IAI. Assignment 1

Author: Ruslan Sabirov
Group: BS17-02

Task #1

1. [5pts (PEAS) + 5pts (Environment)] Aeroflot is designing a newly automated check-in counter. The counter should be able to check from the PNR number, passport scan, etc. look up the flight. It will then verify the visa if required as well as check in baggage and print any required tags and passes. They need a PEAS and a description of the environment (ignore Known/Unknown) with all necessary justifications.

Agent - automated check-in counter
Performance measure: passenger is successfully registered
Environment: Airport, check-in zone
Actuators:

- screen (to show instructions)
- printer (to print required tags and passes)
- conveyer belt (to pass luggage to further points)

Sensors:

- ticket scanner (to recognize PNR)
- passport scanner (to identify passenger)
- visa scanner (to verify it if it is necessary)
- web-cam with face recognition (to identify passenger and compare his photo with images from passport and visa)
- weighing scale (to weight the luggage)

Environment - check-in zone of Airport
Properties:

- fully observable - agent does not miss elements which are relevant to the decision making process (ticket, passport, visa information).
- multiple agent - there are more than one automated check-in counters (for different avia companies, different passenger classes).
- deterministic - the next state of the environment is completely determined by the current state and the action performed by the agent (e.g., no printed pass + printing pass = printed pass).
- episodic - the agents experience is divided into atomic episodes, in each episode agent receives a percept and performs a single action (scans documents and then print items, pass the luggage).
- static - environment cannot change while an agent is thinking about its actions.
- discrete - there are discrete states in the environment (scanned passport, passed luggage, printed pass and so on).

Task #2

2. [5pts+5pts] Fallout 76 has had major problems with it's NPCs. They would like you to make a PEAS for a new edition of the game – examining the non-vendor NPC, as well as give them an idea of the environment (ignore Known/Unknown). Be sure to justify your decisions.

Agent - Mister Prize bot
Performance measure: give the player things
Environment: Appalachia
Actuators:

- give the player things
- speaker (to talk to the player)
- jet engine (to soar and fly)

Sensors:

- eyes (to see if player is nearby)

Environment - Appalachia
Properties:

- partially observable - sensors of the agent do not give it access to the complete state of the environment (what is happening in 1 kilometers far).
- multiple agent - there are more than one Misters Prizes bots (located at different places).
- stochastic - the next state of the environment is not determined by the current state and the action performed by the agent (if agent will compliment the player, after that someone can hit or not hit the bot).
- episodic - the agents experience is divided into atomic episodes, in each episode agent receives a percept and performs a single action.
- dynamic - environment can change while an agent is thinking about its actions (weather could change).
- continuous - environment smoothly act over time.

Task #3

3. [20pts] Create a set of Prolog rules which implement Wumpus World on a 5 by 5 space. There can be any number of pits, one gold, and one Wumpus. The player may be assumed to begin in the bottom left-hand square (1,1). The code should be able to print out all possible solution paths.

Prolog code is in file wumpus_wold.pl
Run with one of the queries:

```
?- start(possible).
?- start(impossible).
?- start(without_pits).
```

Task #4

4. [10pts] Write a short report – about 2 pages – testing your code on a variety of inputs. Show both possible and impossible boards (note – you need to figure out what makes for an impossible board). Show an example where your agent kills the Wumpus.

Since the task was to print all possible solutions path and there was no any other restriction, I decided to use backtracking (DFS) algorithm.

Finding all paths could run infinitely, because we can go from ceil A to ceil B, and then back to A, and then back to B and so on. That's why we should somehow restrict our algorithm running time, I used upper bound for recursion depth (number of cells visited). To avoid A-B-A moves I also added list of visited nodes to recursion arguments: my agent does not enter to cells which was already visited.

For testing I used 2 boards: impossible and possible. By impossible I mean that is not possible to reach the gold without falling into a Pit (there is no save paths).

			P	
W	G	P		
				P
S		P		

a) Possible board

				P
P	G			
	P	W		
S		P		

b) Impossible board

Legend: S - start, G - gold, P - pit, W - wumpus

a) Possible board

We run it with query `?- start(possible).`

If maximal depth of recursion is equal to 15, we will get 6 paths:

- [[1,1],[1,2],[2,2],[2,3]]
- [[1,1],[1,2],[2,2],[3,2],[4,2],[4,3],[5,3],[5,4],[5,5],[4,5],[3,5],[3,4],[2,4],[2,3]]
- [[1,1],[1,2],[2,2],[3,2],[4,2],[4,3],[5,3],[5,4],[5,5],[4,5],[3,5],[2,5],[2,4],[2,3]]
- [[1,1],[2,1],[2,2],[2,3]]
- [[1,1],[2,1],[2,2],[3,2],[4,2],[4,3],[5,3],[5,4],[5,5],[4,5],[3,5],[3,4],[2,4],[2,3]]
- [[1,1],[2,1],[2,2],[3,2],[4,2],[4,3],[5,3],[5,4],[5,5],[4,5],[3,5],[2,5],[2,4],[2,3]]

b) Impossible board

We run it with query `?- start(impossible).`

Program does not print anything since gold is unachievable (there is no path from [1,1] to location of gold).

W	G			
S				

c) Without pits

c) Without pits

We run it with query `?- start(without_pits).`

It will print 199 paths to gold

d) Custom board

To create custom board you can add one more predicate named `initialize_world(World):`

```
initialize_world(World) :-
  World = name_of_custom_world, %name of the world
  assert(wumpus_location(1,3)), %location of wumpus
  assert(wumpus_health(alive)),
  assert(gold(2,3)), %location of gold
  assert(pit(3,1)). %any number of pits
```

and run query `start(name_of_custom_world).`