**Exercise 1 :**

You are given the following four domains:

i. A group of people playing poker.

ii. A person driving a car.

iii. A machine detecting chocolate bars weighing less than 50g.

iv. A doctor performing a medical diagnosis.

Classify each of the domains above along domain the properties that we discussed in the lecture by entering yes/no in the cells of the following table:

| | Accessible | Deterministic | Episodic | Static | Discrete | Single agent |
|---|---|---|---|---|---|---|
| **Poker** | | | | | | |
| **Car** | | | | | | |
| **Machine** | | | | | | |
| **Doctor** | | | | | | |

---

**Solution:**

| | Accessible | Deterministic | Episodic | Static | Discrete | Single agent |
|---|---|---|---|---|---|---|
| **Poker** | *no* | *no* | *no* | *yes* | *yes* | *no* |
| **Car** | *no* | *no* | *no* | *no* | *no* | *no* |
| **Machine** | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| **Doctor** | *no* | *no* | *no* | *no* | *no* | *yes* |

**Exercise 2 :**

For each of the following assertions, say whether it is true or false and give a short explanation in 1 or 2 sentences for each of your answers.

(a) An agent that senses only partial information about the state cannot be perfectly rational.

**Solution:** False. Perfect rationality refers to the ability to make good decisions given the sensor information received.

(b) There exist task environments in which no pure reflex agent can behave rationally.

**Solution:** True. A pure reflex agent ignores previous percepts, so cannot obtain an optimal state estimate in a partially observable environment. For example, correspondence chess is played by sending moves; if the other player's move is the current percept, a reflex agent could not keep track of the board state and would have to respond to, say, "a4" in the same way regardless of the position in which it was played.

(c) There exists a task environment in which every agent is rational.

**Solution:** True. For example, in an environment with a single state, such that all actions have the same reward, it doesn't matter which action is taken. More generally, any environment that is reward-invariant under permutation of the actions will satisfy this property.

(d) The input to an agent program is the same as the input to the agent function.

**Solution:** False. The agent function, notionally speaking, takes as input the entire percept sequence up to that point, whereas the agent program takes the current percept only.

(e) It is possible for a given agent to be perfectly rational in two distinct task environments.

**Solution:** True. For example, we can arbitrarily modify the parts of the environment that are unreachable by any optimally acting agent as long as they stay unreachable.

(f) Every agent is rational in an unobservable environment.

**Solution:** False. Some actions are stupid—and the agent may know this if it has a model of the environment—even if one cannot perceive the environment state.

(g) A perfectly rational poker-playing agent never loses.

**Solution:** False. Unless it draws the perfect hand, the agent can always lose if an opponent has better cards. This can happen for game after game. The correct statement is that the agent's expected winnings are non-negative.

(h) Suppose an agent selects its next action uniformly at random from a set of possible actions. There exists a deterministic task environment in which this agent is rational.

**Solution:** True. Imagine an environment with only one action.

**Exercise 3 :**

You want to design an agent for playing tic-tac-toe (see e.g. http://boulter.com/ttt/).

- What is the state space the agent needs to reason with?
- How many states are there in the state space?
- Design 2 different feature-based representations of this state space.
- Design one relational representation of this state space.

---

**Solution:**

- The relevant states are all configurations of the $3 \times 3$ grid

- The 9 cells of the grid can be marked with any of the symbols "X","O","empty". This gives $3^9 = 19683$ states. However, this includes many states that can never be reached in an actual game (for example, the number of Xs and Os can differ by at most one).
- (a) Number the grid cells 1,...,9. Define the features *mark of cell 1*,...,*mark of cell 9* with values "X", "O", "empty". (b) Both players can have at most 5 moves in the game. Let X1,...,X5 and O1,...,O5 represent the position that the X-player (respectively the O-player) places his mark in move 1,...,5. The possible values for each feature are the grid cells 1,...,9, and "not played".
- One can use a binary relation state_of. Then state_of(cell 3,X), for example is the boolean feature that says whether cell 3 is marked with an X.

**Exercise 4 :**

You want to design a soccer playing robot (see e.g. http://www.robocup.org).

- Compared to the tic-tac-toe problem, is there a single "right" state space?
- Design one possible feature-based hierarchical state representation for the robot.

---

**Solution:** The soccer playing robot operates in a much more complicated environment than a tic-tac-toe playing agent. The current situation in a soccer match can not be fully represented by a small, fixed number of attributes, as in the tic-tac-toe problem. In principle, there are infinitely many possible positions the robots can have on the playing field, and ideally the soccer playing robot can distinguish them all, and adapt his actions to the exact scenario. Even though the robot may need to have a rather detailed state space representation to enable it to perform certain actions, it is useful to have a more abstract top-level representation on the basis of which only high-level goals are formulated. For example, we could design the robots so that they can operate in two modes: defend and attack. Then, at the highest level the robot only needs to decide in which mode it needs to play. For that decision, only some high-level features of the environment are required. Examples for features at the top-level could be:

$$\begin{aligned} &\textit{ball\_possession}\ \{\textit{own team,opponent}\} \\ &\textit{ball\_location}\{ownhalf, opponentshalf\} \\ &\textit{current\_score}\{leading, equal, trailing\} \end{aligned}$$

Once the robot has made the decision whether to play attack or defend mode, it must plan its further actions (for example *attack_via_left_field* or *attack_via_right_field*) based on lower-level features. For example, now a more precise feature for the ball location would be needed, as well as more information on the current position of all robots on the playing field. For this, the playing field might be divided

into n regions, and the current situation would be more precisely represented by:

*ball_location* {*region_1,...,region_n*}
*teammate_1* {*region_1,...,region_n*}
...
*teammate_5* {*region_1,...,region_n*}
*opponent_1* {*region_1,...,region_n*}
...
*opponent_5* {*region_1,...,region_n*}
*ball_in_possession_of* {*teammate_1,..., teammate_5, opponent_1,..., opponent_5*}

Finally, at the lowest level of the hierarchy, the robot will need to plan and execute concrete movements. For example, when it has decided to try a shot at the goal, it needs still more precise information, as expressed by features like

*direction_to_goal* {*1,2,...,360*}
*distance_to_goal* {*1,2,...,10*}
*own orientation relative to goal* {*1,2,...,360*}

**Exercise 5 (Optional):**

Discuss the differences between the problem domains from Exercises 3 and 4 according to the dimensions of complexity summarized in Section 1.5.10 (see Table 1).

| Dimension | Values |
|---|---|
| Modularity | flat, modular, hierarchical |
| Representation scheme | states, features, relations |
| Planning horizon | non-planning, finite stage, indefinite stage, infinite stage |
| Sensing uncertainty | fully observable, partially observable |
| Effect uncertainty | deterministic, stochastic |
| Preference | goals, complex preferences |
| Learning | knowledge is given, knowledge is learned |
| Number of agents | single agent, multiple agents |
| Computational limits | perfect rationality, bounded rationality |

Table 1: Dimensions of complexity