Wumpus World Project – Status Report

# Design/Implementation

## ***WWRationalAgent***

WWRationalAgent is my modification of WWAgent and includes both additions and changes to the original implementation of WWAgent.

### **Additions/Changes:**

* Member Variables (and values at initialization)
  + *self.last\_position = (3, 0)* 
    - Used to avoid retracing steps when all adjacent cells have been explored
  + *self.last\_move = None*
    - Used to check if agent picked up gold on last move
  + *self.kb = set()* and *self.models = set()*
    - Might be redundant and/or unnecessary, see **Issues** section for more info…
    - self.kb is the agent’s knowledge base: everything it’s perceived so far
    - self.models is the possible models of the dungeon given self.kb
  + *self.journey = {(3, 0): 1}*
    - Keeps track of the cells the agent has visited so far. Used to determine whether a dungeon is solvable.
* Member Functions
  + *tt\_entails(self, alpha), tt\_checkall(self, alpha, symbols, model), pl\_true(query, model)*
    - Implements the truth table enumeration algorithm
  + *create\_map(self, query=False)*
    - Creates a 4x4 map of Cell/QueryCell objects (explained later)
  + *update\_kb(self)*
    - Adds QueryCells to *self.kb* for each percept at the agent’s given position
  + *gen\_models\_from\_map(self)*
    - Adds all possible locations of wumpuses and pits, given *self.kb*, to *self.models*
  + *get\_possible\_moves(self)*
    - Returns a list of the cell coordinates adjacent to the agent’s current location
  + *purge(self, possible\_moves)*
    - Given a list a coordinates adjacent to the agent’s current position, calls *tt\_entails* to remove dangerous, possible next positions
  + *get\_action(self, next\_pos)*
    - Returns the proper action based on the agent’s current direction and location
  + *exit\_dungeon()*
    - Called when the agent determines the dungeon is not solvable; prints message to command line and returns ‘climb’

## ***Cell, QueryCell***

Dungeon cells are implemented as *Cell* objects; with *QueryCells* being a *Cell* with only one percept.

### **Important Cell Methods**

* *adjacents(self, tt=True)*
  + Depending on value of *tt* either a list of adjacent cells to this cell, or coordinates of adjacent cells

### **QueryCell Info**

QueryCells are essentially columns of the truth table, with each having a true percept at initialization. Those values are negated in enumeration as part of the second half of the return statement.

# Issues

As of now, there’s only one bug: sometimes a WWRationalAgent will eliminate all possibly dangerous cells except for the one containing the actual danger. In other words, sometimes a WWRationalAgent will eliminate cells next to a pit, but not the pit itself. My theory is that this is a result of how I have implemented QueryCells and/or *self.kb/self.models*. Once I’ve solved this bug, my WWRationalAgent should always find the gold in the fewest moves possible, or will deem the dungeon unsolvable.