CS 511: Artificial Intelligence II
Project 1: Simple Reflex Agent

`Agent Theseus511`

Spring 2025

1 Design

This project implements a **simple reflex agent (SRA)** that operates in the wumpus world. Specifically, it implements the agent function

$$f_{\text{SRA}} : P \to A$$

that maps the current percept to an action. There are 32 possible percepts and the implementation condenses these cases to just 5 distinct ones.

The agent function is **nondeterministic** with a probabilistic choice of the action for some cases of percepts. Before describing the condition-action rules, one notable design choice is the use of only left turns and not right turns. It is observed that weighing right and left turns equally causes the agent to keep nullifying left turns by successive right turns, and right turns by successive left turns. This causes a wastage of moves hence decreasing the score. Since a right turn is nothing but three consecutive left turns, I conjectured that using only left turns in the implementation is optimal for maximizing the score. If one thinks about it, an optimal strategy to maximize the score can be to keep turning on the same spot (or keep bumping into a wall!) if a `breeze` or stench is perceived, and never move to another square. The agent never dies with this strategy, which never causes the score to below -50 assuming that a limit of 50 time steps is imposed. However, this is not a general strategy since if the time limit is extended to 50,000 time steps then the score will almost consistently be $-50{,}000$, which is clearly not optimal. Hence, another important design choice is to maximize the agent's exploration of the world by generally weighing `GO_FORWARD` more than `TURN_LEFT`. Also, for the choice of turning direction, left is chosen over right since the agent starts in $(1, 1)$, facing east, and turning left increases the chance of exploring the world. The five cases are described next, where the wildcard "`_`" is used to denote a "catch-all":-

1.1 Condition-Action Rules

1.1.1 Case `<_,glitter,_,_,_>`

The agent GRABs deterministically when it perceives `glitter`.

1.1.2 Case `<bump,none,_,_,_>`

The agent `TURN`s`_LEFT` deterministically when it perceives a `bump`.

1.1.3 Case `<none,none,_,stench,_>`

When the agent perceives a `stench`, the wumpus can either be in the square in front, or to the left, or to the right of the agent. The primary action is to `SHOOT`, which by default shoots in front, but there should also be a chance to `SHOOT` to the left or the right i.e. to turn and shoot. If turning and shooting are the only action choices here, then the agent never moves from the square hence hindering exploration. This leads to the design choice where `SHOOT` is weighed more than `TURN_LEFT` and `GO_FORWARD`. Ideally, `GO_FORWARD` should be

weighed more than `TURN_LEFT` as explained above, however in this case, they are weighed equally since I experimentally conjectured that a random choice between turning and going forward increases chances of evading the wumpus. Quantitatively, the probabilities are assigned as `SHOOT` $\mapsto 0.65$, `GO_FORWARD` $\mapsto 0.175$, `TURN_LEFT` $\mapsto 0.175$.

1.1.4 Case <none,none,breeze,none,_>

When the agent perceives a `breeze`, the possible choices for actions are `GO_FORWARD` and `TURN_LEFT` since the agent should not keep turning on the same spot. Ideally,`GO_FORWARD` should be weighed more than `TURN_LEFT`as explained above, however in this case, they are weighed equally since I experimentally conjectured that a random choice between turning and going forward increases chances of evading the pit. Quantitatively, the probabilities are assigned as `GO_FORWARD` $\mapsto 0.5$, `TURN_LEFT` $\mapsto 0.5$.

1.1.5 Case <none,none,none,none,_>

When the agent perceives nothing or just a `scream`, the design aims to maximize the agent's exploration of the world. Hence, `GO_FORWARD` is weighed more than `TURN_LEFT`. Quantitatively, the probabilities are assigned as `GO_FORWARD` $\mapsto 0.65$, `TURN_LEFT` $\mapsto 0.35$.

2 Getting Started

The simple reflex agent is implemented in Scala. The implementation is contained in the files `AgentFunctionImpl.scala` and `SimpleReflexAgent.scala`, and the code is well-documented for reference. The project directory contains a Makefile that automates building and running the project. The Makefile runs the project with the options `nonDeterministicMode` and `randomAgentLoc` set to `false`. The Makefile contains a `check` target that checks the system for the necessary tools (`scala`, `java`). It is recommended that the project is run after checking for the necessary tools as:-

```
$ make check
$ make #or "make run"
```

The project was tested using:-

- **Scala Version:** 3.6.3
- **Java Version:** OpenJDK 22.0.1

3 Evaluation

The project repo contains a shell script `wumpus_eval.sh` that can be used to evaluate the agent. The script can be run using `-h` to display a help message. In general, the script takes two arguments: `num_trials` with option`-n` and `output_file` with option `-o`. The argument `num_trials` specifies the number of times the simulation is run and all the resulting scores are compiled in the file specified by the argument `output_file`. The arguments are optional; the default values for `num_trials` and `output_file` are 10 and "wumpus_eval.txt" respectively. The script prints out the average score over the `num_trials` simulations run, i.e. the average of all the scores written to `output_file`. The project must be compiled (using `make build`) before running the script for it to work. The usage of the evaluation script can be summarized as:-

```
$ ./wumpus_eval.sh [-n <num_trials>] [-o <output_file>] [-h]
```

Note: The script uses the Unix shell utility `bc` for arithmetic computations. Hence, it should be installed for the script to work.