A1 & Wumpus World Notes
CS4300 AI, Fall 2016

Assignment 1 Statistics

Your assignment is to:

- develop an agent function that randomly (uniformly) selects actions (from FORWARD, ROTATE_RIGHT, ROTATE_LEFT) in the Wumpus world (starting at x=1, y=1 and facing right).
- run 2000 trials and determine the mean and variance of:
    - the percentage of trials in which the gold cell is reached, and
    - the total number of actions taken before dying.

The statistics are found by running 2000 number of trials and recording the above values for each. Then, find the mean and variance. Given that μ is the mean and $S^2(n)$ is the variance of n trials, then find the confidence interval for each; e.g., for 95% over the 200 trials:

$$X(n) = \mu \pm t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2(n)}{n}} = \mu \pm 1.645\sqrt{\frac{S^2(n)}{n}}$$

Board Layout

The board is 4x4 where each cell represents a room in Wumpus world (see Figure 1). In the figure, the

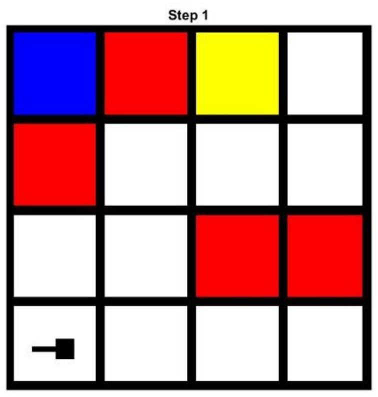

Figure 1. An Example Board Starting State (the A1 board).

open cells are shown in white, the pits are in red, the Wumpus is blue, and the gold is in yellow. It is possible for the Wumpus to be in the same room as the gold as shown in Figure 2.

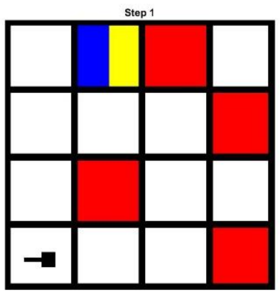

Figure 2. An Example Board with Wumpus in same Room as Gold.

A board can be generated as follows:

```
>> clear all
>> board = CS4300_gen_board(.25);
```

The indexing of the rooms is given by the Cartesian coordinates as shown in Figure 3.

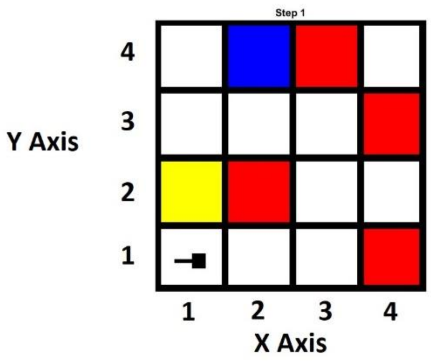

Figure 3. The Coordinate System of the Wumpus World.

Agent Description

The agent is located in Room [1,1] as indicated by the hammer like shape (the big end is the heading direction). The agent has associated state as follows:

- X location
- Y location
- Heading (4 possible: Right (0), Up(1), Left(2), Down(3))
- Alive: Boolean value of 1 if agent is still alive
- Gold: has grabbed Gold
- Climbed: has climbed out of Wumpus World (only possible from Room [1,1])

- Succeed: if has grabbed Gold and climbed out of Wumpus world

In terms of behavior, the agent is instantiated as a Matlab function which takes as input a variable called *percept*, and delivers an output called *action*. *percept* is a 1x5 Boolean vector whose elements correspond to:

1. Stench: Wumpus is in a 4-neighbor room
2. Breeze: A Pit is in a 4-neighbor room
3. Glitter: The Gold is in the current room
4. Bump: The agent tried to move forward, but ran into a wall
5. Screamed: The Wumpus was killed by the arrow

action can be one of the following:

1. FORWARD: move one unit in the direction of the agent heading (if possible)
2. ROTATE_RIGHT: change agent heading 90 degrees to the right
3. ROTATE_LEFT: change agent heading 90 degrees to the left
4. GRAB: grab Gold if it is in current room
5. SHOOT: shoot an arrow in the heading direction (it will go until it hits the Wumpus or the edge of the board)
6. CLIMB: if in Room [1,1], then move out of Wumpus World; else no effect.

Defining an Agent Behavior Function

CS4300_Example1 demonstrates a simple agent behavior: move forward (right to [2,1]), rotate left, rotate left, move forward (left to [1,1]), climb out.

```
function action = CS4300_Example1(percept)
% CS4300_Example1 - simple agent example
%    It moves right, then left, then climbs out
% On input:
%     percept (1x5 Boolean vector): percept values
%      (1): Stench
%      (2): Pit
%      (3): Glitters
%      (4): Bumped
%      (5): Screamed
% On output:
%     action (int): action selected by agent
%       FORWARD = 1;
%       ROTATE_RIGHT = 2;
%       ROTATE_LEFT = 3;
%       GRAB = 4;
%       SHOOT = 5;
%       CLIMB = 6;
% Call:
%     a = CS4300_Example1([0,1,0,0,0]);
% Author:
%     T. Henderson
%     UU
%     Summer 2015
%
persistent state
```

```
FORWARD = 1;
ROTATE_RIGHT = 2;
ROTATE_LEFT = 3;
GRAB = 4;
SHOOT = 5;
CLIMB = 6;
 if isempty(state)
    state = 0;
end
switch state
    case 0
        action = FORWARD;
        state = 1;
    case 1
        action = ROTATE_LEFT;
        state = 2;
    case 2
        action = ROTATE_LEFT;
        state = 3;
    case 3
        action = FORWARD;
        state = 4;
    case 4
        action = CLIMB;
end
```

The agent is defined as a simple finite state machine and uses a persistent variable to maintain state information across invocations by the simulation driver. Note that every time the function is used in a new simulation, the 'clear CS4300_Example1' command must be given to clear the persistent variable; this will clear all variables currently in the top level Matlab environment. To use this function, a driver like CS4300_WW1 is used.

```
function trace = CS4300_WW1(max_steps,f_name,board)
% CS4300_WW1 - Wumpus World 1 simulator
% On input:
%     max_steps (int): maximum number of simulation steps
%     f_name (string): name of agent function
%     board (4x4 array): Wumpus world board
% On output:
%     trace (nx3 int array): trace of state
%       (i,1): x location
%       (i,2): y location
%       (i,3): action selected at time i
% Call:
%     t=CS4300_WW1(5,'CS4300_Example1',[0,1,0,0;0,0,1,0;0,3,2,0;0,0,0,0]);
% Author:
%     T. Henderson
%     UU
%     Summer 2015
%

agent.x = 1;
agent.y = 1;
agent.alive = 1;
agent.gold = 0;  % grabbed gold in same room
agent.dir = 0;  % facing right
agent.succeed = 0;  % has gold and climbed out
agent.climbed = 0; % climbed out
```

```
trace(1).board = board;
trace(1).agent = agent;
trace(1).action = 0;

step = 0;
done = 0;
bumped = 0;
screamed = 0;

while step<max_steps&done==0
    step = step + 1;
    percept = CS4300_get_percept(board,agent,bumped,screamed);
    action = feval(f_name,percept);
    [board,agent,bumped,screamed] = CS4300_update(board,agent,action);
    trace(step+1).agent = agent;
    trace(step+1).board = board;
    trace(step+1).action = action;
    if agent.alive==0|agent.succeed==1|agent.climbed==1
        done = 1;
    end
end
```

The *trace* data structure is indexed by time step and at each step saves a field containing the board, the agent and the action selected by the agent. A function is provided to produce a movie of the trace:

```
>> clear CS4300_Example1
>> t = CS4300_WW1(50,'CS4300_Example1',[0,1,0,0;0,0,1,0;0,3,2,0;0,0,0,0]);
>> M = CS4300_show_trace(t,1);
```

Yielding the movie M which can be viewed using:

```
>> movie(M)
```