Wumpus World
Due October 27th
Will be discussed in class on the 25th

Your assignment is to build a knowledge-based agent who can solve problems in the Wumpus World.

You will have access to code that builds and manages a world for you. It will take actions as input and return the state of the world that results from those actions.

In general, you can take actions that change the direction you are facing, move forward and either pick things up or shoot an arrow. Once you have scored some points and returned to your home, you can also leave the game.

Moves that change your orientation: Up, Down, Left, Right

The move that changes your position: Step

The move that allows you to exit the board: Exit

Actions that change the world aside from your position: PickUp**,** Shoot

You invoke these actions by calling the function take_action (in the file updatewumpus.py) that takes the name of a world (more on that shortly) and a move. It returns a list of strings that describes the current perceptions that are available to you:

(smell, air, glitter, bump, scream, location, orientation, status, score)

The possible values of these perceptions are:

Smell - clean|nasty
Air - calm|breeze
Glitter - bare|glitter
Bump - no_bump|bump
Scream - quiet|scream
Location - unique identifier for current square
Orientation - the direction you are facing
Status - living|dead|won
Score - current score

Before you start the game, you need to initialize a "world." You do this by calling intialize_world, a function that has no arguments. This returns the name of a new world that you need to use when calling take_action. This function builds a folder ("WumpusWorldDataFolder ") where the world data will be saved. It does not

delete any of the worlds it builds (but they are tiny) so you might want to clear this folder occasionally. It didn't want to include any code that removes files/folders.

You will need to download updatewumpus.py and include it in your agent file:

```
include updatewumpus
```

You will need to initialize a world before you start to play and hold onto the name of that world:

```
world_name = updatewumpus.intialize_world()
```

At run time, you will get a message like this:

```
Initializing your new Wumpus world!
Your new world is called: Wumpus3231
You are starting in Cell 11, looking to the Right.
You are starting with 0 points, 1 arrow(s).
You are alive.
```

Now you can take actions by calling take_action:

```
updatewumpus.take_action(name, "Up")
```

or

```
updatewumpus.take_action(name, "Step")
```

take_action will return a precept list every time it is called that looks like this:

```
['clean', 'calm', 'bare', 'no_bump', 'quiet', 'Cell 11', 'Up', 'living', 0]
```

or

```
['nasty', 'calm', 'bare', 'no_bump', 'quiet', 'Cell 12', 'Up', 'living', 0]
```

It will also provide you with messaging so you can track the game:

```
**********

Turing to face Up
Perception = (clean, calm, bare, no_bump, quiet, Cell 11, Up, living, 0)

**********

Taking a step
```

Moving to Cell 12
There is a nasty smell in here
Perception = (nasty, calm, bare, no_bump, quiet, Cell 12, Up, living, 0)

You can also get the names of adjacent cells by calling look_ahead. It takes the name of the world and returns a list of the cells that are next to you.

updatewumpus.lookahead(world_name)

returns

['Cell 13', 'Cell 11', 'Cell 22']

The idea behind this is you can use this information start to ASSERT facts about the world as you play.

Once you die, the world will no long update and your status will change to "dead" in you perception list. Also:

You cannot PickUp the gold if there is no gold.
You only have one arrow to Shoot.
You can only Exit if you have scored some points and are at your home.
Once you have killed the Wumpus, you can enter its room without dying.

The assignment.

You goal is to build an agent that is able to move safely through its world, collect gold, kill the Wumpus, and then exit.

Given that you are starting with no knowledge of the world, you are going to have to have your agent figure things out as it goes along. You will need to build up a set of inference rules that will fire as new facts are known.

For example, if you are on a square with no breeze, then you can add this fact to the data, add facts about what squares are adjacent to it, and then infer that those squares have solid floors.

That is, ASSERT:

Calm(Cell 11)
Adjacent(Cell 11, Cell 12)
Adjacent((Cell 11, Cell 21)

And then use a rule like:

Calm(?x) V Adjacent(?x ?y) => Solid(?y)

to make the right inferences.

You will need to use your knowledge base to figure out which squares have solid floors and no Wumpus, are safe (solid floor and no Wumpus), and where the Wumpus is so you can shoot it. AS you wander, you should probably also be thinking about how to save information about what constitutes as good path back to the start.

The approach we are taking is to use a knowledge base as the repository of what is true (i.e., known by the system) and another system (a planner) to make decisions based on what it knows and what it wants to accomplish.

The knowledge side of things requires a repository of facts and a repository of rules. As we discussed, as new rules are added, we check to see if they allow you to infer anything given the fact that you know. Likewise, as you add new facts, you need to check to see if any rules can be fired.

In order to access this information you will need to be able to:

> **ASSERT** facts and rules. (This room has no breeze.)
> **ASK** if things you care about are true? (Is this room I want to go into Safe?)

Your agent should have specific goals it is trying to accomplish (pick up the gold) and know that it will have to take actions in order to set up the conditions that makes this possible (stand in the same room as the gold).

The challenge is that you will need to be able to not just add facts but also draw inferences from them. Your inference rules will need to have variables so that they are general. You will need to be able to **MATCH** facts against patterns in your inference rules. And when you draw an inference, you will need to **INSTANTIATE** the result and thus create a new fact with all of the variables replaced with the elements from the fact that was matched against it.

And you will need to be able to do the same sort of matching for **ASK** so you can look for the answers to general queries.

There is a very simple implementation of **MATCH** and **INSTANTIATE** in the file FOPC.py.

On Tuesday, we will discuss this in class and you will get the second piece of this assignment that will be a somewhat expanded Wumpus world in which you will need to kill the Wumpus in order to get to your goal.

Deliverables

You will be handing in your code and the trace generated by the world systems that results when you run it.