Wumpus World

In this project we tried the solve the Wumpus world problem by using FOL in c++. The main reason we used c++ was to use SFML libraries to create GUI. As C++ offers lots of data structures we tried to keep it simple by using only boolean variables to make it look like Prolog language and to make logical deductions easily.

Other than the structure that has been used to create the GUI, the AI side uses only two main classes. Percept and Agent. Percept class stores the Wumpus world environment in a matrix divided as cells. Agent class is the rational agent, that tries to find the gold by actions and logical deductions. There are two instances of Percept class. The first one stores all the data in the world that is already predefined. The second instance belongs the the agent. The agent updates its own map of percept as he reads from the cell that he is currently on.

The AI part mainly occurs on agent's map. As a programming characteristic, the agent is pessimistic. A s he sees a percept like Breeze or Smell, he automatically assumes that all cells adjacent the that one is either has a pit or Wumpus, respectively, unless told otherwise. There are AI operations which understands if a cell definitely has a Wumpus or pit, otherwise that a cell definitely has no pit or/and wumpus on it. The cells, that has neither of them, are considered safe, and marked on agent's map. The agent's map also marks if a cell has been visited. As a agent comes to a crossroad it tries to go to a cell that has been marked as safe before, and not has been visited. With this option the agent can explore the world faster and updates it's own map, which gets more similar with current world with each iteration.

One of the failures we had when we started the project was using the prolog interface for c++ language. There are examples in which people managed the connect them with each other but in our case, we failed at two different operating systems. In windows the amzi prolog requires to set a server, which later will be connected to the c++ phase. In linux system swi prolog requires a header file for the c++ project but we couldn't compile it.

To revert this failure we had, as mentioned before, we tried to make our code look like a Prolog code by only using boolean values. As one of the logic programmers I find it very educational because that requires more complex algorithms by using less inputs. The booleans can only hold two values , true and false, so either one has to define more booleans variable to keep more info about the current state of the world or use the given variables in harmony to make better deductions.

The AI algorithm also helps us to understand how we , as humans, see the world. As we tried our agent on test cases we have encountered with situations in which also the human mind was unable to solve. But other than these situations we tried to put ourselves in the place of the agent and implemented our own thinking ways. As humans can think simultaneously , it was very cruel for the AI algorithm the write the respective functions in the correct order because every aspect is dependent on the previous one. To prevent our agent making the wrong moves, we first tried to read every data from the outer world, than merge it with the prior knowledge and agent's own experiences before making a move. In this case the prior knowledge is the rules of the world(every cell adjacent to pit is breeze etc.), and the experiences are the agent's percept on other cells that is stored in agent's map.

If the agent is stuck on a cell, it moves to the previous one by using a simple coordinate stack and finds an unvisited safe cell. So at the meantime the agent also solves a maze.