(2,1)$ to $(1,1)$. The Minotaur follows you first to the left and then down to $(1,2)$.
2. Now move back to the field $(2,1)$, and the Minotaur is trapped! He moves to the right to become closer, going to the field $(2,2)$, and then he cannot move any more: to come closer he would have to move down, but there is a wall in between.
3. Now move right to $(3,1)$ and the Minotaur remains trapped: he could come closer going right or down, but there are walls there.
4. Finally you can move up safely to the goal field $(3,2)$, while the Minotaur stays at $(2,2)$.

Observe how the Minotaur first tries to shorten the horizontal distance, and only when that is not possible he tries to shorten the vertical gap. If none of these is possible, the Minotaur remains at the same position. The movements of the Minotaur are thus completely deterministic.

You win the game if you move from the initial to the goal field and the Minotaur never moves to the same field where you are.

Representation in ASP. The labyrinth, the initial situation and the maximal number of steps are represented by facts of the following predicates:

```
field(X,Y).          % the field [X,Y] belongs to the labyrinth
start(X,Y).          % the start field is [X,Y]
goal(X,Y).           % the goal  field is [X,Y]
mino(X,Y).           % the Minotaur starts at [X,Y]
wall(X,Y,XX,YY).     % there is a wall between [X,Y] and [XX,YY]
maxsteps(S).         % there are at most S steps to reach the goal
```

Facts of the predicate `wall/4` always have the form `wall(X,Y,X,Y+1)` or `wall(X,Y,X+1,Y)`, in this way the walls refer to adjacent horizontal or vertical fields, and this reference is unique for every possible wall. The game of figure 1 with a limit of 10 steps can be described with the following facts:

```
field(1..3,1..3).
start(2,1).
goal(3,2).
mino(2,3).
wall(2,1,2,2). wall(2,2,3,2). wall(2,2,2,3).
maxsteps(10).
```

If there is a limit of S steps, then a solution is a sequence of at most $S+1$ positions of a winning game. It is represented by facts of the predicate `at/3`:

```
at(X,Y,S).           % we are at field [X,Y] at step S
```

The facts of a solution should represent a sequence. This means that there must be no gaps between the steps, i.e., if there is a fact `at(X,Y,S)` for `S` $>$ 0 then there must be a fact `at(XX,YY,S-1)` for some `XX` and `YY`. We represent the solution in figure 1 by the following atoms:

```
at(2,1,0) at(1,1,1) at(2,1,2) at(3,1,3) at(3,2,4)
```

Observe how at step 0 the position coincides with the start field, and at the last step 4 the position coincides with the goal field. If the limit of steps is S the sequence of positions may go from 0 to $S+1$, but it may be shorter, as in the example. We say that a solution is *optimal* if it goes from 0 to T, and for every $U < T$ there is no solution that goes from 0 to U.

Task. Write an ASP encoding such that given an instance, the stable models of the encoding and the instance correspond to the *optimal* solutions of the problem represented by the instance. To compute optimal solutions, use the optimize statements of `gringo`. The two fastest encodings will get **1 bonus** point.

Framework. In the `mino.zip` archive at Moodle you will find some example instances with their solutions. You have to submit a file named `mino.lp` that contains the following line (and no more `#show` statements) so that in the output only the atoms of predicate `at/3` appear:

```
#show at/3.
```

Formalities. You can work on the project 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 **Friday 19th of December, 2014** via YETI (All group members have to create a YETI account!). Be sure to submit your encoding in a file named `mino.lp` containing only lowercase letters.

We will test your encoding with the 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.

Tips:

- Commands to find all optimal stable models look as follows:

  ```
  $  gringo-4 mino.lp example.lp | clasp3 --opt-mode=optN
  ```

  With option `--quiet=1` you can avoid printing non optimal solutions. More `clasp3` options can be found using option `--help=3`.

- 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).