CS 5100 Spring Assignment 2 Wumpus World Report
Rui Zhang NUID: 001989420

1. Wumpus World Implementation

Since the world is a grid format like the TicTacToe we finished before, it's straightforward that keep using a two-dimension array to represent the world is a great choice here. And the process of the agent to explore, find as many as possible gold and reach to goal when appropriate is much more complicated than TicTacToe's, so I chose to use GUI to show the process to be more clear and vivid.

I defined 4 classes here, WumpusGUI, WumpusWorld, Agent, TheoremProver.

Wumpus GUI is the main controller of the game and is the entrance of game. It creates widgets of UI, the world that GUI to show and an Agent that will perform exploration in the given world.

WumpusWorld load configuration from given world file, and initialize the proper data structure to store the world information.

Agent is the key role in this game, it implemented a hybrid algorithm to make sure the agent to dig as many as possible gold before leave the world. Also, it makes sure only if we really need to kill wumpus to explore more cells in the world, we kill wumpus. Because killing a wumpus will result a 100 score lose, which is critical to the find score.

TheoremProver is responsible for proving the theorem based on knowledge. Both theorem and knowledge are generated by Agent and sent by Agent. It use python subprocess to invoke the prover9 binary to finish the job.

2. Prover9 FOL Model Design

According to the game rules, if there is a wumpus, there will be stenches adjacent to it. If there is a pit, there will be breezes adjacent to it. So, it's straightforward that we need a Adjacent(x, y) function to tell if position x is close to y. Because we are using two-dimension array to represent the world, so every position is specified by (x, y) coordinates. To represent the "close to" logic, we need functions NearbyX(x1,x2) and NearbyY(y1,y2) to tell if a position (x1,y1) is close to (x2,y2). Therefore, we need the most basic FOL functions like Adjacent(X1,Y1,X2,Y2) , NearbyX(X1,X2) and NearbyY(Y1,Y2)).

What's more, we need to deduce a position whether there is wumpus or not and whether there is pit or not based on Breeze and Stench information. So, we need Breeze(x,y) function to record that in (x,y) there is a Breeze, Stench(x,y) therefore means there is a Pit in (x,y).

The last and the most important thing is we are going to use the current knowledge to prove our theorem, so the deduction rules are the most important thing in our FOL model. As I've mentioned above, if there is a wumpus, there will be stenches adjacent to it. If there is a pit, there will be breezes adjacent to it. So, I created:

all X1 all Y1 all X2 all Y2 ((-Stench(X1,Y1) & Adjacent(X1, Y1, X2, Y2))-> -Wumpus(X2, Y2)).

all X1 all Y1 all X2 all Y2 ((-Breeze(X1,Y1) & Adjacent(X1, Y1, X2, Y2))-> -Pit(X2, Y2)).

all X1 all Y1 (Breeze(X1,Y1) -> (exists X2 exists Y2 (Pit(X2,Y2) & Adjacent(X1, Y1, X2, Y2)))).

all X1 all Y1 (Stench(X1,Y1) -> (exists X2 exists Y2 (Wumpus(X2,Y2) & Adjacent(X1, Y1, X2, Y2)))).

to be used for deduction.

3. Problem Encountered And Solutions

This assignment is really challenging and complicated. I've encountered a lot of problems along the way to define the solution I am having now.

The first problem is what interface we need to use to show the game. Console output maybe the simplest, but it's really hard to look at the text output to figure about what an agent just did. Using a GUI may be a good idea. However, I'd never done any python GUI before and knew nothing about the built-in Tkinter. So, I spent a lot of time learning online and practicing. Finally, the UI problem is solved.

Then it comes to defining the explore algorithm for Agent to traverse the world. I had some troubles

as well. Although the A* is suggested in the assignment page, but in this game, we are almost blind to the world at the beginning, we don't know how many gold are there and where they are. So, we basically have no target. In this case, A* may not be a good idea, because we all know that A* is for targeted searching. Then, what search algorithm is needed here? After discussing with others, I figured out that there is no way that I only use one kind search algorithm to finish the whole job. Because when traverse the world, we don't know where to go, but after we knew where wumpus is and we want to go to kill it, we need to target at the close cell to it, and go to there with the lowest cost. Also, when Agent figured out there is no more cells can be explored, it has to target at the goal cell to leave the world. So, a hybrid algorithm, which combines general searching for traversing the world and informed search for moving to specific target, was created in the solution.

For traversing the world, we can just use basic DFS or BFS here to complete the job easily. However, after some experiments, I found out DFS here is easier to be implemented in this situation. Because for BFS, we need to visit the all parent nodes before visiting the children nodes, however, when the parent nodes are not adjacent to each other, we need to go back to the super node of those parent nodes. To achieve this, we need some complicate tracing method. However, DFS will be much easier because in DPS we visit the child node first, then backwards to their parent. Therefore, in the end, I changed my traversing algorithm to DFS.

For going to the target, I thought about using A*, but at the time we have a target to go, we've already traverse part or whole world. And not all the cells in the world are safe to go through. So, A* seems like not a good fit because the environment is exposed to us. Maybe some other search algorithm is better here. Then I came up with the idea using informed search with heuristic to finish the job.

4. Creativities In The Solution

The most creative things in my solution are the informative GUI and the efficient hybrid search algorithm implemented for Agent.

To check out the informative GUI, just play with the program with more maps. You will find out it's really easy to monitor and trace the Agent.

To show the hybrid algorithm, I'd like to use some pseudo-code here:

```
def play:
   explore_world_using_DFS()
   while no_more_cell_is explorable():
      if all_cells_explored():
         go_to_goal()
      deduce_more_safe_cells_from_unknown_cells()
      if more_safe_cells_found():
         go_to_safe_cell()
         explore_world_using_DFS()
      else:
         if wumpus_must_be_killed()
            kill_wumpus()
         if wumpus_cell_safe():
            go_to_wumpus_cell()
            explore_world_using_DFS()
      else:
         go_to_goal()
```

5. Insight Of The Assignment

In this assignment, I learned how to use theorem prover like prover9 to prove theorem based on knowledge base and this made me understood the FOL better because I had don't a lot of FOL reasoning along the way. Also, I did a lot of research in those search algorithms to figure out the best fit for the solution.

In short, although in the process, things got crazy and I got frustrated when I was debugging the program, but it's really a great practice for learning AI.