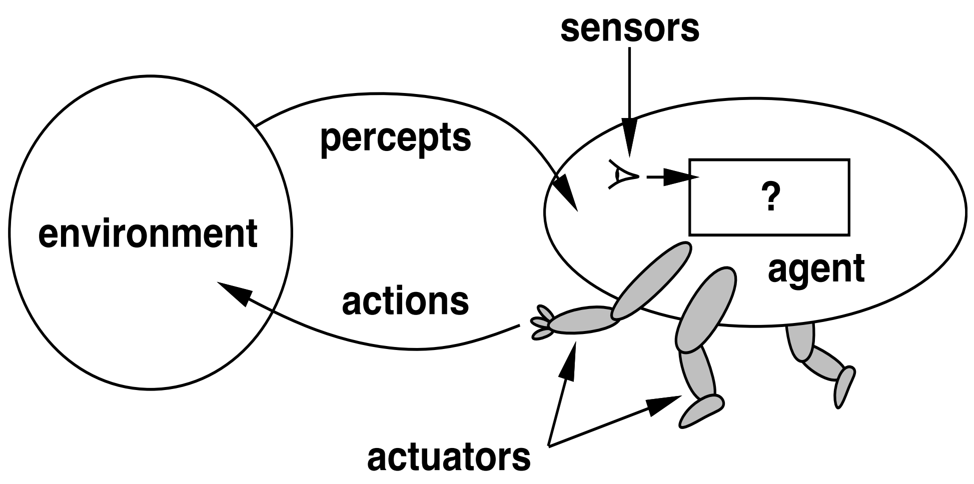
Intelligent Agents

Simon Dixon
School of Electronic Engineering and Computer Science
Queen Mary University of London

What is an agent?

- An *agent* is anything that *perceives* its *environment* through *sensors* and *acts* upon that environment through *actuators*
- A goal of AI is to design agents that can do a good job of acting on their environment, i.e., to build rational agents
- Examples
  - robots
  - softbots
  - people
  - thermostats

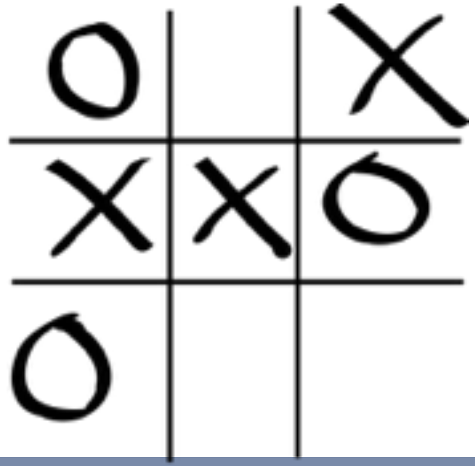


Rational Agents

- A *rational agent* is one that does the *right thing in context*
  - but what does this really mean?
- The right action should cause the agent to be most successful
  - Success is evaluated with respect to an objective *performance measure*, which depends on what the agent is designed to achieve
- But agents don't know everything about the world
  - What is rational at any time depends on the agent's background knowledge, percepts, and available actions
  - The best action may sometimes be to gather more information
- For each possible percept sequence, an ideal rational agent should do whatever is expected to maximise its performance measure, on the basis of the evidence provided by the percepts and built-in knowledge
  - Given this definition, what problems do you see (i.e. what can go wrong) with an ideal rational agent?

From Percept Sequence to Action

- An agent's behaviour depends only on its percept sequence to date, interpreted in the context of its background knowledge of the world
- So we could define behaviour as a simple mapping from every possible percept sequence to an action
  - but for any realistic agent that will probably be an infinite list
  - and the agent would only be able to do what it was explicitly told
- E.g. a tic-tac-toe agent has to choose a move, given a board state
  - how large would a complete lookup table be?
    - to store a move for all possible states
  - how else could the mapping to actions be coded?
  - how much space would that take?



Autonomy

- The agent’s behaviour is based on its own experience and its built-in knowledge
- An agent is *autonomous* to the extent that its behaviour is determined by its own experience
- Complete autonomy from the start is too difficult
  - the agent’s designer must give guidance in terms of some initial knowledge and the ability to learn and/or reason as it operates in its environment
- But agents must be able to adapt when something unexpected happens
  - otherwise the agent’s behaviour may *appear* intelligent but is actually *inflexible*

An Agent Description Schema: PEAS

Queen Mary University of London

- Designing an agent requires a good understanding of its
  - Performance Measure: evaluates how well an agent does what it is designed to do
  - Environment: the domain in which it operates and with which it interacts
  - Actuators: the means by which an agent acts upon its environment
  - Sensors: the means by which an agent perceives its environment

Agent	Performance measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable, maximise profit	Roads, other road users, customers	Steering, accelerator, brake, signals, horn	Cameras, sonar, GPS, speedometer, fuel gauge, odometer, etc.
Doctor	Healthy patient, minimise costs, avoid lawsuits	Patient, hospital, hospital staff	Ask questions, apply tests, apply treatments, refer patients	Ears, eyes, test equipment

Exercise: PEAS Analysis

Agent	Performance measure	Environment	Actuators	Sensors
Internet shopping agent				
Interactive English tutor				
Chess player				

Structure of an Intelligent Agent

AGENT = ARCHITECTURE + PROGRAM

- The agent *program* is a function that implements the agent mapping
- The program runs on a computing device, the *architecture*, which
  - makes percepts from the *sensors* available to the program
  - runs the program
  - feeds the program's actions to the *actuators*
- Since this is not a Robotics class, we focus on agent programs in this module, and not on sensing or physical action

Agent Programs

- All agent programs have the same skeleton structure
  - Sense the environment
  - Generate an action
  - Store the percept and action in memory (if necessary and feasible)
  - Repeat
- The goal and performance measure are not part of the skeleton
  - Agent may not need to know explicitly how it is being judged
  - Performance measure may be contained in its Choose-Best-Action function

Agent Programs

```
function Skeleton-Agent( percept ) returns action
    static: memory, the agent's memory of the world

    memory ← Update-Memory( memory, percept )
    action ← Choose-Best-Action( memory )
    memory ← Update-Memory( memory, action )
    return action
```

Why not just look up the answers?

- Keep the entire sequence in memory
  - use it as an index into a table
  - simply look up actions

```
function Skeleton-Agent( percept ) returns action
    static: memory, the agent's memory of the world

    memory ← Update-Memory( memory, percept )
    action ← Choose-Best-Action( memory )
    memory ← Update-Memory( memory, action )
    return action
```

Why not just look up the answers?

- Keep the entire sequence in memory
  - use it as an index into a table
  - simply look up actions

```
function Lookup-Agent( percept ) returns action
    static: percepts, a sequence, initially empty
            table, full, indexed by percept sequence
    percepts ← Append( percepts, percept )
    action ← Lookup( percepts, table )
    return action
```

Why not just look up the answers?

Queen Mary
University of London

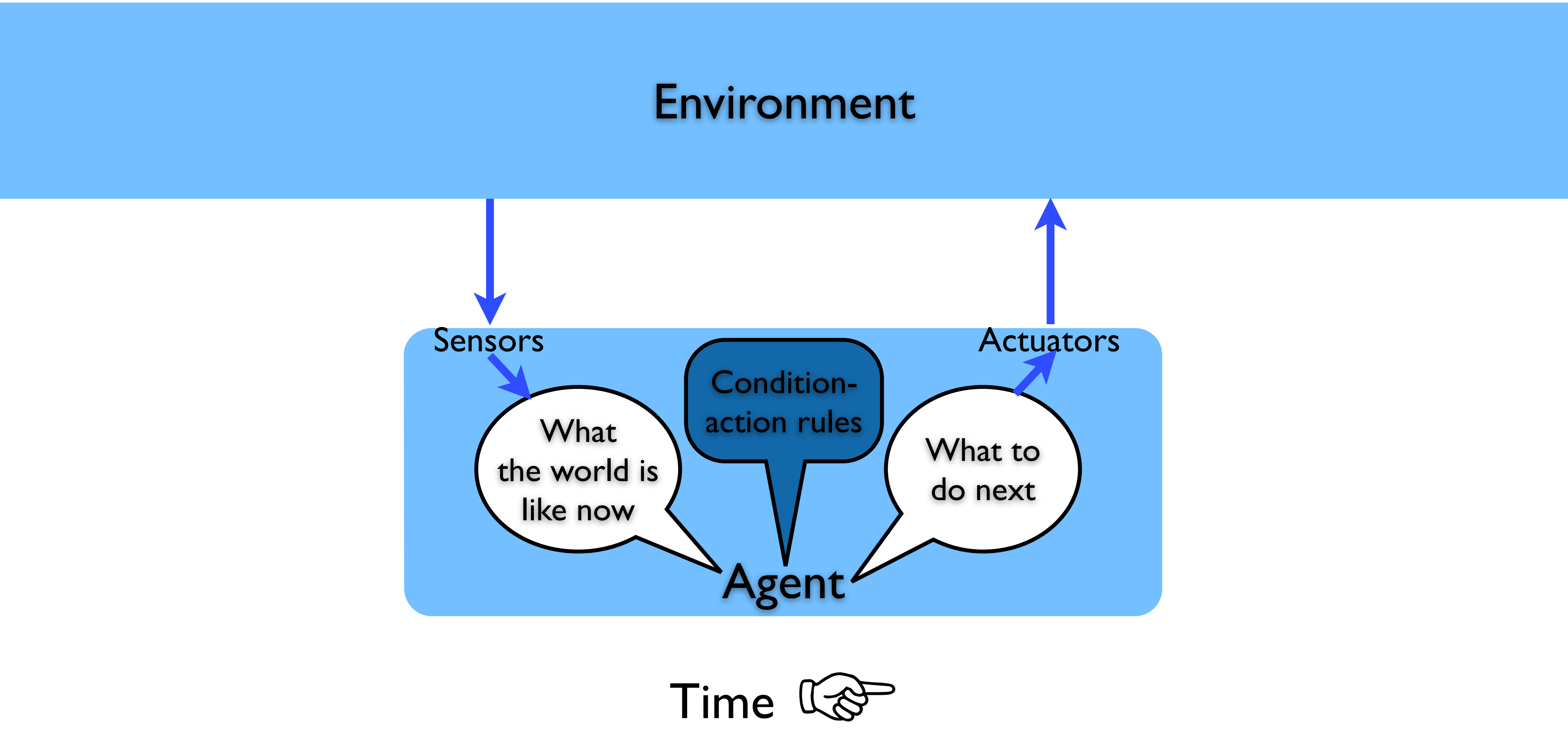
- This approach will fail because
  - The look-up table for will be too large (for chess, ~35^{100} entries)
  - Designer has to build whole of table in advance
  - No autonomy, so no flexibility
  - Even a learning agent would take forever to learn a real-world table

```
function Lookup-Agent( percept ) returns action
    static: percepts, a sequence, initially empty
            table, full, indexed by percept sequence
    percepts ← Append( percepts, percept )
    action ← Lookup( percepts, table )
    return action
```

Different Types of Agent

- Different types of agent are useful in different circumstances
    - it is generally best to use the simplest possible in context
- Types of agent
    - simple reflex
    - agents with state
    - goal-based agents
    - utility-based agents

Simple Reflex Agents



Simple Reflex Agents

```
**function** Simple-Reflex-Agent( *percept* ) **returns** action
    **static**: *rules*, a set of productions

    state ← Interpret-Input( percept )
    rule ← Rule-Match( state, rules )
    action ← Rule-Action( rule )
    return action
```

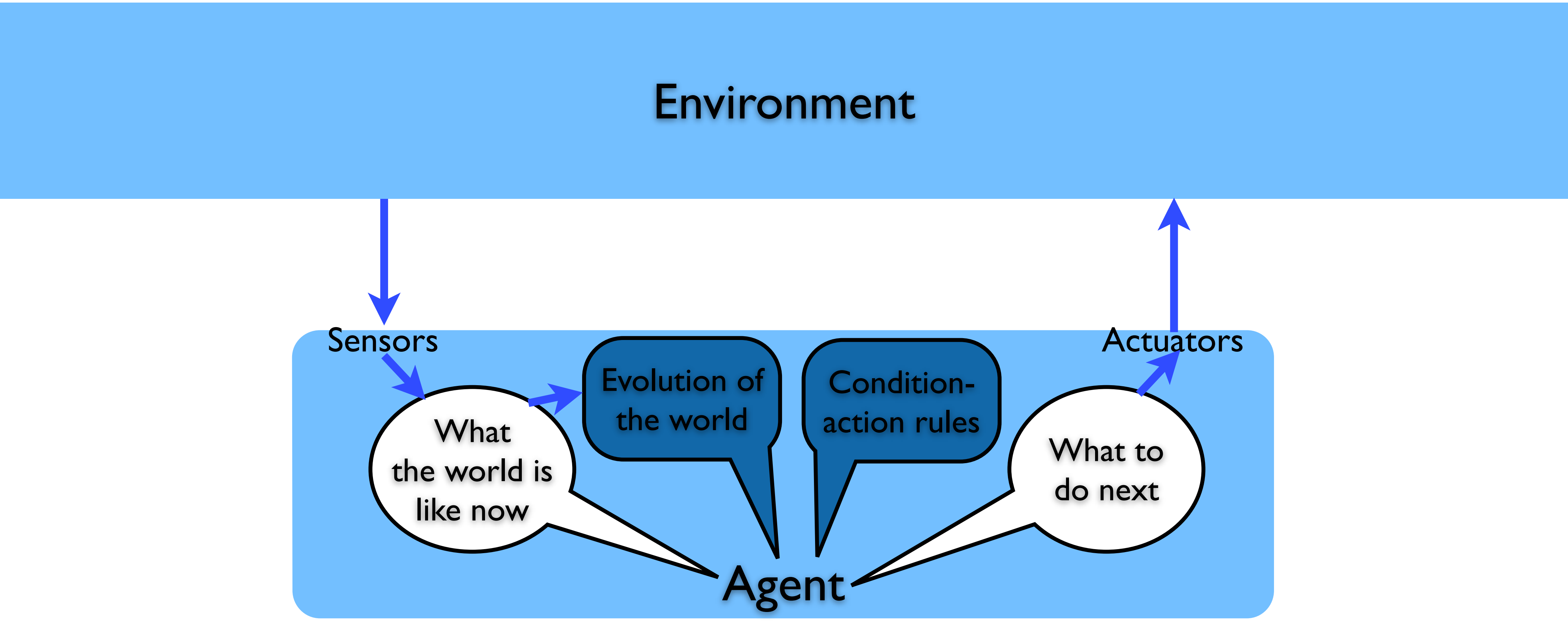
Simple Reflex Agents

- Some drivers' actions are more or less automatic
  - e.g., braking if car in front is braking
- Such reflex rules are sometimes called
  - condition-action rules
  - production rules
  - if-then rules
- Some processing needed to test conditions
- Efficient, but narrow

Reflex Agents with State

- Sometimes an agent needs more than just its current percept and must maintain an internal state
  - This helps to distinguish between world states that generate the same perceptual input but nonetheless are significantly different in terms of the actions it must take
- Example
  - An agent that is looking for its car keys needs to remember which drawers it has already looked in, otherwise it might get stuck in a loop: open drawer, look for keys, close drawer

Reflex Agents with State



Time ☞

Reflex Agents with State

function
static
state
rule
action
return

```
function Reflex-Agent-State( percept ) returns action
    static: rules, a set of productions
            state, a description of the current world
    state ← Update-State( state, percept )
    rule ← Rule-Match( state, rules )
    action ← Rule-Action( rule )
    state ← Update-State( state, action )
    return action
```

Reflex Agents with State

- The Update-State function must
  - interpret new percepts in light of existing knowledge about state
  - use information about how the world evolves to keep track of unseen parts
    - e.g. tracking a moving object which is sometimes occluded
  - know about what the agent’s actions do to the state of the world
    - particularly if the effects are not directly perceivable
  - combine all these into a new internal state description

Goal-based Agents

- Knowing the state of the environment is not always enough
  - The correct action may depend on the *goals* of the agent
  - The same agent may have different goals at different times
- An agent program combines goal information with information about the results of possible actions to choose actions that achieve the goal
  - sometimes simple
    - when the goal is satisfied by a single action
  - sometimes complex
    - when the goal requires a sequence of actions
  - Search and planning are sub-fields of AI devoted to finding action sequences that achieve the agent's goals

Goal-based Agents

- Decision making is more complex with goals
  - it requires consideration of the future
    - What will happen if I do X?
    - Does X bring me closer to achieving my goal?
- Although they may be less efficient, goal-based agents are more flexible
  - Taxi-driver agent may have its destination as a goal causing it to behave in different ways for each possible percept sequence
- Goals alone may not be sufficient to generate high-quality behaviour
  - many action sequences may achieve the goal, but some are better than others

Utility-based Agents

- Goal satisfaction provides a crude distinction between good states and bad states, whereas a more general performance measure would allow a comparison between states, or sequences of states, according to exactly how desirable they are
- This measure of preference over world states is known as *utility*
- Utility (usually) maps states onto real numbers so they are comparable
- Utility is needed when
  - there are conflicting goals, e.g., speed versus safety
  - there are several means of reaching a goal
  - several goals exist but cannot be reached with certainty
    - utility provides a way in which the likelihood of success can be weighed against the importance of the goals

Environments

- Environments have different properties, which determine what kinds of agents can work in them

Fully Observable	Partially Observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Single Agent	Multiple Agent

Classifying Environments

- Fully observable (*vs* partially observable)
  - sensors give complete state of the environment
  - agent needn't keep track of the world state
  - *effectively observable* means that sensors can detect all *relevant* aspects
  - most AI environments are only partially observable
- Deterministic (*vs* stochastic)
  - next state completely determined by current state and agent action
  - in stochastic environments next state is uncertain (though a probabilistic model may be available)
  - if environment is deterministic except for actions of other agents, it is called *strategic*

Classifying Environments

- Episodic (vs sequential)
  - agent's experiences divided into episodes: agent perceives then acts
  - quality of action depends only on the episode itself
  - subsequent episodes are independent of previous ones
    - e.g., one game of chess in a tournament vs. each move in one game
  - agents do not have to think ahead
- Dynamic (vs static)
  - environment may change while agent is deliberating
  - *semi-dynamic*: agent's *utility* scores changes with passage of time though the environment is static
  - in a static environment an agent does not need to look at the world while deciding on an action, nor does it need to worry about the passage of time

Classifying Environments

- Discrete (vs continuous)
  - limited number of distinct, clearly defined percepts and actions
    - playing chess is discrete: there is a limited number of moves
    - driving a taxi is continuous: speed and location vary infinitessimally
- Multi-agent (vs single-agent)
  - competitive: maximising agent A's performance measure implies minimising agent B's
  - cooperative: performance measures of both agents A and B are maximised by the same action(s)
- The real world is
  - partially observable, stochastic, sequential, dynamic, continuous and multi-agent

Exercise: Environment Types

Task	Observable?	Deter-ministic?	Episodic?	Static?	Discrete?	Multi-agent?
Chess player (for a timed game)						
Poker player						
Taxi driver						

Past **Exam**ple Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer A: Performance measure is a function which produce a numeric value describing how successful (close to goal) an agent's action is.

Environment decides the nature of the agent's task. It can be fully or partially observable, deterministic or stochastic, discrete or continuous, episodic or sequential, adversarial or not, dynamic or static.

Actuators are possible actions the agent can do.

Sensors are the way the agent perceives the world.

Past **Exam**ple Question

Queen Mary
University of London

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer A (continued): chess agent: Performance measure = win gets 1 point, lose gets -1 point, draw gets 0 point.

Environment: fully observable, adversarial, static, discrete, sequential, deterministic.

Actuators: in the agent's turn, move one of the agent's pieces according to chess rules.

Sensors: the make-up of the board.

Past **Exam**ple Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer B: Performance measure: goals that should be achieved by the agent and ways to measure them.

Environment: context in which the agent is inserted.

Actuators: medium of interaction between the agent and its environment.

Sensors: mechanisms through which the agent perceives its environment.

Past **Exam**ple Question

- Q1b (2016): *The PEAS description (Performance measure, Environment, Actuators, Sensors) can be used to describe the task environment of an agent. Define each of the four terms and illustrate them with reference to a chess-playing agent.* [8 marks]

Answer B (continued): Chess-playing agent

P.M.: Number of opponent's pieces captured, number of pieces in the game, check-mate, value of pieces in the game, check attempts

E: online game platform, computer, board, pieces

A: moving piece of interface, writing new position on screen, interface.

S: time, opponent's movement, board knowledge