Knowledge Representation & Reasoning COMP9016

Dr Ruairí O'Reilly
ruairi.oreilly@cit.ie

Intelligent Agents

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Acting rationally

>> *Rational* behavior: doing the right thing

>> The *right* thing: that which is expected to maximize goal achievement, given the available information

>> This does not necessarily involve thinking—e.g.? blinking reflex

>> Thinking should be in the service of rational action Aristotle (Nicomachean Ethics): “Every art and every inquiry, and similarly every action and pursuit, is thought to aim at some good”

Rational Agents

>> An *agent* is an entity that perceives and acts.

>> This course is about designing rational agents. Abstractly, an agent is a function from percept histories to actions:

$$f : P* \rightarrow A$$

>> For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance

>> Caveat: computational limitations make perfect rationality unachievable

=> design best **program** for given machine resources

Acting rationally

>> *Rational* behavior: doing the right thing

>> The *right* thing: that which is expected to maximize goal achievement, given the available information

>> This does not necessarily involve thinking—e.g.? blinking reflex

>> Thinking should be in the service of rational action Aristotle (Nicomachean Ethics): “Every art and every inquiry, and similarly every action and pursuit, is thought to aim at some good”

Which of the following can be done at present?

- Play a decent game of table tennis
- Drive safely along a curving mountain road
- Drive safely along Telegraph Avenue
- Buy a week's worth of groceries on the web
- Buy a week's worth of groceries at Berkeley Bowl
- Play a decent game of bridge
- Discover and prove a new mathematical theorem
- Design and execute a research program in molecular biology
- Write an intentionally funny story
- Give competent legal advice in a specialized area of law
- Translate spoken English into spoken Swedish in real time
- Converse successfully with another person for an hour
- Perform a complex surgical operation

Outline of Lecture

>> Agents and environments

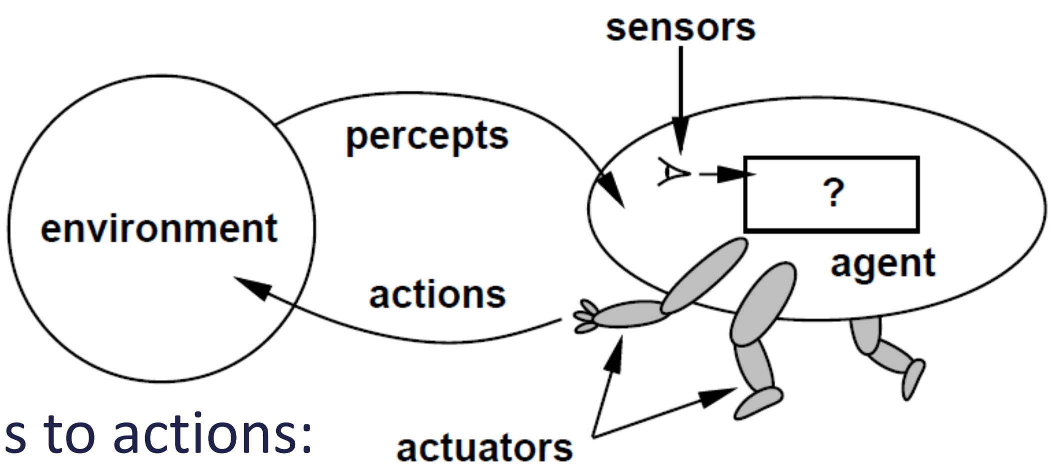
>> Rationality

>> PEAS (Performance measure, Environment, Actuators, Sensors)

>> Environment types

>> Agent types

Agents and environments



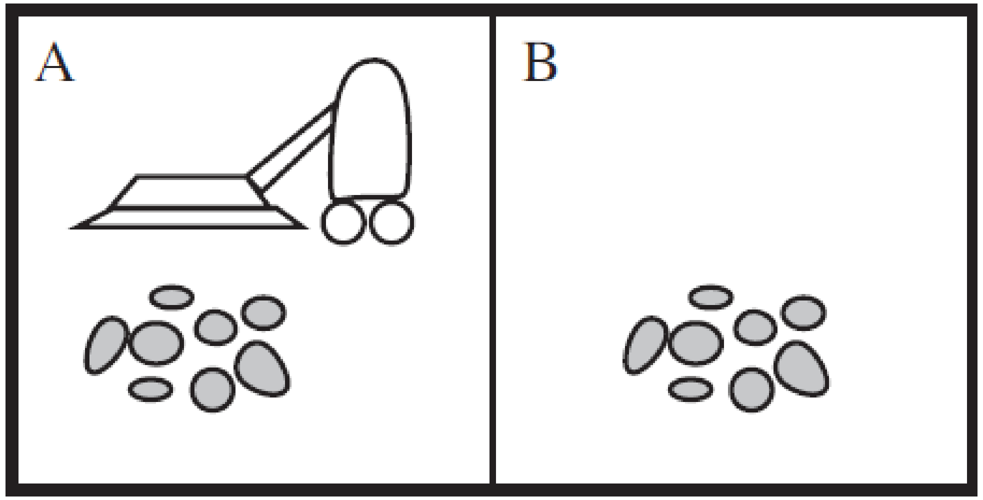
>> Agents include humans, robots, softbots, thermostats, etc.

>> The agent function maps from percept histories to actions:

$$f : P* \rightarrow A$$

>> The agent program runs on the physical architecture to produce: f

Vacuum-cleaner world



Percept sequence	Action
[A : Clean]	Right
[A : Dirty]	Suck
[B : Clean]	Left
[B : Dirty]	Suck
[A : Clean, A : Clean]	Right
[A : Clean, A : Dirty]	Suck
…..	…..

>>Tabulating the **agent function**

>> Implementing via the **agent program**

Notion of agency

A tool for analysing systems, not an absolute characterization that divides the world into agent and non-agents

Rationality

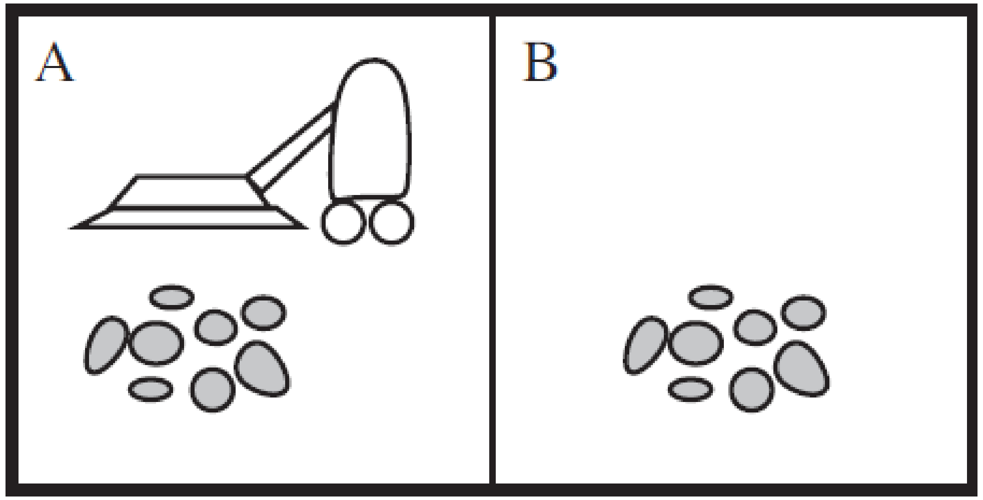
What is rational at any given time depends on four things:

- The performance measure that defines the criterion of success.
- The agent’s prior knowledge of the environment.
- The actions that the agent can perform.
- The agent’s percept sequence to date.

Definition of a **rational agent**: *“For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.”*

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Is this a rational agent



Performance measure?

Environment?

Actions?

Percept sequence?

> The agent is rational.

> What would make the agent irrational?

Omniscience, learning, and autonomy

>> Rationality is not the same as perfection. Rationality maximises expected performance, while perfection maximises actual performance

>> Information Gathering & Exploration

>> Learning - not only to gather information but also to learn as much as possible from what it perceives

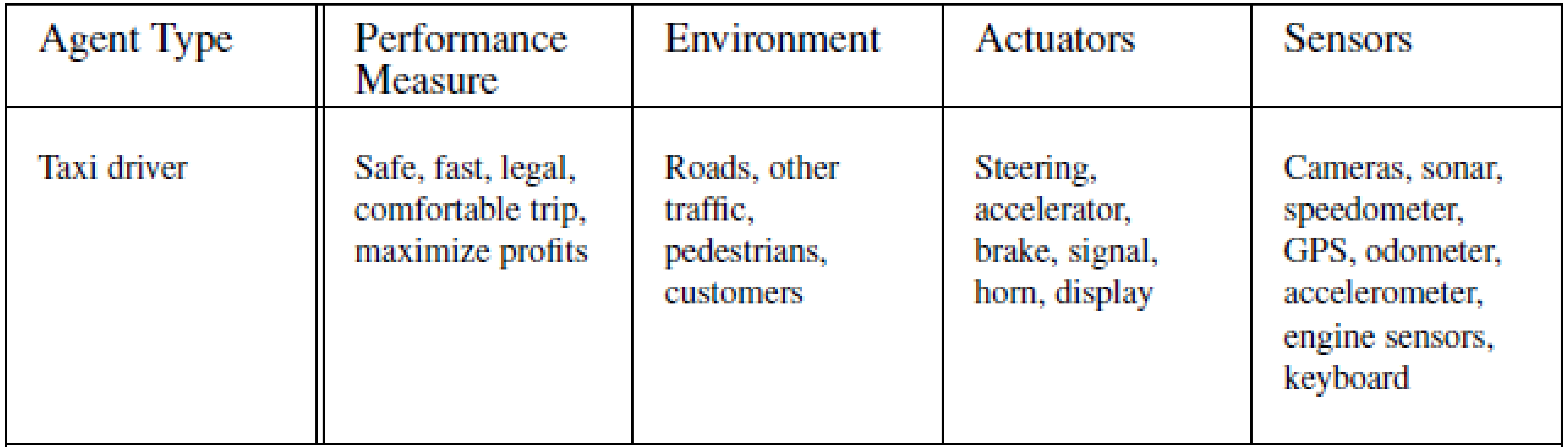
>> Autonomy - A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge. For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.

Task Environments

>> *Specifying a task environment*

>> PEAS (Performance, Environnent, Actuators, Sensors) description.

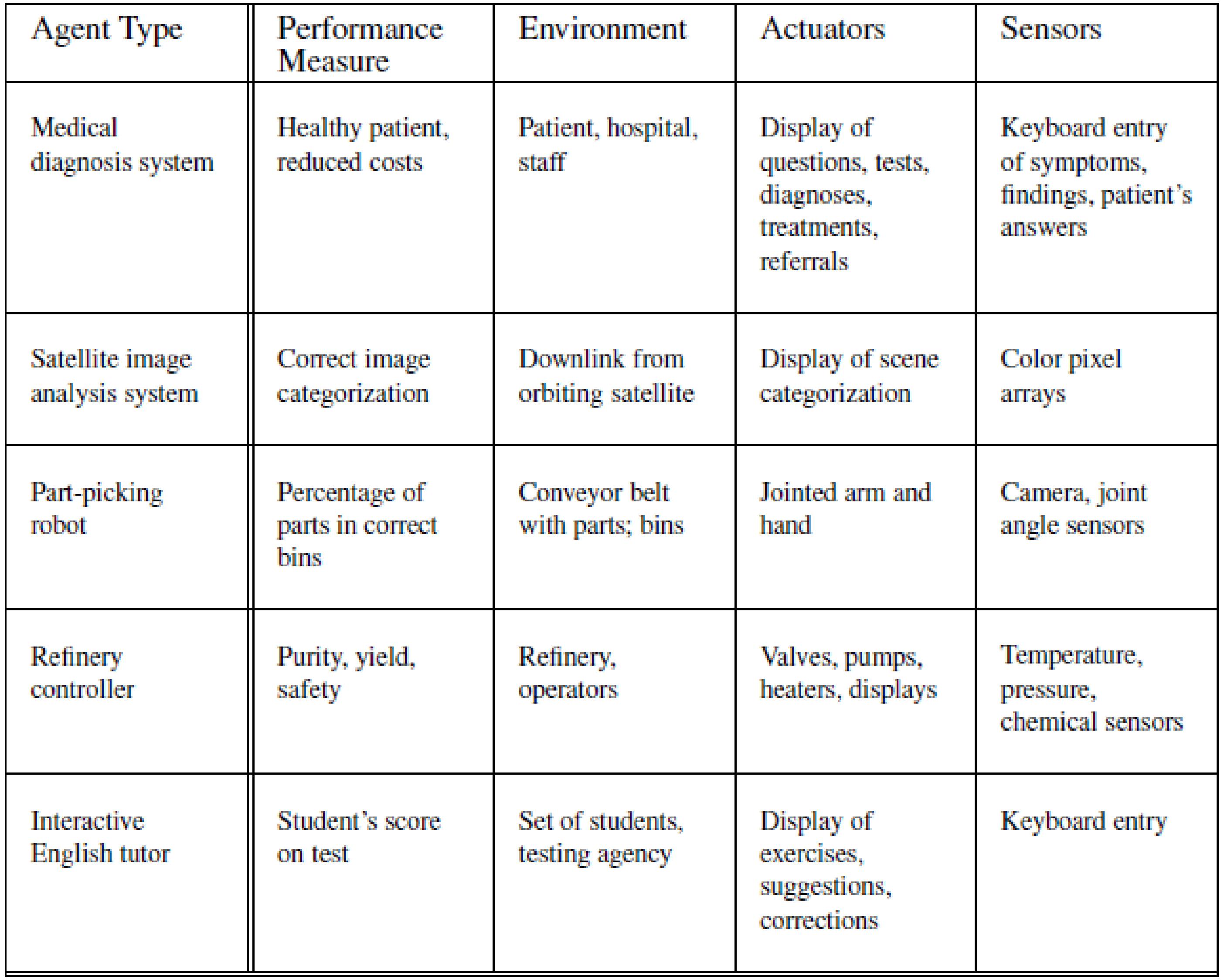
>> A more complex problem – autonomous taxi driver



Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Examples of agent types and their PEAS descriptions



Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Task Environments

>> Fully observable vs. partially observable

>> Single agent vs. multiagent

a. competitive

b. co-operative

c. communication

d. randomized behaviour

>> Deterministic vs. stochastic

a. uncertain

b. nondeterministic

>> Episodic vs. sequential

>> Static vs. dynamic

a. semidynamic

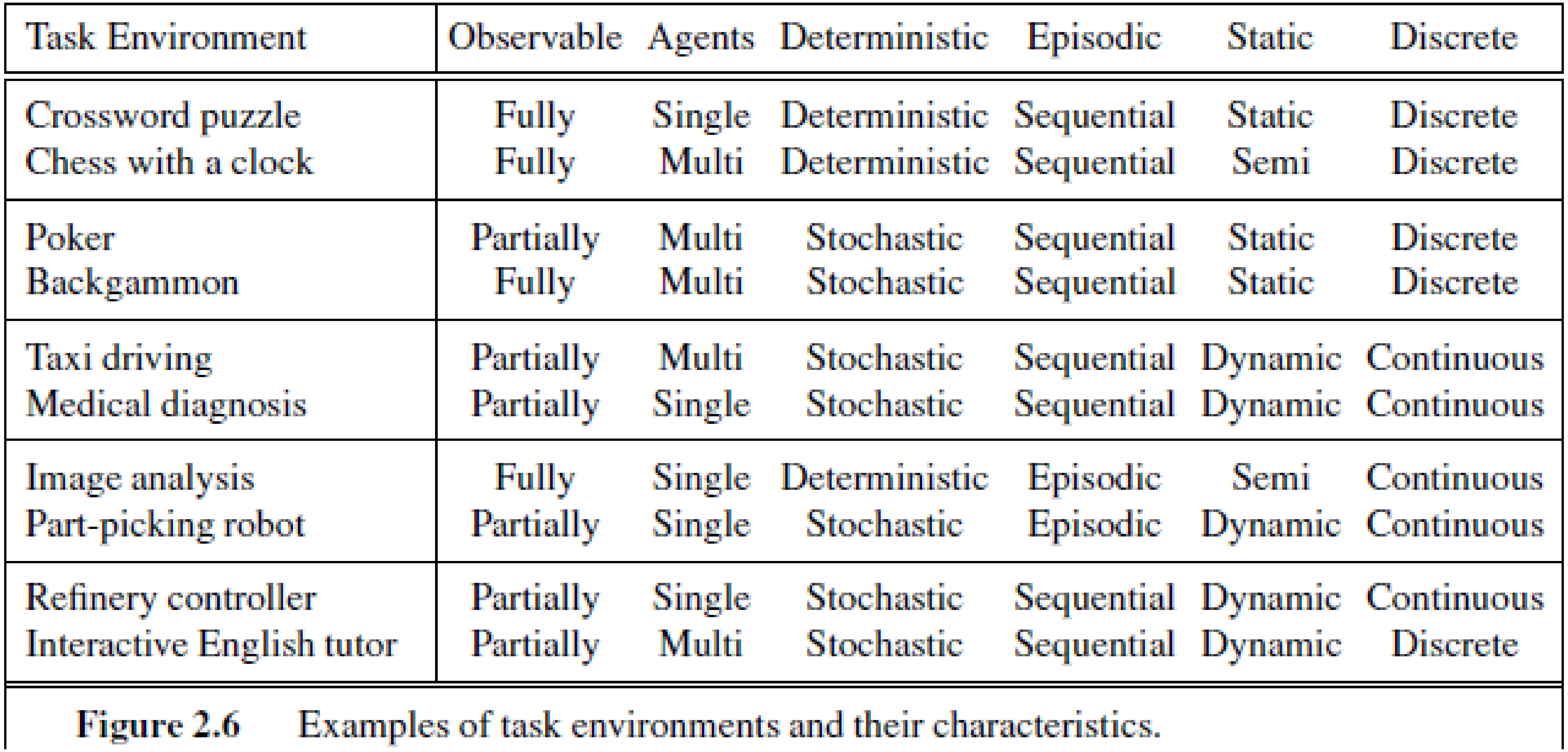
>> Discrete vs. continuous

>> The hardest case?

Partially observable, multiagent, stochastic, sequential, dynamic, continuous, and unknown

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Task Environments Examples



Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Figure 2.6 Examples of task environments and their characteristics.

>> Rationality is not the same as perfection. Rationality maximises expected performance, while perfection maximises actual performance

Structure of Agents

>> *So far we have talked about agents by describing behavior……*

>> The job of AI is to design an agent program that implements the agent function—the mapping from percepts to actions. We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the architecture:

agent = architecture + program

>> Appropriateness of program chosen for architecture

Agent programs

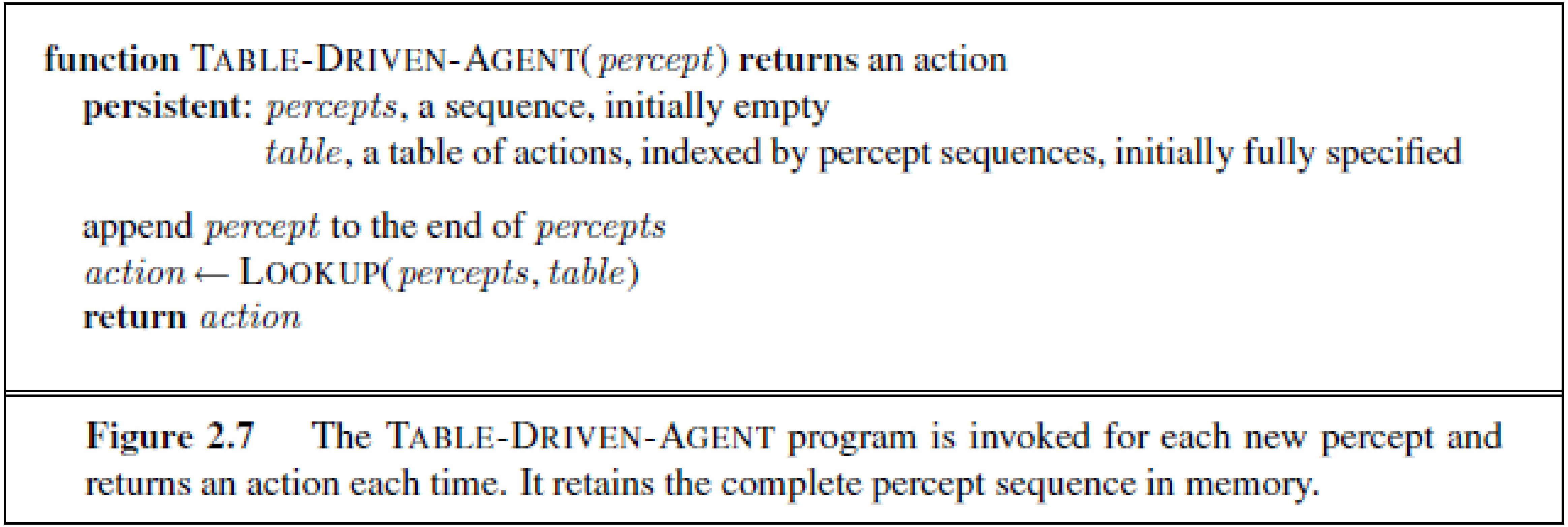
>> *Agent skeleton: current percept as input from the sensors and return an action to the actuators*

>> Differentiate between the agent program, which takes the current percept as input, and the agent function, which takes the entire percept history?

The agent program takes just the current percept as input because nothing more is available from the environment; if the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts.

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Pseudocode agent program



```
function TABLE-DRIVEN-AGENT(percept) returns an action
    persistent: percepts, a sequence, initially empty
                table, a table of actions, indexed by percept sequences, initially fully specified

    append percept to the end of percepts
    action ← LOOKUP(percepts, table)
    return action
```

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

>> A trivial agent program that keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do.

>> The table (vacuum world) represents explicitly the agent function that the agent program embodies. To build a rational agent in this way, we as designers must construct a table that contains the appropriate action for every possible percept sequence.

Why is the table-driven approach doomed to fail?

>> *P* = set of all possible percepts, *T* = lifetime of the agent

>> What will the lookup table look like? $\sum_{t=1}^{T} |P|^t$

>> *Example: Taxi*

>> Despite all this, TABLE-DRIVEN-AGENT does do what we want: it implements the desired agent function. The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table.

Basic agent programs

>> Simple reflex agents

>> Model-based reflex agents;

>> Goal-based agents; and

>> Utility-based agents.

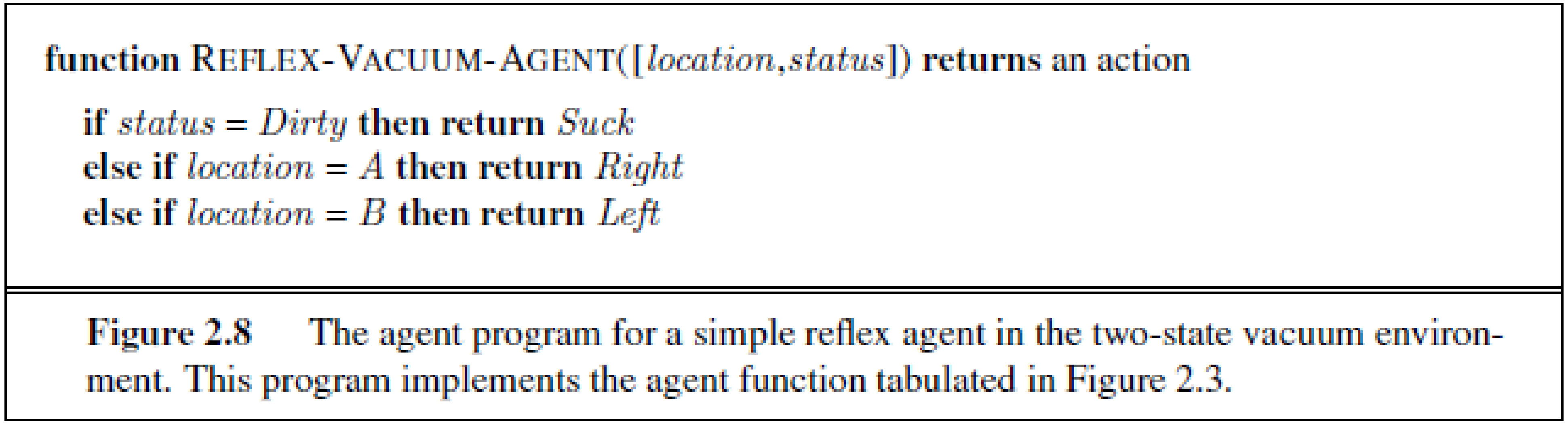
>> Each kind of agent program combines particular components in particular ways to generate actions.

Simple Reflex Agents

>> These agents select actions on the basis of the current percept, ignoring the rest of the percept history. e.g. vacumn agent

>> The basis of its decision making?

>> Agent program:



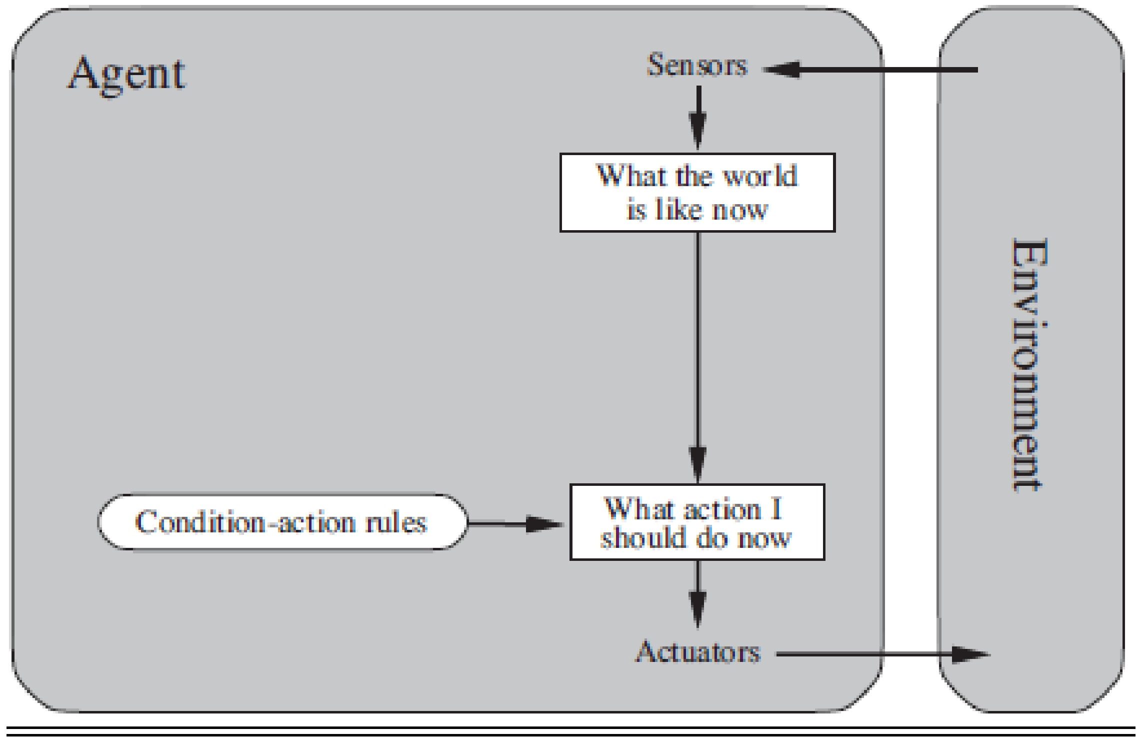
```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Simple reflex agent schema & program

>> Build a general-purpose interpreter for condition action rules

>> Create rule sets for specific task environments



Schematic diagram of a simple reflex agent.

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Simple reflex agent - reflection

>> Simple reflex agents have the admirable property of being simple, but they turn out to be of limited intelligence

>> The agent will only work only if the correct decision can be made based on only the current percept—that is, only if the environment is fully observable.

>> Workaround?

>> Randomisation: a randomized simple reflex agent might outperform a deterministic simple reflex agent.

Model-based reflex agents

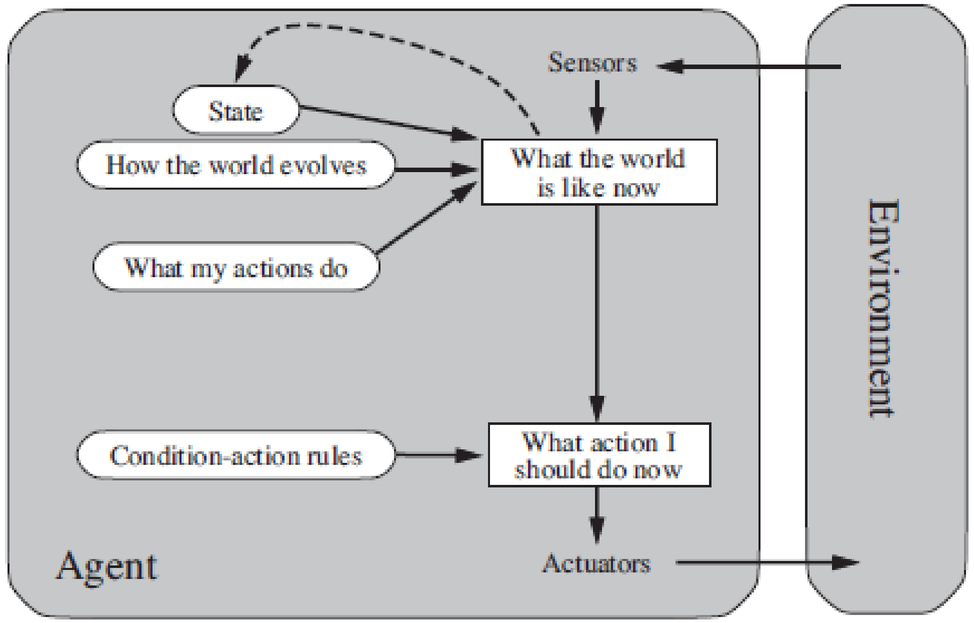
>> *The most effective way to handle partial observability is for the agent to keep track of the part of the world it can’t see now.*

>> That is, the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

>> Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program:

1) How the world evolves independent of the agent.
2) How the agent's actions affect the world.

A model-based reflex agent



```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Goal-based agents

>> *Knowing something about the current state of the environment is not always enough to decide what to do.*

>> What if we combine a “goal” with a model of the world?

>> Straightforward and not so straightforward? Examples…

Goal-based agents

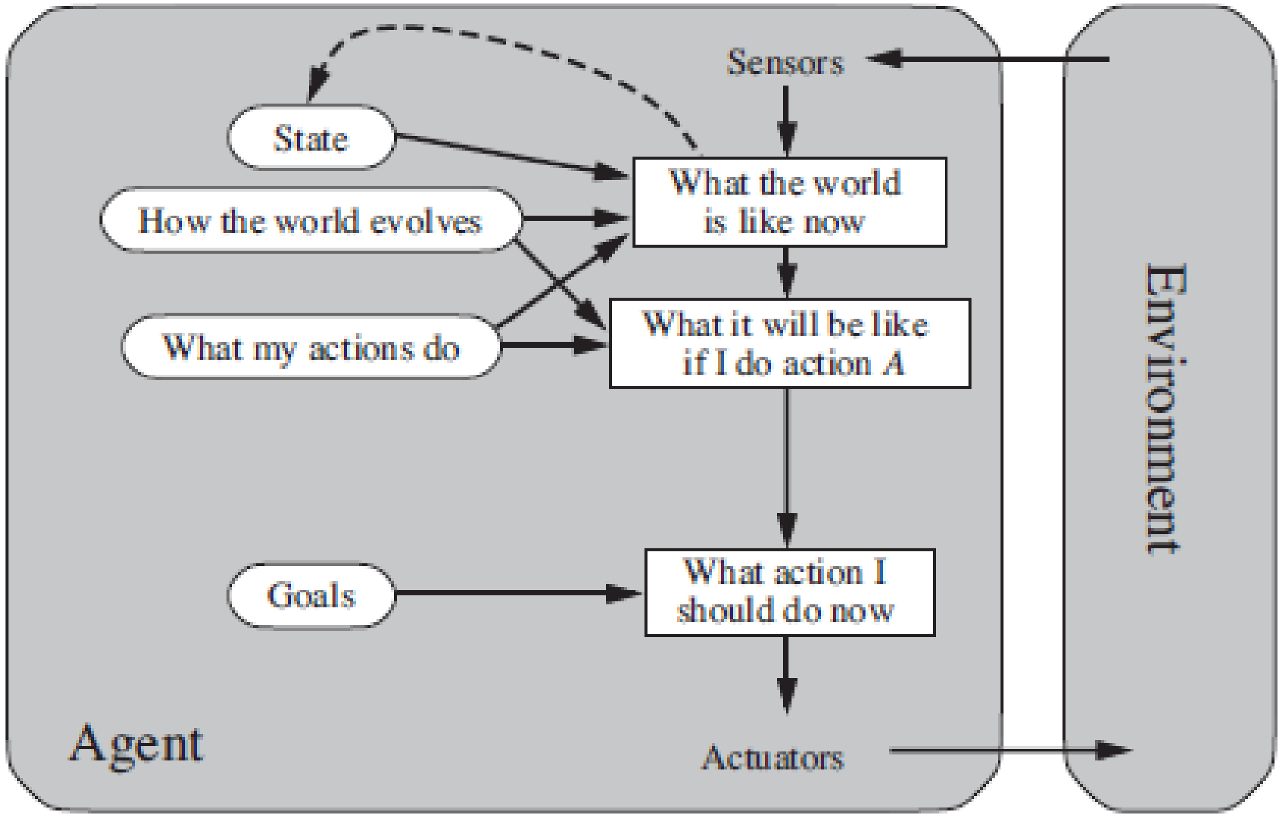

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Goals
What action I should do now
Agent
Actuators
Environment

Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Utility-based agents

>> High-quality behavior in most environments – enter “**utility**”

>> An agent’s **utility function** is essentially an internalization of the performance measure.

>> Remember – this is not the only way to be rational

>> Decision making under uncertainty => **expected utility**

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Utility-based agents

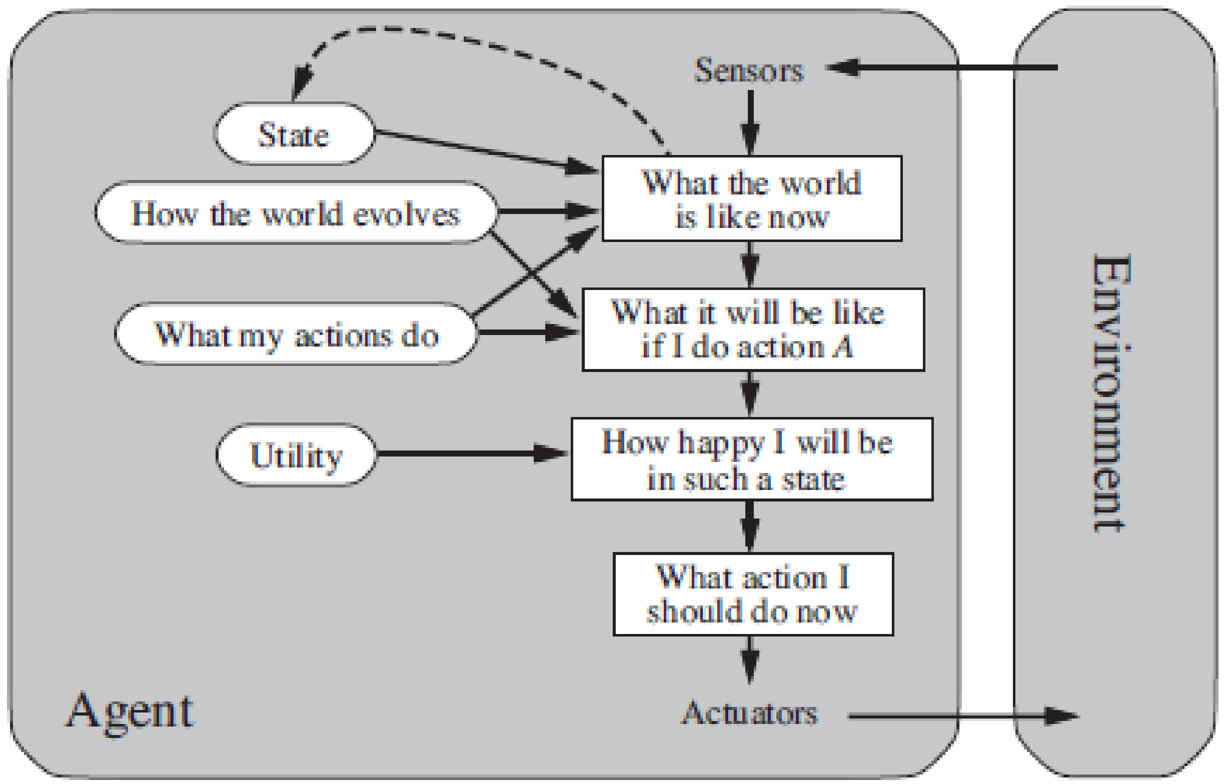
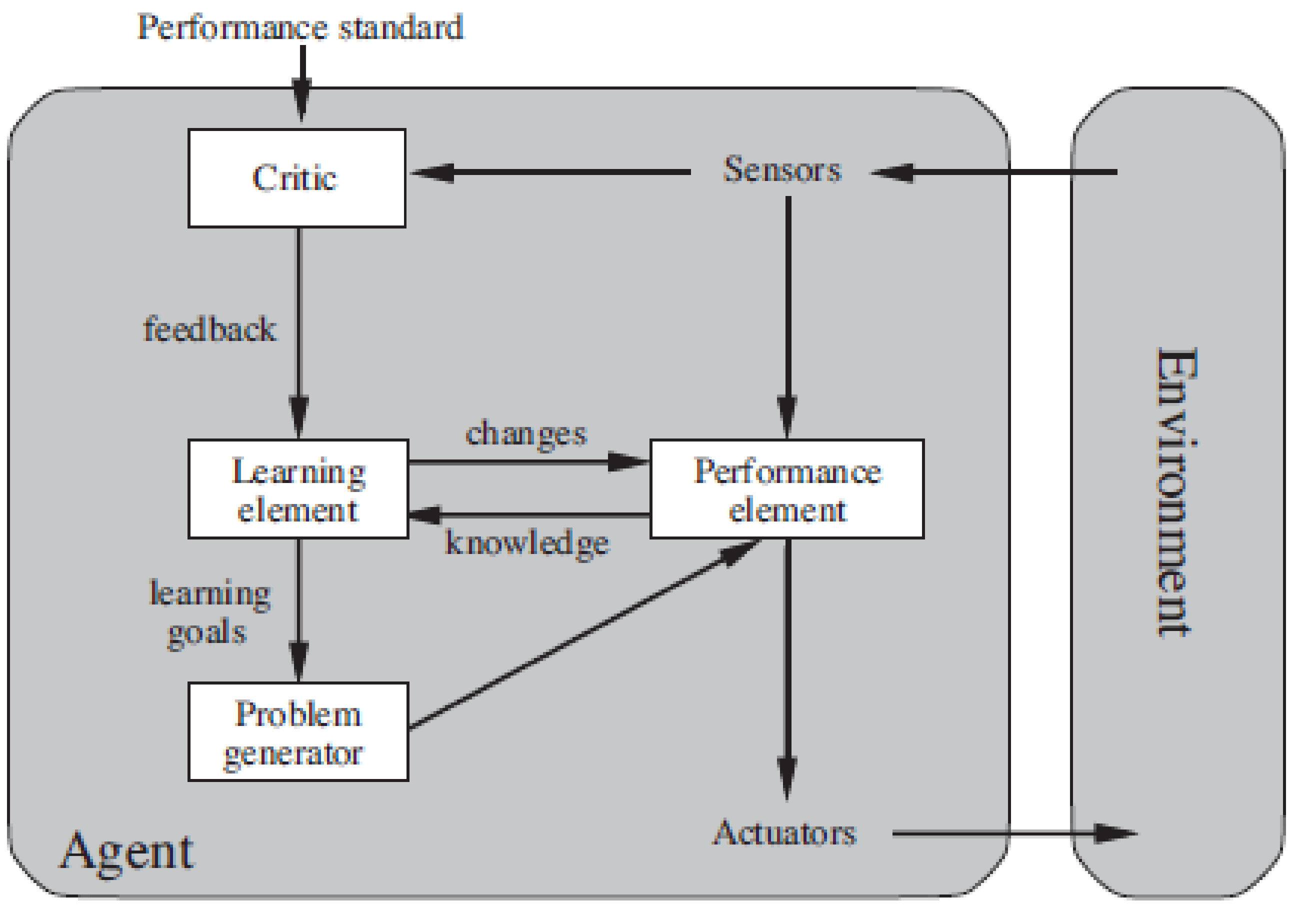

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
How happy I will be in such a state
Utility
What action I should do now
Agent
Actuators
Environment

Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Learning agents

>> A learning agent can be divided into four conceptual components, as depicted



How the components of agent programs work agents

>> We have described agent programs (in very high-level terms) as consisting of various components, whose function it is to answer questions such as: "What is the world like now?" "What action should I do now?" "What do my actions do?" The next question for a student of AI is, "How on earth do these components work?"

>> Some basic distinctions among the various ways that the components can represent the environment that the agent inhabits.

>> We can place the representations along an axis of increasing complexity and expressive power—atomic, factored, and structured. To illustrate these ideas, it helps to consider a particular agent component, such as the one that deals with "What my actions do."

How the components of agent programs work agents

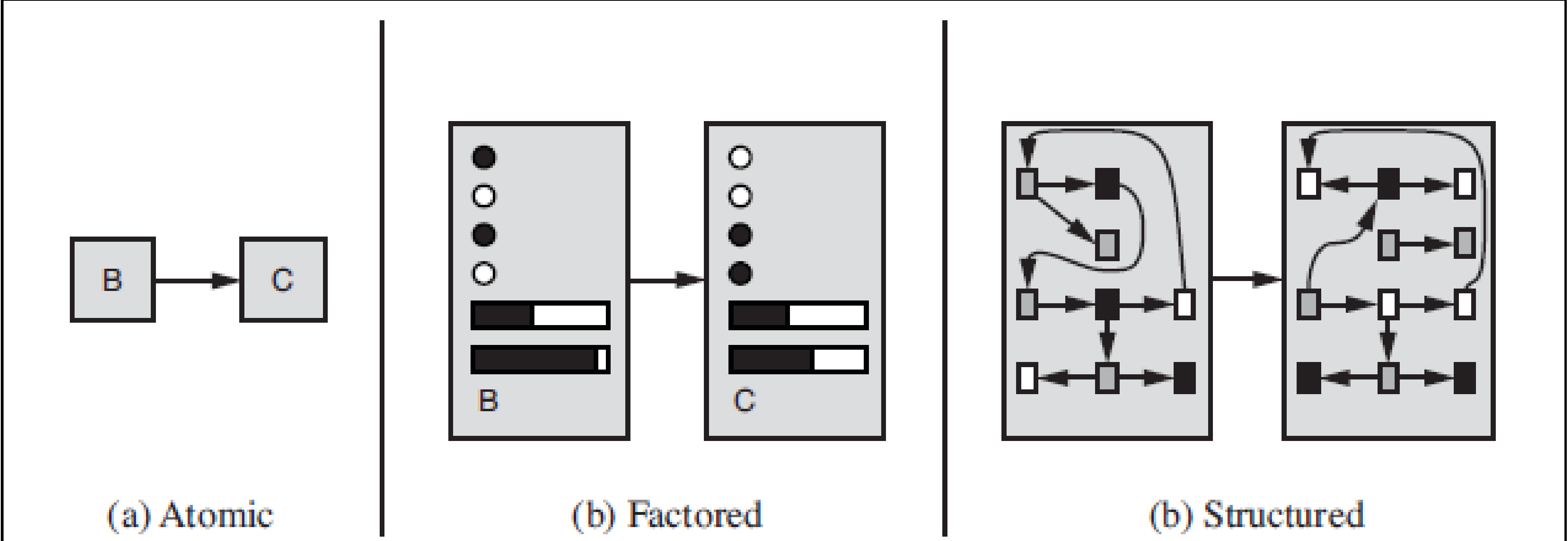


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

Expressiveness

>> The *axis along which atomic, factored, and structured representations lie is the axis of increasing expressiveness.*

>> Expressive language is much more concise. e.g. chess in first order logic as opposed propositional logic

>> Reasoning and learning become more complex as the expressive power of the representation increases.

To gain the benefits of expressive representations while avoiding their drawbacks, intelligent systems for the real world may need to operate at all points along the axis simultaneously.

References

[1] Russell, S. and Norvig, P., 2002. Artificial intelligence: a modern approach – Chapter 1