Queen Mary
University of London

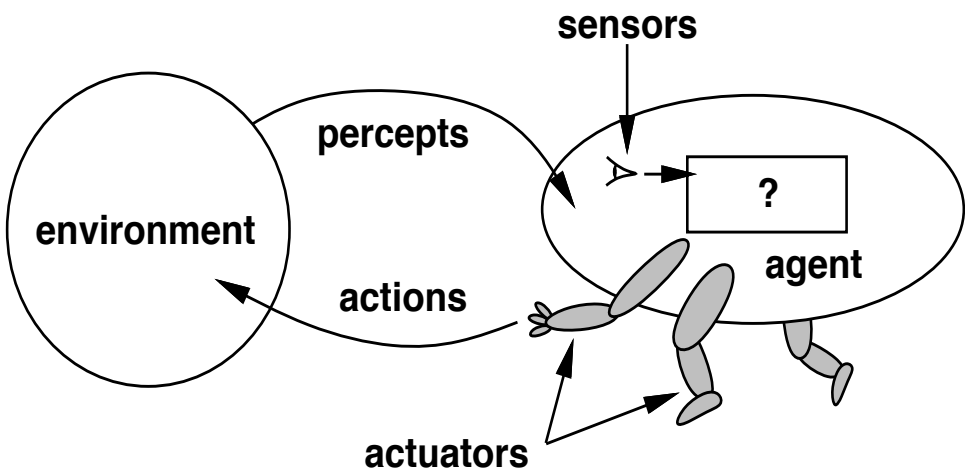
Intelligent Agents

Geraint A. Wiggins
Professor of Computational Creativity
Department of Computer Science
Vrije Universiteit Brussel

What is an agent?

Queen Mary University of London

- An *agent* is anything that *perceives* its *environment* through *sensors* and *acts* upon that environment through *actuators*
- This module aims to introduce the design of agents that can do a good job of acting on their environment, i.e., to build rational agents
- For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance
- Examples
  - robots, softbots, people, thermostats



Rational Agents

Queen Mary University of London

- 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
  - but how and when should the agent's success be evaluated?
- Success needs to be evaluated with respect to an objective *performance measure*, which depends on what the agent is designed to achieve
- *When* to look at performance is also important and depends on what the agent has been designed to achieve

Agent	**Performance Measure**	**When to measure performance**
Vacuum cleaner	**?**	**?**
Call Router	**?**	**?**

Rational Agents: performance measures

Queen Mary University of London

- Performance measure must be realistic
  - An omniscient agent knows the actual outcome of its actions and can act accordingly; but this is impossible in reality.
  - Example
    - *An agent walking along the Boulevard Franklin Roosevelt/Franklin Rooseveltlaan sees an old friend on the other side of the road. It looks both ways, sees that no traffic is coming and starts to cross the road. Meanwhile, at 33,000 feet, a cargo door falls off a plane passing overhead which crashes to the ground flattening the agent. Was the agent behaving rationally?*
  - Need to consider expected success given what has been perceived.
- What is rational at any given time depends on
  - the performance measure defining degree of success
  - the agent's percept sequence
  - what the agent knows about the environment
  - the actions available to the agent

An Ideal Rational Agent

Queen Mary University of London

- 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 percept sequence and whatever built-in knowledge the agent has
- NB: The best action may sometimes be to gather more information

From Percept Seqence to Action

Queen Mary University of London

- Agent's behaviour depends only on its percept sequence to date, interpreted in context of its rules of the world
- So we 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 anyway, the agent would only be able to do what it was explicitly told
- Specifying the best action an agent ought to take in response to any given percept sequence is a good design for an ideal agent

From Percept Sequence to Action

Queen Mary University of London

- Example:
  - A square root agent
    - given a percept, x, display the positive number, z, such that $z^2 = x$
  - Ideal mapping could be a lookup table or a function:

x	z
1.000	1.0000000000000
1.100	1.0488088481702
1.200	1.0954451150103
1.300	1.1401754250991
1.400	1.1832159566199
1.500	1.2247448713916
1.600	1.2649110640674
1.700	1.3038404810405
1.800	1.3416407864999
1.900	1.3784048752090

```
FUNCTION SQRT(X)
  Z ← 1.0 /* INITIAL GUESS */
  REPEAT UNTIL |Z² − X| < 10⁻¹⁵
    Z ← Z − (Z²−X)/(2Z)
  END
  RETURN Z
```

Autonomy

Queen Mary University of London

- 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
  - Without the ability to learn or adapt, the agent's behaviour may *appear* intelligent but is actually *inflexible*

When an Agent is not Autonomous

Queen Mary University of London

- Example
  - After a dung beetle digs its nest and lays its eggs, it fetches a ball of dung from a nearby heap to plug the entrance

When an Agent is not Autonomous

Queen Mary University of London

- Example
  - After a dung beetle digs its nest and lays its eggs, it fetches a ball of dung from a nearby heap to plug the entrance
  - but if the ball of dung is removed en route, the beetle carries on and mimes plugging the nest with the nonexistent dung ball, never noticing that it is missing.

When an Agent is not Autonomous

Queen Mary University of London

- Example
  - After a dung beetle digs its nest and lays its eggs, it fetches a ball of dung from a nearby heap to plug the entrance
  - but if the ball of dung is removed en route, the beetle carries on and mimes plugging the nest with the nonexistent dung ball, never noticing that it is missing.
- A truly autonomous intelligent agent should be able to operate successfully in a wide variety of environments, given sufficient time to adapt.

Structure of an Intelligent Agent

Queen Mary University of London

- What's inside an agent?
- 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

AGENT = ARCHITECTURE + PROGRAM

An Agent Description Schema: PEAS

Queen Mary University of London

- Designing an agent program requires a good understanding of its
  - **P**erformance Measure(s)
  - **E**nvironment(s)
  - **A**ctuators
  - **S**ensors

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 guage, 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

Queen Mary University of London

Agent	Performance measure	Environment	Actuators	Sensors
Oil refinery controller				
Interactive English tutor				
Call-routing agent				
Internet shopping agent				

Agent Programs

Queen Mary University of London

- All agent programs have the same skeleton structure
  - Take just one percept at a time
  - Generate an action
  - Agent must build up its percept sequence in memory (if necessary and feasible)
- 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

Queen Mary University of London

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?

Queen Mary University of London

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?

Queen Mary University of London

- 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?

Queen Mary University of London

- 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*

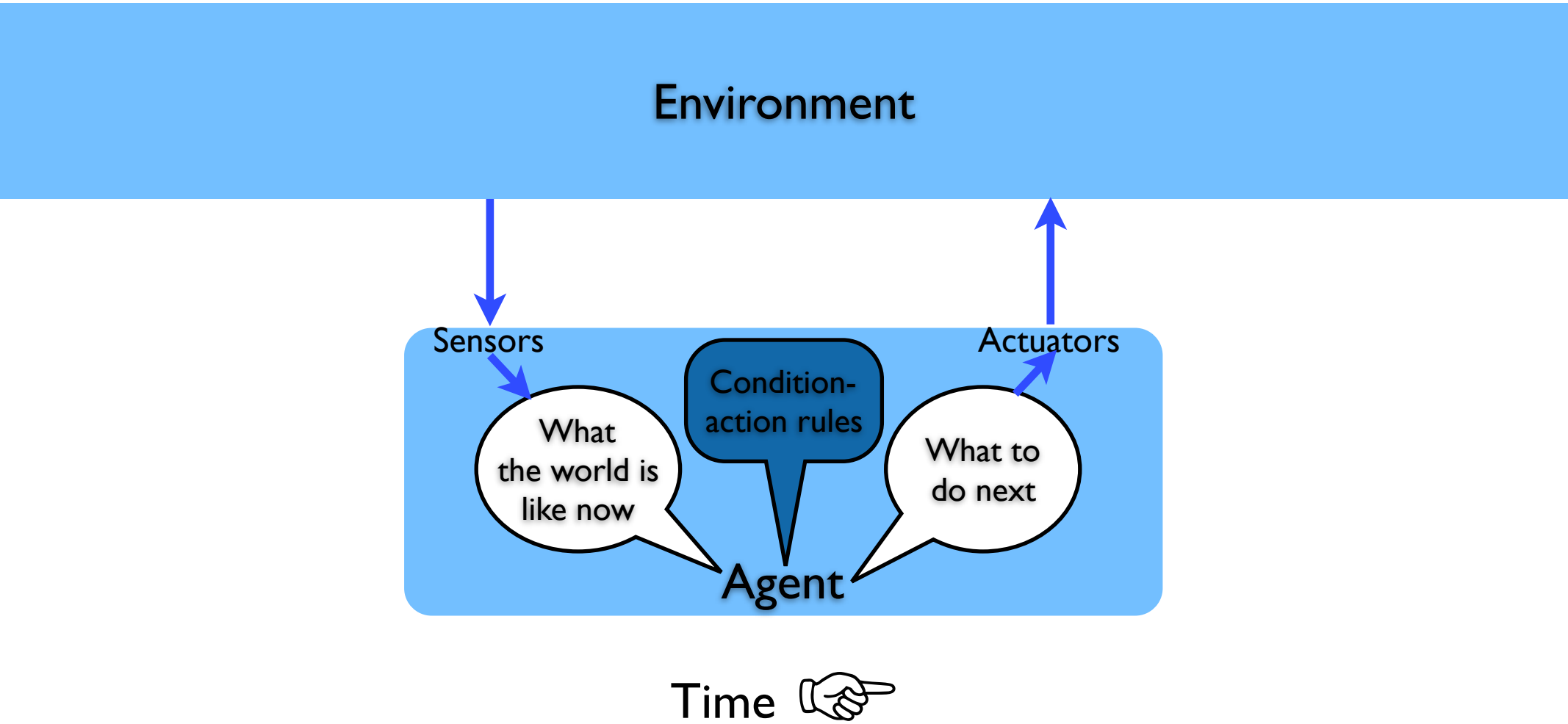
Different types of agent

Queen Mary University of London

- Different types of agent are useful in different circumstances
  - it's useful to know which because it's 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

Queen Mary University of London



Simple Reflex Agents

Queen Mary University of London

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

Simple Reflex Agents

Queen Mary University of London

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*

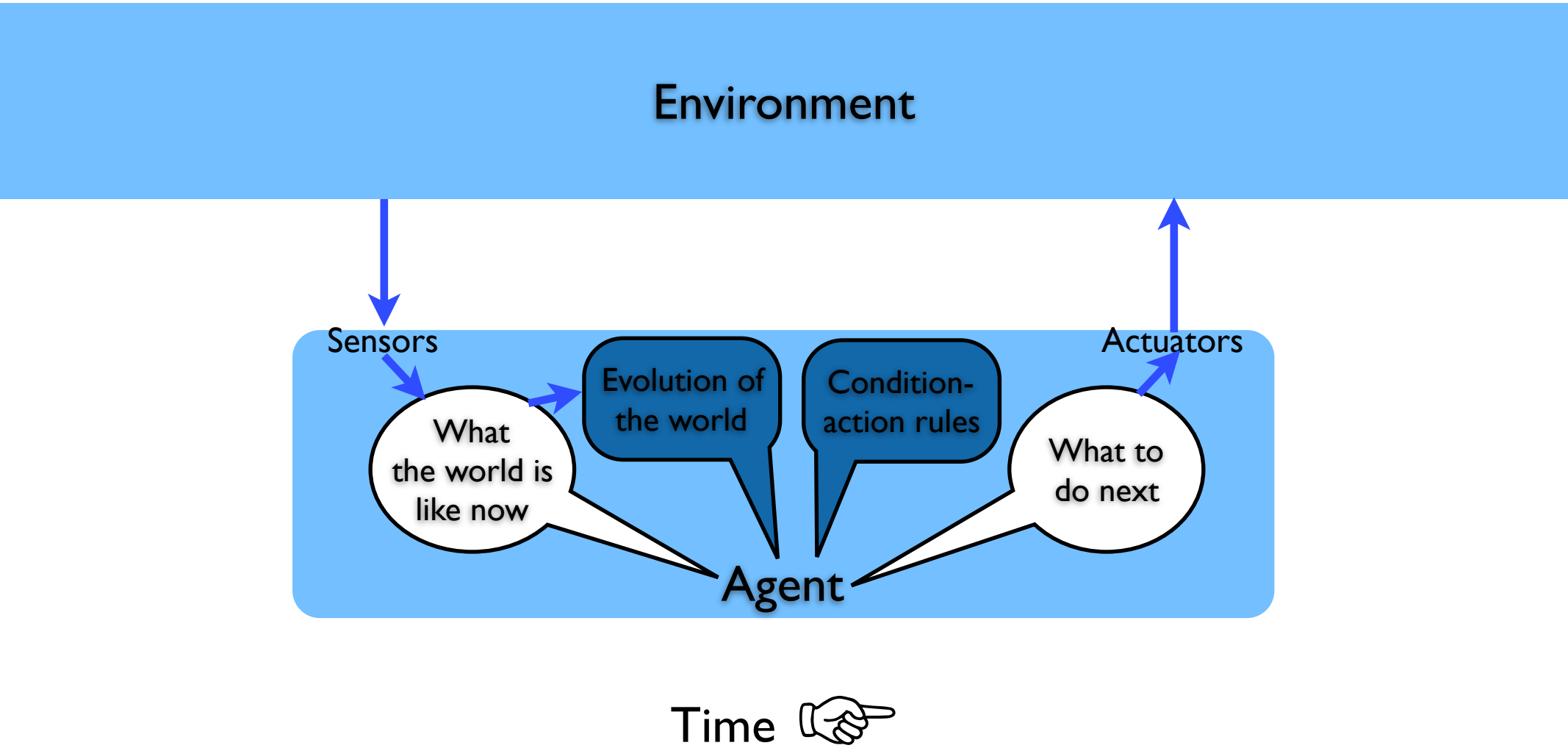
Reflex Agents with State

Queen Mary University of London

- 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
  - A taxi-driver agent wants to overtake but cannot tell if there are cars nearby
  - By recording its last look in the rear-view mirror in its internal state, it can judge whether it is safe to pull out.

Simple Reflex Agents

Queen Mary University of London



Simple Reflex Agents

Queen Mary University of London

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*

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

Queen Mary University of London

- Maintaining and updating an internal state requires
  - information about how the external world evolves
    - e.g., a car that was overtaking is likely to be closer than it was a moment ago
  - information about how its own actions affect the world
    - e.g., when the agent overtakes, it leaves a gap in the lane it has left
- The Update-State function must
  - create a new internal state description
  - interpret new percept in light of existing knowledge about state
  - use information about how the world evolves to keep track of unseen parts
  - know about what the agent's actions do to the state of the world

Goal-based Agents

Queen Mary University of London

- Knowing the state of the environment is not always enough
  - The correct action may depend on the *goals* of the agent
- An agent program combines goal information with information about the results of possible actions to choose actions that achieve the goal
  - sometimes simple
    - when goal is satisfied by a single action
  - sometimes complex
    - when 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

Queen Mary University of London

- Decision making is more complex with goals
  - it requires consideration of the future
    - What will happen if I do X?
    - Will that make me happy?
- 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

Queen Mary University of London

- 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 state 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

Queen Mary University of London

- 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

Queen Mary University of London

- 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

Queen Mary University of London

- 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

Queen Mary University of London

- 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: maximising performance measures of both agents A and B
- The real world is
  - inaccessible, stochastic, sequential, dynamic, and continuous

Exercise: Environment Types

Queen Mary University of London

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