TDT4136 Introduction to Artificial Intelligence

Chapter 2 - Intelligent Agents

Pinar Öztürk

Norwegian University of Science and Technology
2023

From last week

- Systems that think like humans
- Systems that think rationally
- Systems that act like humans
- **Systems that act rationally**

Outline

What is an agent?

Rationality

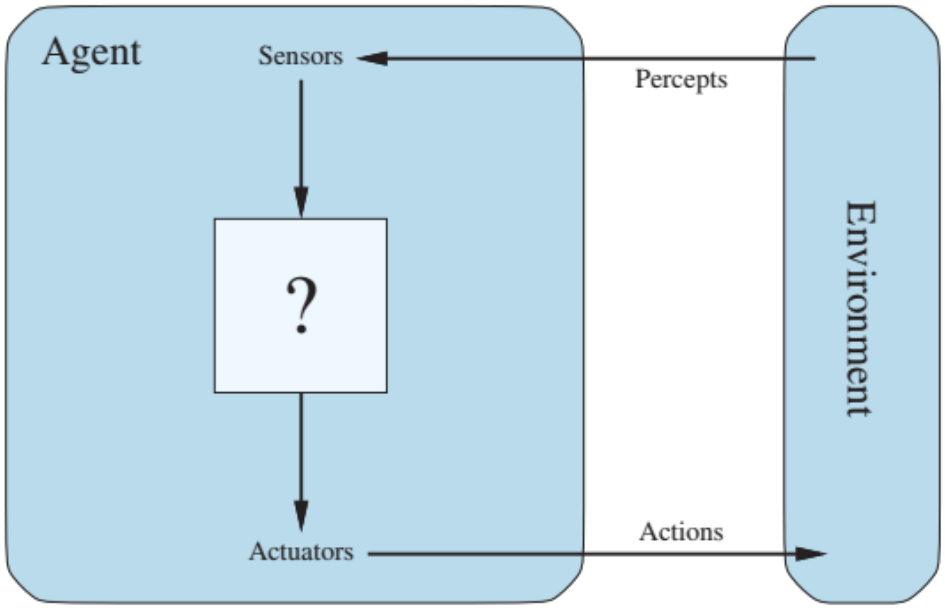
PEAS (Performance measure, Environment, Actuators, Sensors)

Environment types

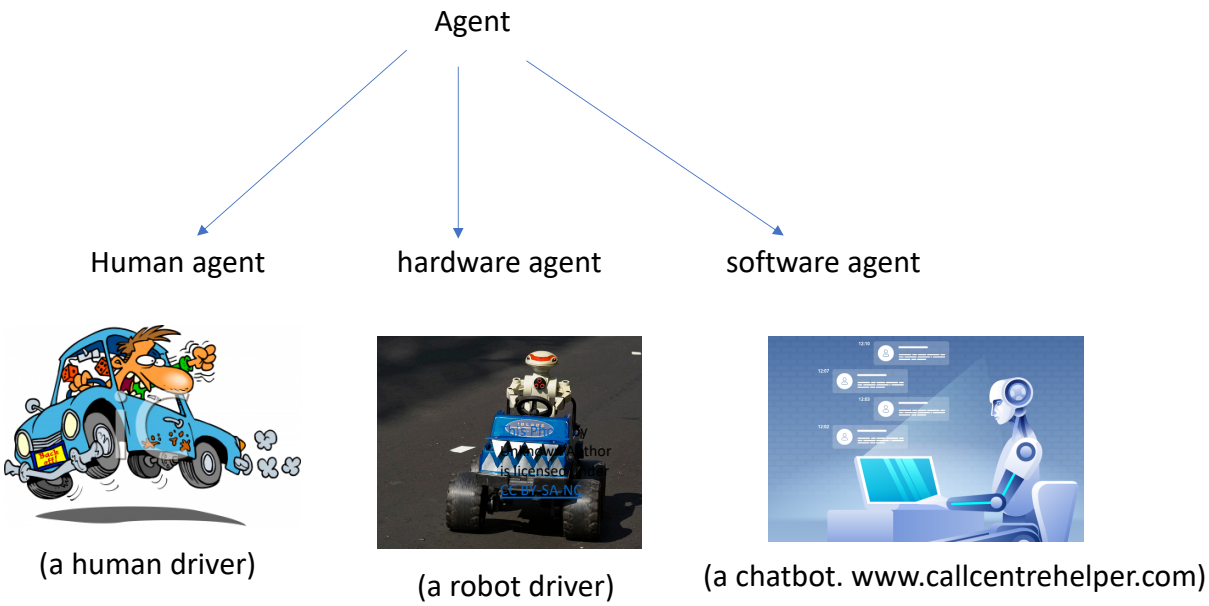
Agent types

What is an agent

An agent is anything that can be viewed as *perceiving* its *environment* through *sensors* and acting upon that environment through *actuators*



What is an agent



robot:

perceives: with video-cameras, infra-red sensors, radar, ...
acts: with wheels, motors

softbot:

perceives: receiving keystrokes, files, network packets, ...
acts: displaying on the screen, writing files, sending network packets

. . . .

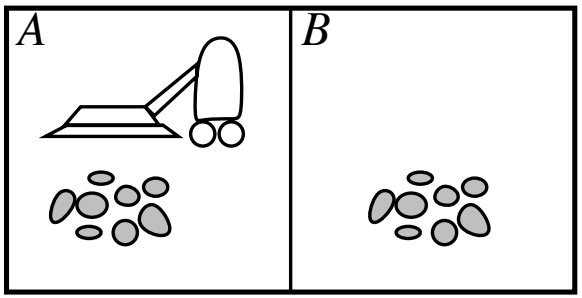
Agent's behaviour

An agent's choice of action at any given instant can depend on the entire percept sequence observed to date

Hence, agent's behaviour is described by the **agent function** that maps any percept sequence to an action.

A simple agent function

- Example: vacuum cleaner world. If the current square is dirty, then suck; otherwise, move to the other square.
- Tabulating the vacuum's agent function:



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*
⋮	⋮

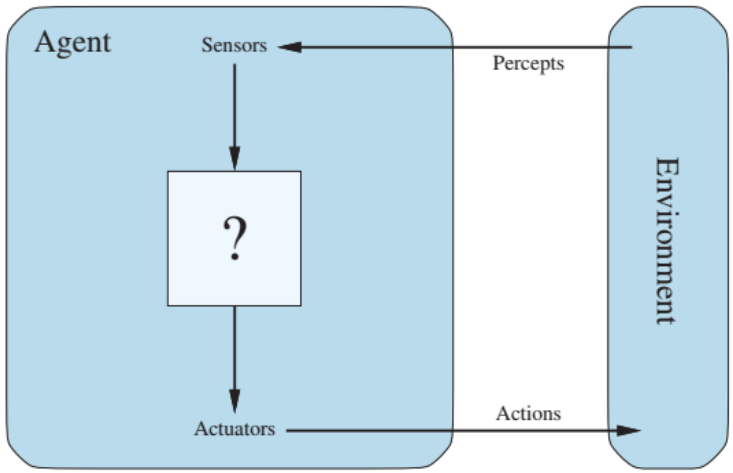
What is the right way to fill this table?

Agent Behaviour

Agent function - math description of behaviour, maps from percept sequences to actions:

$$f : \mathcal{P} \rightarrow \mathcal{A}$$

Agent program - concrete implementation of agent function, runs on the physical *architecture* to produce f



Rational agent

- 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

Performance measure

Performance measure evaluates the environment sequence - consequentialism.

- For example, in case of the vacuum cleaner, the designer could focus only on collecting as much dirt as possible in time T
- Or, take also other factors into account, e.g :
  - amount of time taken to clean
  - amount of electricity consumed
  - amount of noise generated, etc
- e.g., one point per square cleaned up in time T, or
  - one point per clean square per time step, minus one per move
  - etc

Rational agent

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.

Is the "tabulated vacuum agent" rational under these conditions:

- performance measure gives 1 point for each clean room at each time step, over a lifetime of 100 time steps
- there are 2 rooms, the agent has location and dirt sensors. Clean room stays clean, and "suck" works properly
- available actions are right, left, suck
- no noise or error in sensors

Rational vs omniscient vs perfect

Performance measure must be realistic

Rational $\neq$ omniscient agent which knows the actual outcome of its actions

Rational $\neq$ perfect agent which maximizes the actual performance

Rationality requires learning and autonomy

- Learning agent modifies knowledge of the environment from experience

- 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
- 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....

Task Environment

The task environment is defined through "PEAS":

- Performance measure
- The agent's prior knowledge of the environment
- The actions that the agent can perform
- The agent's percept sequence to date

Task Environment

PEAS (**P**erformance measure, **E**nvironment, **A**ctuators, **S**ensors)

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

Medical diagnosis system agent

Performance measure??

Environment??

Actuators??

Sensors??

Medical diagnosis system agent

Performance measure : healthy patient, minimize costs, lawsuits

Environment: patient, hospital, staff

Actuators: screen display (questions, tests, diagnoses, treatments, referrals)

Sensors: keyboard (entry of symptoms, findings, patient's answers)

Environment Properties

Fully observable environments

- Relevant parts of the state of the environment can be sensed
- No need to maintain any internal state to keep track of the world
- Example: chess, image analysis

Partially observable environments

- Parts of the environment cannot be sensed
- Agent must make informed guesses about world
- Example: poker, taxi driving, medical diasgnosis

Environment Properties

Single agent

- No other agents - there may be but as a part of the environment
- Examples: medical diagnosis, image analysis

Multi-agent

- Which entities will be viewed as "other agents"?
- The environment contains other agents whose performance measure depends on my actions and vice versa
- Competitive and cooperative interactions
- Examples: poker, chess, taxi driving

Properties of environments

Deterministic environment

Any action has a single guaranteed effect, and no uncertainty/failure.

Example: chess, image analysis

Non-deterministic environment

- There is some uncertainty about the outcome of an action
- Multiple outcome alternatives
- Called "stochastic" if alternatives are quantified in terms of probabilities
- Example: poker, taxi driving, medical diagnosis

Properties of environments

Episodic environments

- The agent's experience is divided into atomic episodes.
- Each episode consists of the agent perceiving and then performing a single action
- The choice of action in each episode depends only on the episode itself
- Examples: image analysis

Sequential environments

- The current decision could affect all future decisions
- Examples: poker, chess, taxi driving, medical diagnosis

Properties of environments

Discrete

Finite number of distinct states, percepts and actions,

Examples: chess, poker

Continuous

Continuous time/state/actions

Example: taxi driving, medical diagnosis, image analysis

Properties of environments

Dynamic environment

- May change while an agent is deliberating
- Examples: taxi driving, medical diagnosis

Static environment

- The environment does not change
- . Examples: poker

Semidynamic

- The world does not change but the agent's performance score may
- . Examples: chess with clock, image analysis

Environment	Observable	Deterministic	Episodic	Discrete	Dynamic	Agent
Crossword	fully	deterministic	sequential	discrete	static	
Chess w/clock	fully	deterministic	sequential	discrete	semidynamic	
Backgammon	fully	stochastic	sequential	discrete	static	

Properties of environments

Known environment

- The agent's knowledge about how the environment works/evolves
- Note that a known environment (i.e., the agent knows all the rules that apply) may be only partially observable

Unknown environment

- The agent will have to learn how it works
- An unknown environment can be fully observable.

A simple vacuum cleaner agent

Table-driven agent:

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*
$\vdots$	$\vdots$

Table-driven agent

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*

- looks up the right response in the table
- Infeasible: if there are $|P|$ percepts and a life-time of T, then need for a look-up table of size $\sum_{t=1}^{T} |P|^t$

Agent programs

Writing down such agent functions is not feasible

The goal of AI is to write agent programs with small code which produced the desired rational behaviour

Agent types

- simple reflex agents
- model-based reflex agent
- goal-based agents
- utility-based agents
- learning agents

Simple reflex agent

- Uses only the current percept - ignores the percept sequence,
- Implemented through condition-action rules
- Large reduction in possible percept/action situations from $\sum_{t=1}^{T} |P|^t$ to $|P|$ [1]

Example:

```
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
```

[1] if there are $|P|$ percepts and a life-time of T, then need for a look-up table of size $\sum_{t=1}^{T} |P|^t$

Generic Simple Reflex Agent program

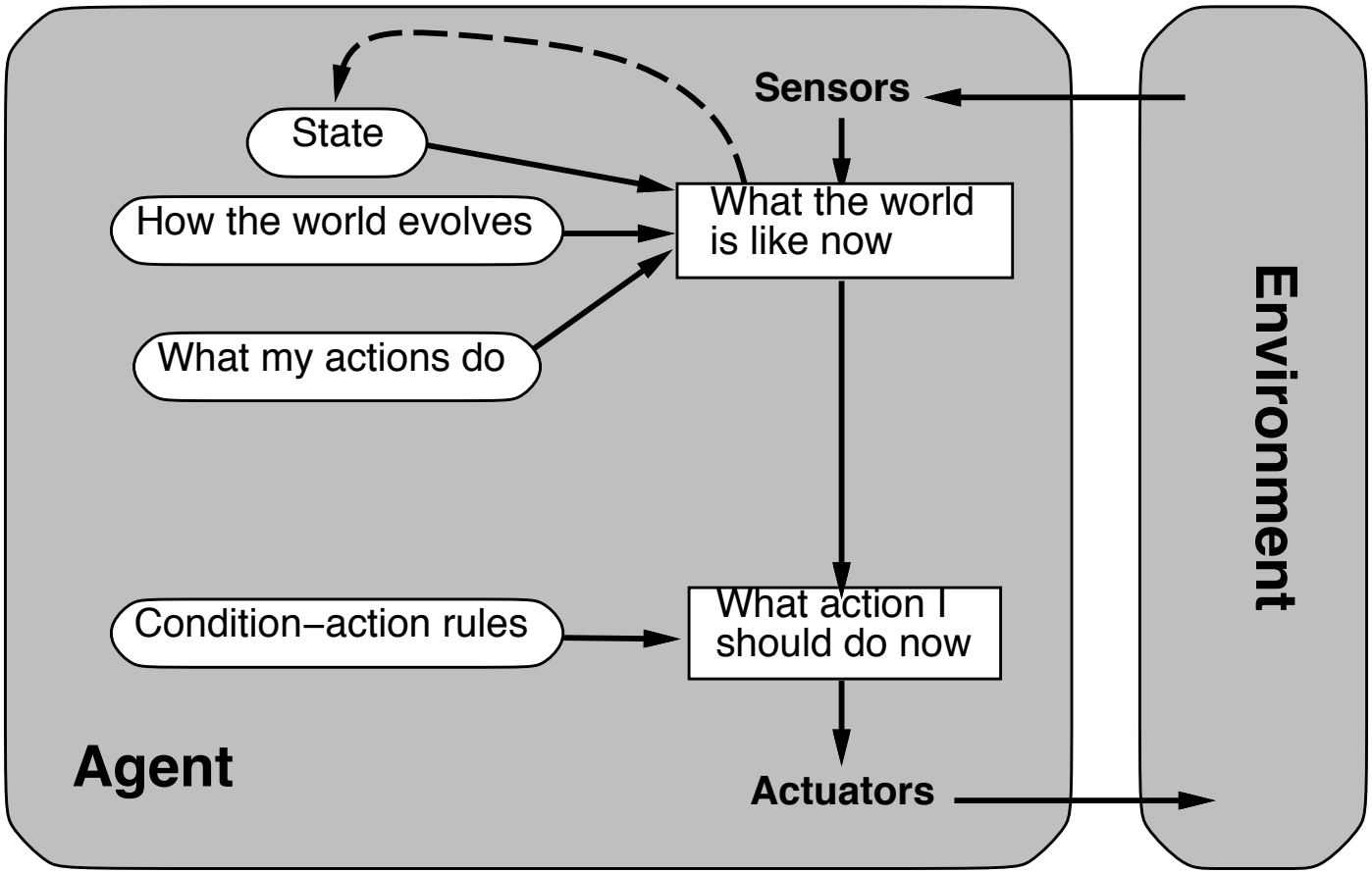
A more general simple reflex agent program:

```
function SimpleReflexAgent( percept) returns an action

 persistent rules
 state ← Interpret(percept)
 rule ← Rule-match(state, rules)
 action ← Rule-action(rule)
 return action
```

- Will only work if the environment is fully observable
- everything relevant needs to be determinable from the current input
- otherwise infinite loops may occur, e.g. in the vacuum world without a sensor for the room, the agent does not know whether to move right or left
  - any possible solution: ?

Model-based Reflex agents



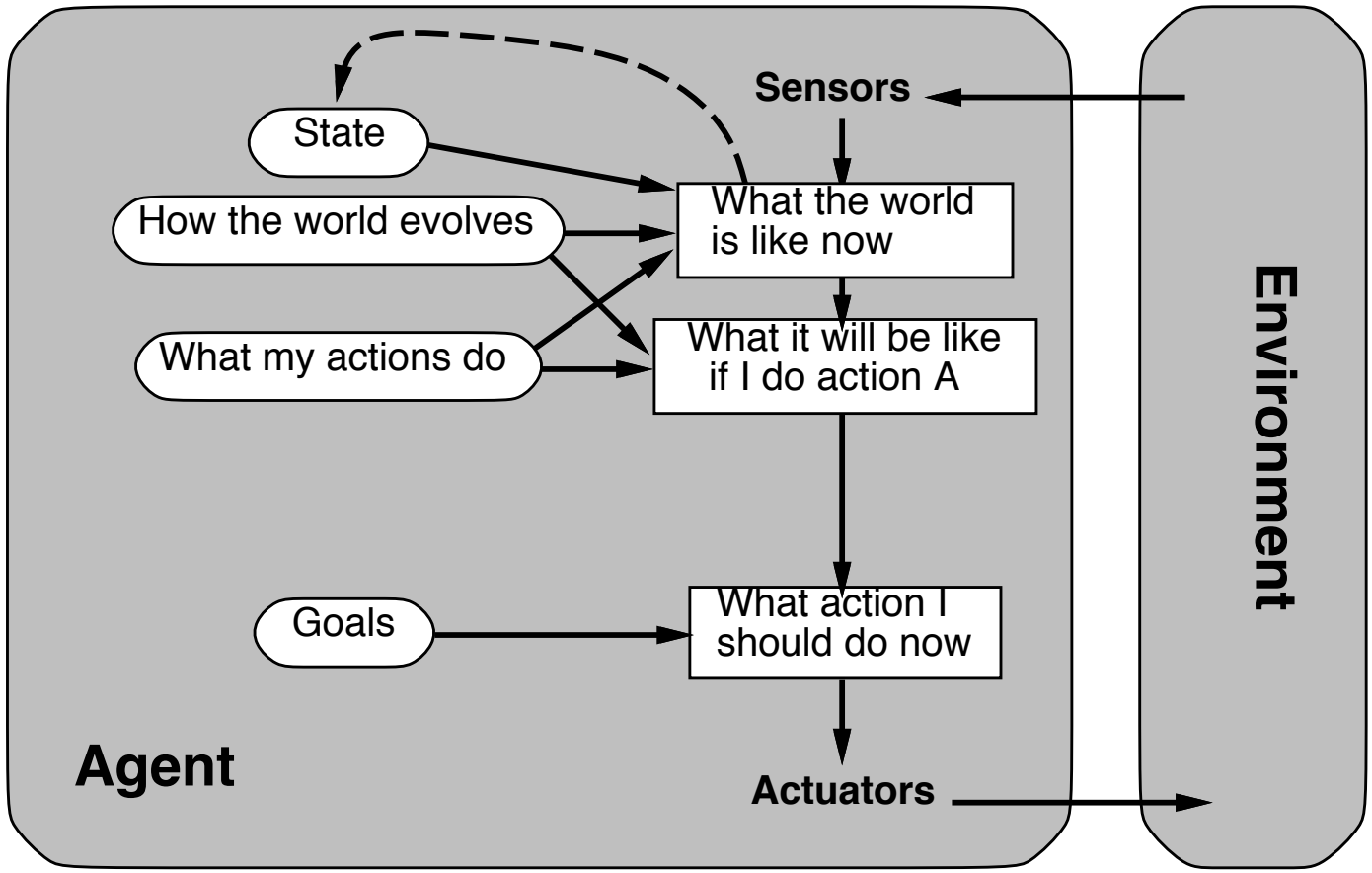
Model-based Reflex agents

function ModelBasedReflexAgent(*percept*) **returns** an action

persistent *state*, the agent's current conception of the world,
transition-model, a description of how the next state depends on
the current state and the action,
sensor-model, a description of how the current world state is reflected
in the agent's percepts
rules, a set of condition-action rules
action, the most recent action, initially none
state ← **UpdateState**(*state,action,percept,transition-model,sensor-model*)
rule ← **Rule-match**(*state*, *rules*)
action ← **Rule-action**(*rule*)
return *action*

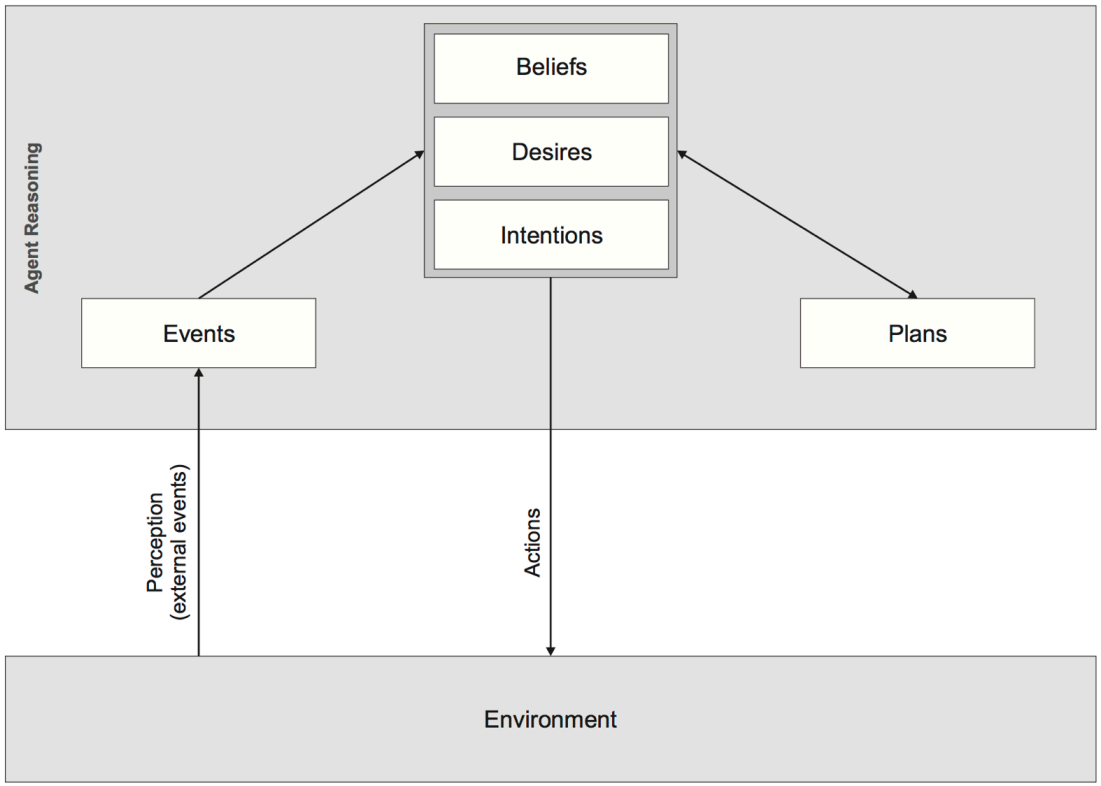
- difficult to exactly determine the current state in partially observable environments - independent from the kind of models used
- hence, may need to "guess" the current situation
- action decision in the same way as the simple reflex agent.

Goal-based agents



Example

- Belief Desire Intention agents (BDI) agents have a **mental state** as the basis for their reasoning.
- Three main mental attitudes: beliefs, desires and, intentions.
- Their reasoning is also called **practical reasoning** - e.g., contrary to deductive reasoning



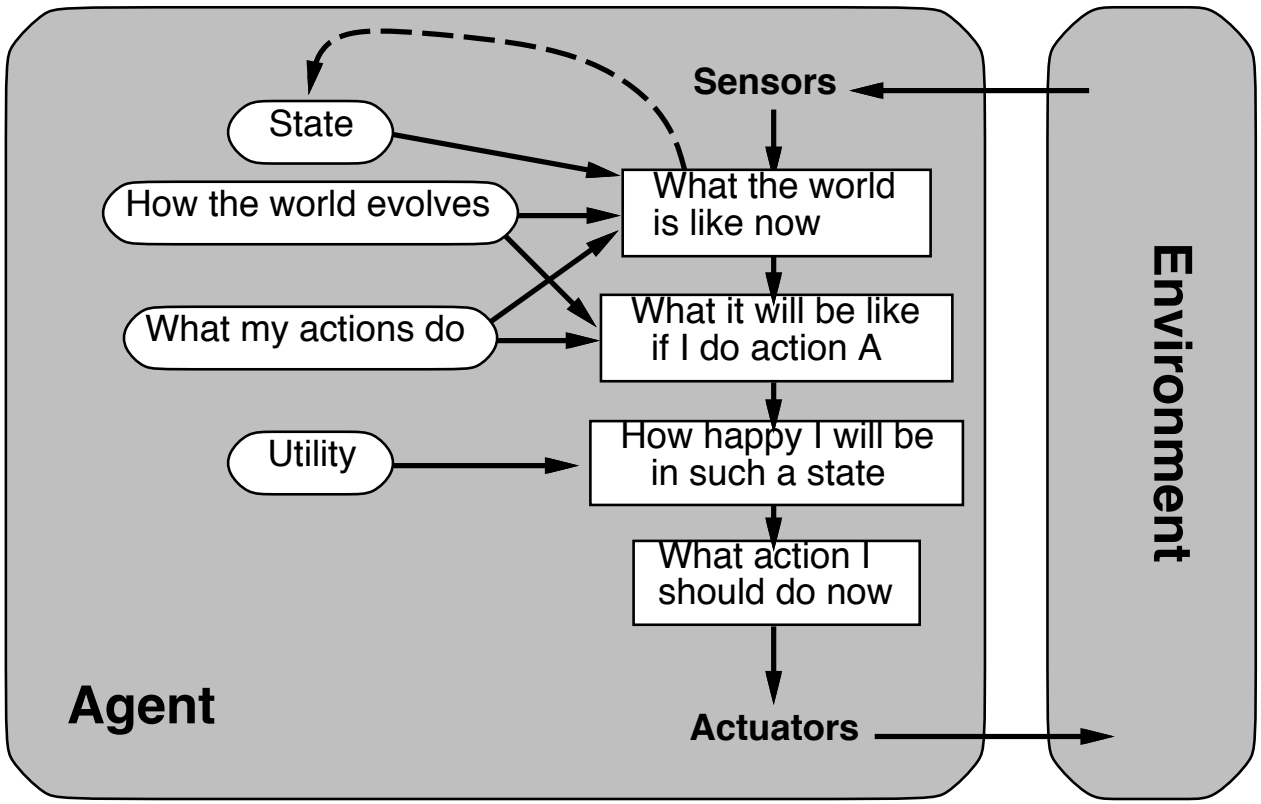
Goal-based agents

- explicit representation of goals: agent knows which states are desirable
- reasoning about goals
- more flexible since knowledge is represented explicitly and can be manipulated
- different reasoning methods than reflex - not condition-action rules and selection of rule/action
- main different from reflex agents: deliberates about future when making decision
- long sequence of actions may be needed to achieve the goal
  - e.g., agents in chapters 3-5 (search) and chapters 10-11 planning
- less efficient than reflex agents - but more flexible

Utility-based agents

Goals provide just a binary happy/unhappy disctinction while utility functions provide a continuous scale

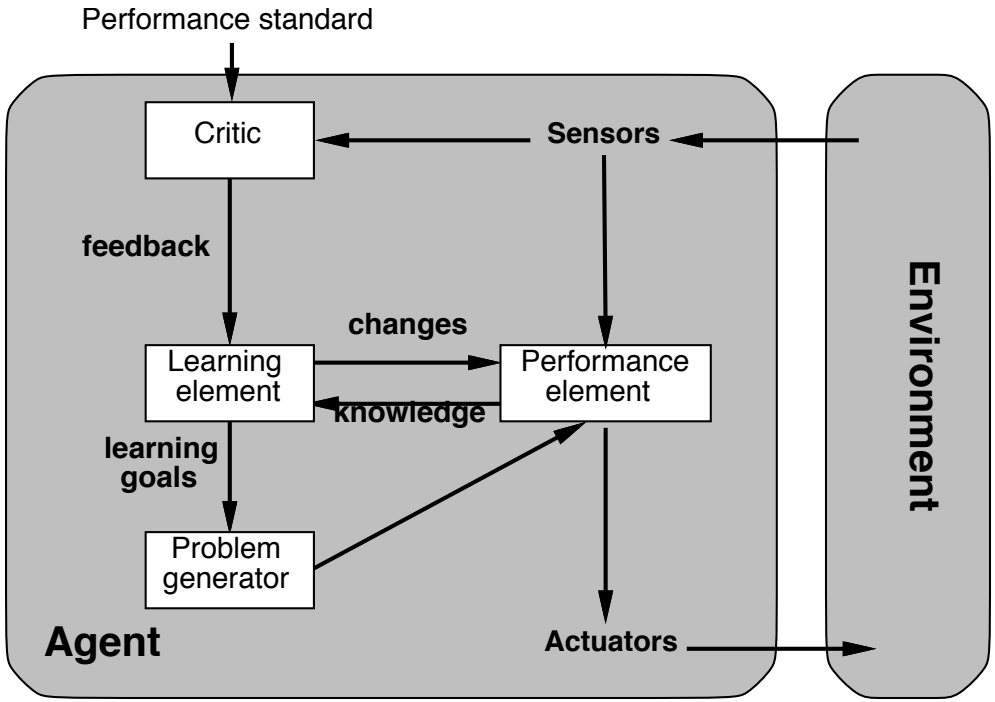
Some goals may be achieved in more than one way, with different utility values



Utility-based agents

- utilities are internalization of the performance measure
- Utility function maps a state (or a sequence of states) onto a real number
- utilities reflect agent's preferences
- not always know the utility of an action/outcome , hence "expected utility"
- agent chooses actions that maximize the expected utility of the outcomes

Learning agents



- Critic: evaluates current world state, determines how the performance should be modified
- Learning element: responsible for making improvements
- Problem generator: suggests explorations
- Performance element: responsible for selecting external actions

Representation of Environment

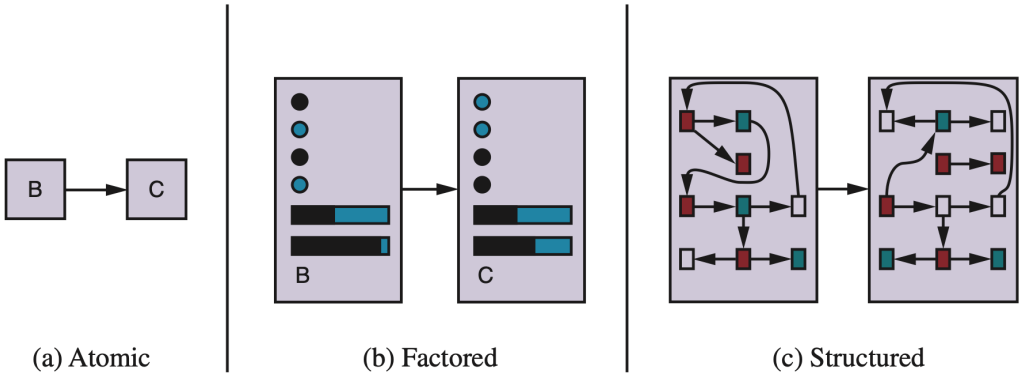
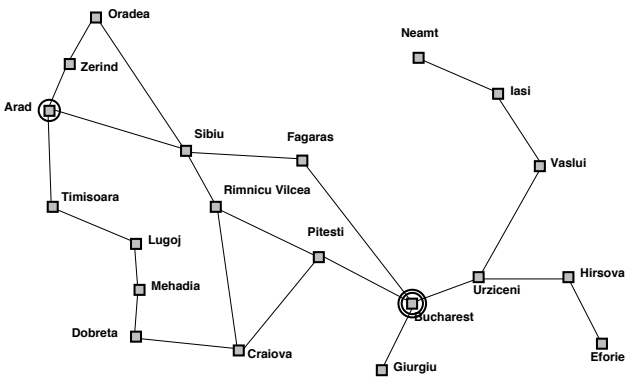


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed 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.

Next Week: Goal-based Agents

Goal-based agents that use search methods as Agent Function.
You are at the end of your holiday in Romania. Your return ticket is from Bucharest and you are leaving Arad for Bucharest.



- **Uninformed** Search method as Agent Function.
- In an environment:
  - Fully observable (the agent can see the current state of the world),
  - Deterministic (action has a single outcome)
  - Discrete
  - Known environment (knows which states can be reached through which actions - has a map of Romania.

The week after the next week: **Informed** search methods.