P7: ASP Dungeons

```
. . _ . . . .  1 1 1 1 1 1 1  - - - 2 2 2 2  - - - - - 3 3
W W W . . a .  - - - - - - - 1  - - - 2 2 2 2  - - - - - - 3
W W W W . . .  - - - - - - - 1  - - - - 2 2 2  - - - - 3 3 3
W W W W . . _  - - - - - - 1 1  - - - - 2 2 -  - - - - 3 3 -
W g . . . . .  - 1 1 1 1 1 -  - - 2 2 2 2 -  - - - - - 3 3
W W W W W W .  - - - - - - - -  - - - - - - - -  - - - - - - 3
W W W W W W .  - - - - - - - -  - - - - - - - -  - - - - - - 3
```

IPython Notebook from Lecture

http://nbviewer.ipython.org/urls/dl.dropbox.com/s/idsexzd9zvq09mc/Textbook.ipynb

Walkthrough

Step 1: Download the Potassco Tools

- Navigate to http://potassco.sourceforge.net/
- Find the clingo-4.5.0 binaries for your platform, and download them.
- Unpack them to your project directory, and find the *gringo*, *clingo*, and *reify* executables.

Step 2: Extract AnsProlog Programs from Example Code

- Copy/paste these files from the example IPython Notebook. Make sure the "%%file" line at the top of the notebook cell is **not** included in the file contents.
    - level-core.lp
    - level-style.lp
    - level-sim.lp
    - level-shortcuts.lp

Step 3: Test the Solver

- In your command line shell, execute a command equivalent to the following to make sure you can run the solver (clingo) on some answer set programs. You may need to change your working directory or specify an alternate path to the clingo executable.
- *$ clingo level-core.lp*
- You should see output like this:
    - ```
      clingo version 4.5.0
      Reading from level-core.lp
      Solving...
      Answer: 1
      ```
      ```

```
param("width",10) tile((0,0)) tile((1,0)) tile((2,0)) tile((3,0))
tile((4,0)) tile((5,0)) tile((6,0)) tile((7,0)) tile((8,0)) tile((9,0))
tile((0,1)) tile((1,1)) tile((2,1)) tile((3,1)) tile((4,1)) tile((5,1))
tile((6,1)) tile((7,1)) tile((8,1)) tile((9,1)) tile((0,2)) tile((1,2))
tile((2,2)) tile((3,2)) tile((4,2)) tile((5,2)) tile((6,2)) tile((7,2))
tile((8,2)) tile((9,2)) tile((0,3)) tile((1,3)) tile((2,3)) tile((3,3))
tile((4,3)) tile((5,3)) tile((6,3)) tile((7,3)) tile((8,3)) tile((9,3))
tile((0,4)) tile((1,4)) tile((2,4)) tile((3,4)) tile((4,4)) tile((5,4))
tile((6,4)) tile((7,4)) tile((8,4)) tile((9,4)) tile((0,5)) tile((1,5))
tile((2,5)) tile((3,5)) tile((4,5)) tile((5,5)) tile((6,5)) tile((7,5))
tile((8,5)) tile((9,5)) tile((0,6)) tile((1,6)) tile((2,6)) tile((3,6))
tile((4,6)) tile((5,6)) tile((6,6)) tile((7,6)) tile((8,6)) tile((9,6))
tile((0,7)) tile((1,7)) tile((2,7)) tile((3,7)) tile((4,7)) tile((5,7))
tile((6,7)) tile((7,7)) tile((8,7)) tile((9,7)) tile((0,8)) tile((1,8))
tile((2,8)) tile((3,8)) tile((4,8)) tile((5,8)) tile((6,8)) tile((7,8))
tile((8,8)) tile((9,8)) tile((0,9)) tile((1,9)) tile((2,9)) tile((3,9))
tile((4,9)) tile((5,9)) tile((6,9)) tile((7,9)) tile((8,9)) tile((9,9))
sprite((4,0),altar) sprite((3,0),gem) sprite((1,0),trap)
sprite((2,0),trap)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.021s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.010s
```

Step 4: Capture One Example Output

- Execute a command equivalent to the following to save the output of one random run of the solver to a text file. The “--sign-def=3” argument tells the solver to use a random heuristic and “--out=2” tells the solver to output in JSON format.
- *$ clingo level-core.lp level-style.lp level-sim.lp --sign-def=3 --outf=2 > example.json*

Step 5: Write a Python Program to Parse and Display the Dungeon

- Using the building blocks provided in the notebook, write a small Python program that can visualize the dungeon named as a command line argument. It should be able to be used like this:
- *$ python p7_visualize.py example.json*

Step 6: Test Shortcut-free Dungeon Generation

- Extract the *metaS.lp* file from the notebook.
- Find where the other meta*.lp files are located. Looking inside of the unpacked archive, find a directory called examples/reify. It will provide *meta.lp*, *metaD.lp* and *metaO.lp*.

- Execute a command equivalent to the following. Note the use of *gringo* instead of *clingo* and the inclusion of *level-shortcuts.lp* in the first part of the pipeline. Adjust your file paths as needed. Backslashes indicated escaped line breaks (this is all one single command).
    - *$ gringo level-core.lp level-style.lp level-sim.lp level-shortcuts.lp *
      *| reify *
      *| clingo - meta.lp metaD.lp metaO.lp metaS.lp *
      --parallel-mode=4 --outf=2
- Within a minute, the command should complete. Add "-c width=7" to the arguments of *gringo* if you want to generate a smaller map more quickly (7 is the smallest valid width for these design requirements).
- Run this command again, but append "> example_noshortcut.json" to capture the output in a file.
- Use your visualization program to visualize the generated shortcut-free dungeon.

Step 7: Create a Driver Program

- Create a new Python program, p7_driver.py, that knows how to execute the answer set solver itself. Borrow the example usage of the subprocess module from the solve() function given in the notebook. You may need to adjust the arguments to Popen to be appropriate to your platform.
- Using a command like the following, the user should be able to generate and visualize a new dungeon with a single command:
- *$ python p7_driver.py*

Step 8: Let's Go to the Boardwalk (Let's make the player touch all the traps.)

- Modify level-sim.py so that the player may *only* step on traps in state 2 (after picking up the gem and before dropping it off). Do this by adding an integrity constraint that rejects any solution where a trap tile is touched when S != 2.
- Modify the concept definition in level-shortcuts.lp to also ensure that every trap is touched.
    - First, create a zero-argument predicate that detects if a trap was ever left untouched:
        - *a_trap_went_untouched :- sprite(T,trap); not touch(T,2).*
    - Next, modify the concept definition to require that this is not the case (keep the existing conditions):
        - *__concept :- …, not a_trap_went_untouched.*
- Generate and save a visualization of an example boardwalk map.

Requirements / Grading Criteria

- A valid *example.json* file describing a generated dungeon was generated.

- A dungeon visualization program that consumes JSON files was created.
- A valid *example_noshortcut.json* describing a generated shortcut-free dungeon was generated.
- A driver program that generates and visualizes dungeons was created.
- A driver program that specifically uses the shortcut-free dungeon generator was created.
- A boardwalk map was generated and visualized.