## 1. Implementation approach and overview of code

In essence, the rational agent moves to the safe squares around them and prefers the ones that they didn’t frequently visit. I used visitedmap to keep track of the visits to a square. When deciding to move from a square, the agent looks at their safe choices (0 in danger map) and chooses the least frequently visited one. At the start, I added some useful functions such as the inside_map function so that if the coordinates are not inside a map, the danger map will return index out of bounds. Truth table enumeration (as a simple form of model checking) is used to enumerate all possible propositional models in the danger_estimate function, which is later used in da_way function to choose the next square and ultimately in the action function to pick the next action of the agent. For example, the statement “a square is a pit” implies the statement “all of its adjacent squares contain breeze.” The negation of this is: “one of the adjacent squares doesn’t contain breeze” implies the statement “the square is not a pit.” The same logic applies for stench. In other words, for a square to be called a pit (or Wumpus) every square around it should contain breeze. So, in order for a square to be safe: 1) at least one of its adjacent squares should not contain breeze or stench; 2) one of its adjacent squares contains breeze or stench and the other does not. The da_way function is used to choose the next square, determining the safest move of the moment. First, the agent searches for a safe choice. If there is no safe choice or if they moved too much in safe squares, they decide to take a risk – this is done by using T. T denotes a set containing the squares in the danger map that the agent can move to. After some iteration, 5 is added to the set so that the agent can take risk and go to probable pits. In the action function, the danger map is updated. This is where everything comes together for the agent to make a move (choose direction, etc.).

## 2. Show that agent is rational

When I run the program many times, the agent is usually apt at making the right move to avoid danger to find gold and exit. But depending on the luck of the circumstances, the agent fails sometimes even from the start. It also takes some time for the agent to make a decision on the next move.

Performance Run Comparison:

| **Run** | **Performance** | |
|---|---|---|
| | **Example** Agent | **My** Agent |
| 1 | -1007 | -8 |
| 2 | -7 | -31 |
| 3 | -1016 | -30 |
| 4 | -1071 | -8 |
| 5 | -1015 | -1002 |
| 6 | -1002 | -31 |
| 7 | -1003 | -63 |

| | | |
|---|---|---|
| 8 | -1004 | -1077 |
| 9 | -1026 | -77 |
| 10 | -1011 | -8 |
| 11 | -59 | -3 |
| 12 | -1026 | -9 |
| 13 | -1005 | -8 |
| 14 | -6 | -8 |
| 16 | -1007 | -17 |

The rational agent did not die in most runs whereas the example agent did.