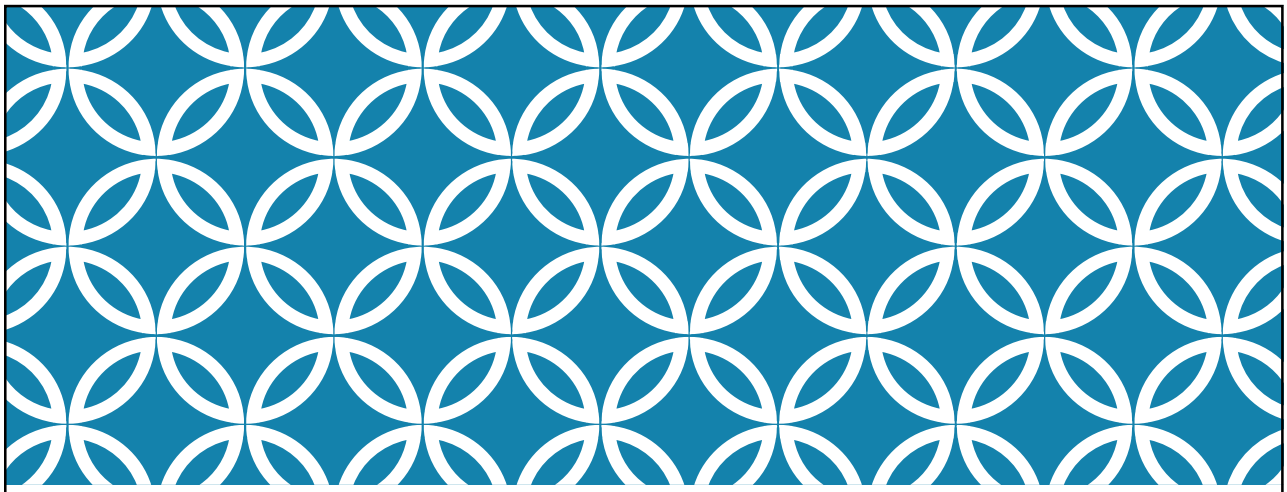


ARTIFICIAL INTELLIGENCE

1

RECAP

AI?

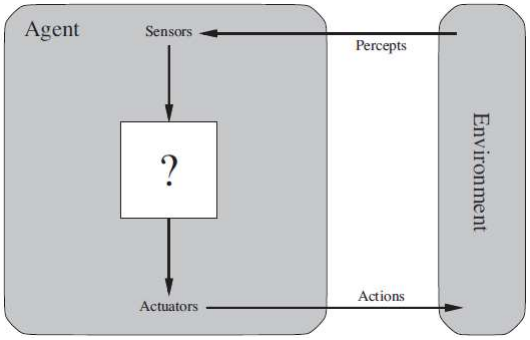
Agent?

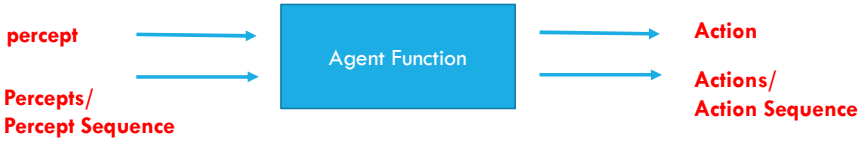
2

AGENTS AND ENVIRONMENT

We use the term **percept** to refer to the agent's perceptual inputs at any given instant.

Agent's **percept sequence** is the complete history of everything the agent has ever perceived.





3

TYPES OF TASK ENVIRONMENT

Fully observable vs. **partially observable**

Single agent vs. **multiagent**

Deterministic vs. **stochastic**

Episodic vs. **sequential**

Static vs. **dynamic**

Discrete vs. **continuous**

Known vs. **unknown**

4

FULLY OBSERVABLE VS. PARTIALLY OBSERVABLE

Fully observable vs. **partially observable**
Single agent vs. multiagent
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. unknown

If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable.

Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.

An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.

For example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.

5

SINGLE VS MULTI AGENT

Fully observable vs. partially observable
Single agent vs. **multiagent**
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. unknown

An agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two agent environment.

For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a **competitive** multiagent environment.

In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative** multiagent environment.

6

DETERMINISTIC VS. STOCHASTIC

Fully observable vs. partially observable
Single agent vs. multiagent
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. unknown

If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic (or non-deterministic).

In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment. If the environment is partially observable, however, then it could *appear* to be stochastic.

Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; The vacuum world is deterministic.

7

EPISODIC VS SEQUENTIAL

Fully observable vs. partially observable
Single agent vs. multiagent
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. unknown

In each episode the agent receives a percept and then performs a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. E.g. spotting faults in an assembly line.

In sequential environments, on the other hand, the current decision could affect all future decisions.

Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences.

Episodic environments are much simpler than sequential environments because the agent does not need to think ahead

8

STATIC VS DYNAMIC

Fully observable vs. partially observable
Single agent vs. multiagent
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. unknown

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static.

Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time.

If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **semidynamic.**

Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm decides what to do next. Chess, when played with a clock, is semidynamic. Crossword puzzles are static.

9

CONTINUOUS VS DISCRETE

Fully observable vs. partially observable
Single agent vs. multiagent
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. unknown

The discrete/continuous distinction applies to the *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent.

Chess -> Discrete states, percepts and actions

Taxi Driving -> Continuous state-time problem including actions like steering angles, although location, speed of taxi and camera input can be "discretized".

10

KNOWN VS UNKNOWN

Fully observable vs. partially observable
Single agent vs. multiagent
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Known vs. **unknown**

Strictly speaking, this distinction refers not to the environment itself but to the agent's (or designer's) state of knowledge about the "laws of physics" of the environment.

In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.

The distinction between known and unknown environments is not the same as the one between fully and partially observable environments.

It is quite possible for a *known* environment to be *partially* observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over.

Conversely, an *unknown* environment can be *fully* observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

11

ENVIRONMENT ASSESSMENT: UNKNOWN

Known: If map of Romania wasn't available, then it would be Unknown

In Unknown environment, we don't know the outcome/resulting state of current action.

E.g. in case of absence of map, out of three actions from Arad, Go Sibiu, Go Timisoara or Go Zerind?

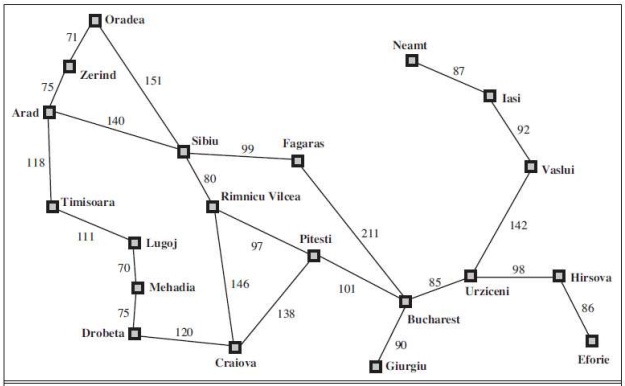


Figure 3.2 A simplified road map of part of Romania.

12

ENVIRONMENT ASSESSMENT: OBSERVABLE

Observable: Agent known which city its currently in.

Assuming: Each city/town has signboards with city name.

In observable environment, agent knows which state its currently in.

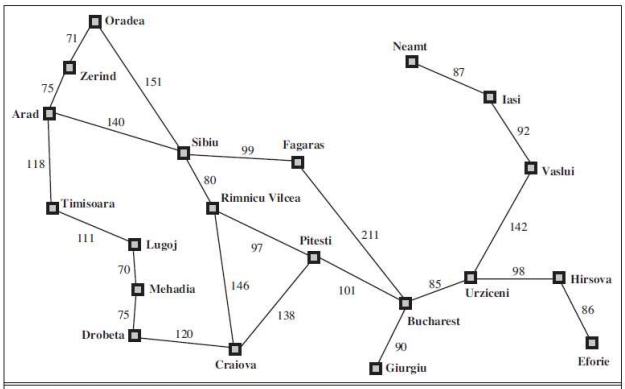
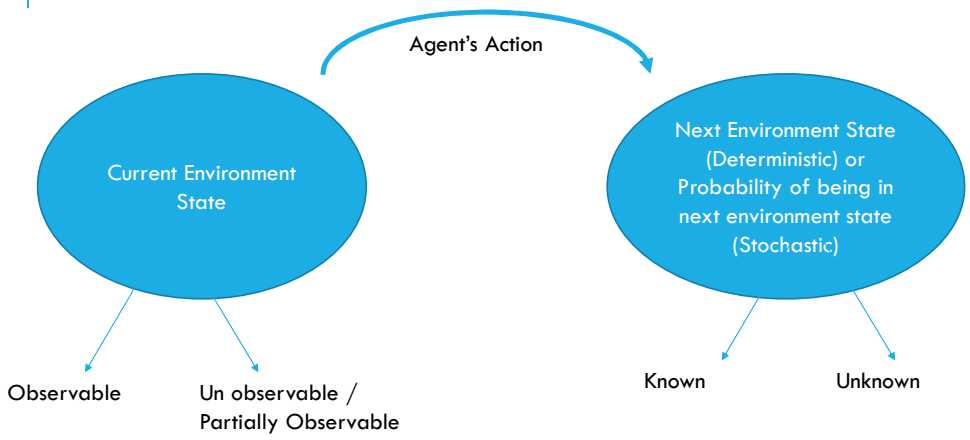


Figure 3.2 A simplified road map of part of Romania.

13

UNKNOWN VS UNOBSERVABLE VS STOCHASTIC



14

ENVIRONMENT ASSESSMENT: DISCRETE

We also assume the environment is **discrete,** so at any given state there are only finitely many actions to choose from.

At most 4 roads leave from one city/state (Bucharest).

So possible actions from Bucharest {Go Guirgin, Go Urziceni, Go Pitesti , Go Fagaras}

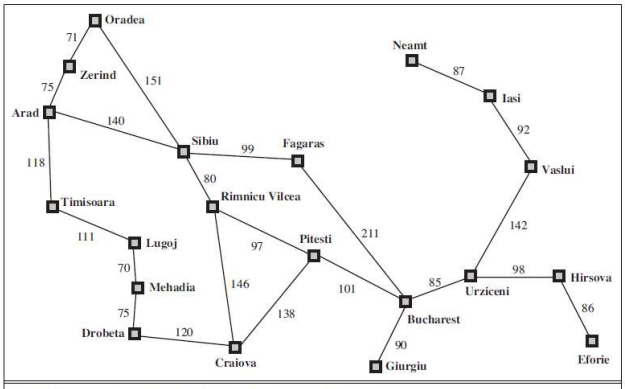


Figure 3.2 A simplified road map of part of Romania.

15

ENVIRONMENT ASSESSMENT: DETERMINISTIC

we assume that the environment is **deterministic,** so each action has exactly one outcome.

If an agent chooses to drive from Arad to Sibiu, it does end up in Sibiu.

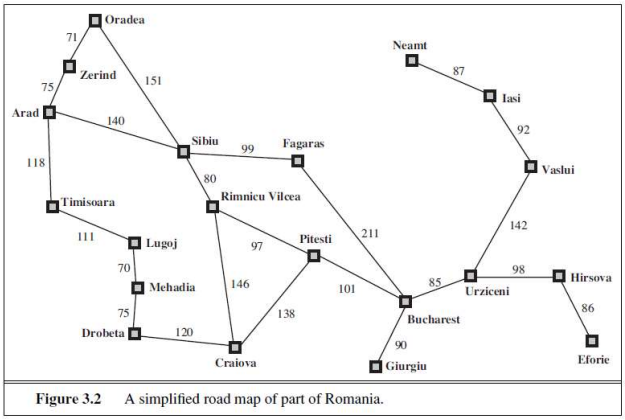


Figure 3.2 A simplified road map of part of Romania.

Known → Deterministic

Known → Stochastic (If an agent chooses to drive from Arad to Sibiu, the probability of arriving in Sibiu is 0.8 (not the case here))

16

17

AGENTS: STRUCTURE

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 = Program + Architecture

18

AGENT TYPES

simple reflex agents

reflex agents with state

goal-based agents

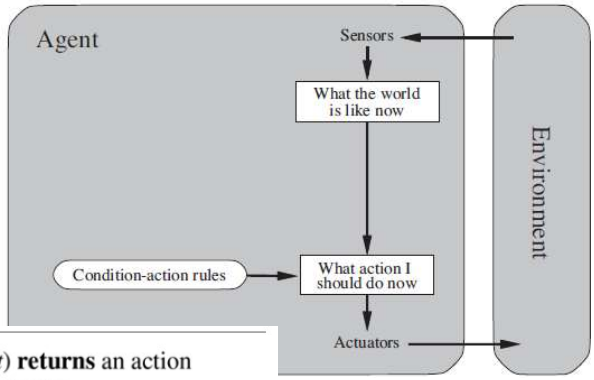
utility-based agents

All of these 4 types can be turned into learning agents

19

SIMPLE REFLEX AGENT

if *car-in-front-is-braking* **then** *initiate-braking*



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*

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

20

CATCH:

"only if the correct decision can be made on the basis of just the current percept—that is, only if the environment is fully observable."

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 in some form.

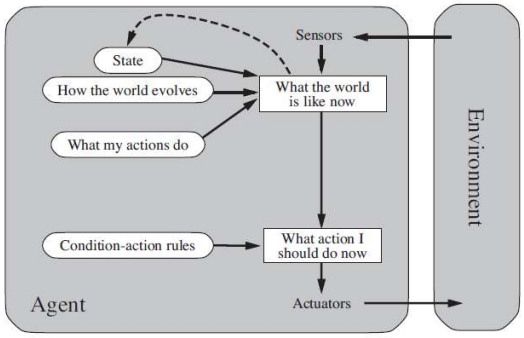
- First, we need some information about how the world changes over time, which can be divided roughly into two parts:
- the effects of the agent's actions and
- how the world evolves independently of the agent.

21

MODEL-BASED REFLEX AGENTS

How the world evolves independently of agent
e.g., road is slippery during rain

What my actions do e.g., when steering wheel turns clockwise, car moves to the right



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*

22

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

- For example, at a road junction, the taxi can turn left, turn right, or go straight on.
- The correct decision depends on where the taxi is trying to get to.

In other words,

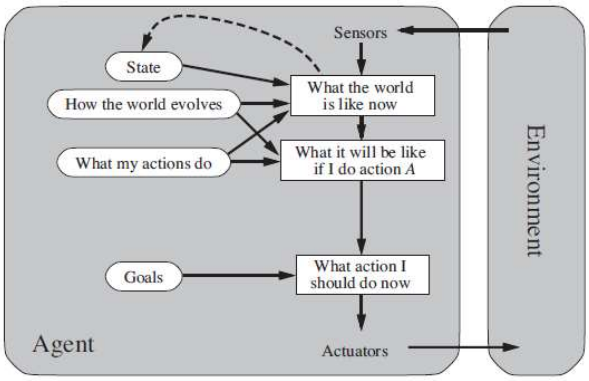
- as well as a current state description,
- the agent needs some sort of **goal** information that describes situations that are desirable

23

GOAL-BASED AGENTS

Sometimes an agent needs some sort of **goal** information that describes situations that are desirable—for example, being at the passenger's destination.

"Searching" and "Planning" are subfields of AI that describe such agents.



A model-based, goal-based agent

24

Goals alone are not enough to generate high-quality behavior in most environments.

For example, many action sequences will get the taxi to its destination (thereby achieving the goal), but some are quicker, safer, more reliable, or cheaper than others

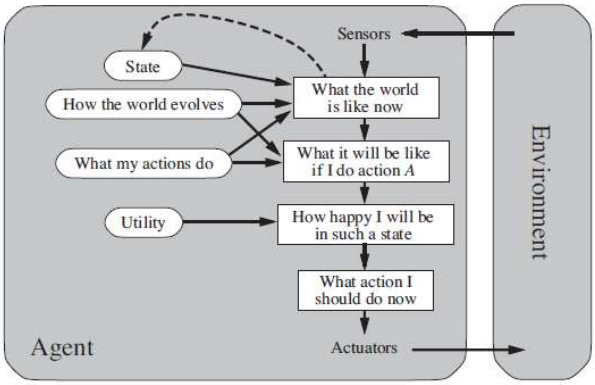
25

UTILITY-BASED AGENTS

Many ways to achieve a "goal" e.g. to reach a particular destination but which one is quicker, cheaper and more reliable?

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

A model-based, utility-based agent chooses the action that leads to the best expected utility, that is, the utility the agent expects to derive, on average, given the probabilities and utilities of each outcome.



A model-based, utility-based agent

26

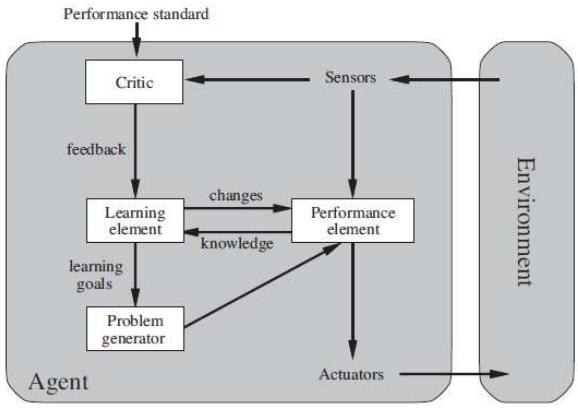
LEARNING AGENTS

Learning Element: making improvements

Performance Element (Agent in previous slides): selecting external actions.

The critic tells the learning element how well the agent/performance element is doing with respect to a fixed performance standard.

Problem Generator: suggesting actions that will lead to new and informative experiences, doing suboptimal tasks in shortrun for better overall experience in longrun.



27