ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Intelligent Agents

Fundamentals of Artificial Intelligence

MSc in Applied Artificial Intelligence, 2023-24

Contents

- Topics included
  - Intelligent agents
  - Agent program types

- These slides were based essentially on the following bibliography:
  - Norvig, P, Russell, S. (2021). Artificial Intelligence: A Modern Approach, 4th Edition. Pearson, ISBN-13: 978-1292401133
  - The slides with a left blue bar are authored by Prof. Joaquim Gonçalves

Intelligent agents

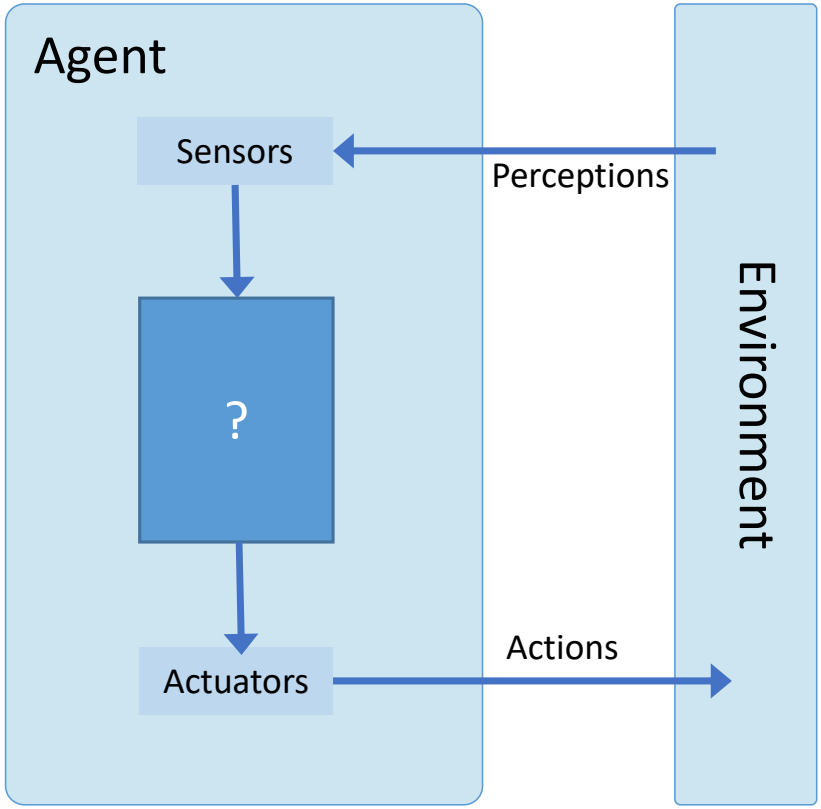
Agents and environments

- An **agent** is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators.**
- A **human agent** has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- A **robotic agent** might have cameras and infrared range finders for sensors and various motors for actuators.
- A **software agent** receives file contents, network packets, and human input (keyboard/mouse/touchscreen/voice) as sensory inputs and acts on the environment by writing files, displaying information or generating sounds.

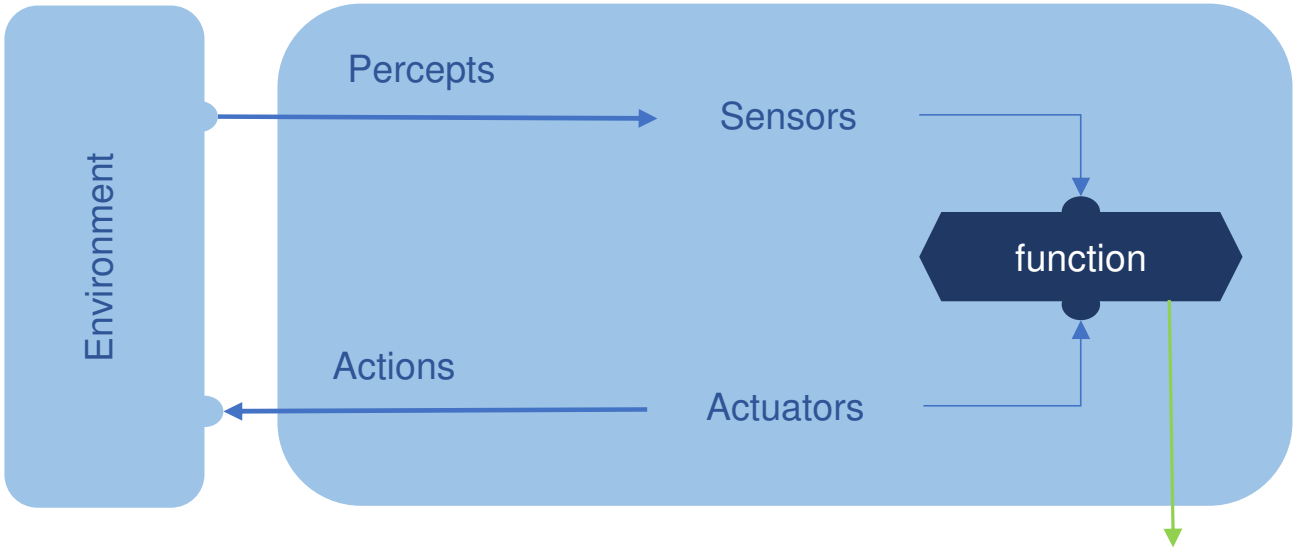
Retrieved on 2022.08.21 from https://hum3d.com/pt/3d-models/boston-dynamics-spot/

Agent function and program

- An agent's percept sequence is the complete history of everything the agent has ever perceived
  - The term **percept** refers to the content an agent's sensors are perceiving.
- We can build a **table** by trying out all possible percept sequences and recording which actions the agent does in response
  - It is an external characterization of the agent.
  - Internally, the agent function will be implemented by an agent program.
- The **agent function** defines mathematically the agent's behavior, mapping any given percept sequence to an action
- The **agent program** is a concrete implementation, running within some physical system



Agent Function

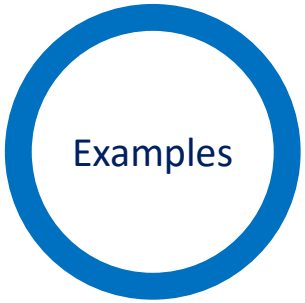


The agent function maps the history of perceptions to actions:

[f: P* → A]

The program executes physical the architecture to produce f

Agents



a human agent

Sensors: eyes, ears, ...

Actuators: hands, legs, mouth, ...

a robotic agent

Sensors: cameras, infrared range detectors, ...

Actuators: motors, ...

a software agent

Sensors: keystrokes, file content, network packets, ...

Actuators: displaying on the screen, recording files, network packets, ...

The Structure of Agents

- The job of AI is to design the **Agent = Architecture + Program**.
  - **Program** implements the agent function — the mapping from percepts to actions.
  - **Architecture is** some computing device with physical sensors and actuators that will run the program

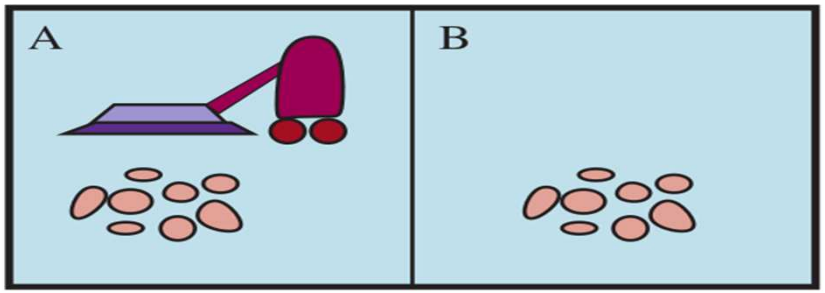
- Consider the design of an agent program that takes the current percept as input from the sensors and return an action to the actuators.
  - The agent program takes the current percept as input
  - The agent function may depend on the entire percept history.

- The agent program has no choice but to take 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.

Example of table for implementing the agent function

Consider a robotic vacuum-cleaning agent in a world consisting of squares that can be either dirty or clean

- The vacuum agent perceives which square it is in and whether there is dirt in the square.
- The agent starts in square A.
- The available actions are to move to the right, move to the left, suck up the dirt, or do nothing.
- One very simple **agent function** is the following:
  - if the current square is dirty, then suck;
  - otherwise, move to the other square.
- Table shows partial tabulation of this agent function.
- Note that the table is of unbounded size unless there is a restriction on the length of possible percept sequences.



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
...	...
[A,Clean], [A,Clean], [A,Clean]	Right
[A,Clean], [A,Clean], [A, Dirty]	Suck
...	...

The concept of rationality

- The question is "what makes an agent good or bad, intelligent or stupid?"
  - A **rational agent** is one that does the right thing.
- **Consequentialism**: evaluate an agent's behavior by its consequences
  - The representation of the performance measure could be explicit or implicit by design.
- Usually, the design of **performance measures** should be according to **what** one actually wants to be achieved, rather than according to **how** one thinks the agent should behave
  - 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 the built-in knowledge the agent has.
- What is **rational** at any given time depends on:
  - 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.

Rational agent

Retrieved on 2022.08.25 from
https://www.thoughtco.com/rational-rationale-and-rationalize-1689601

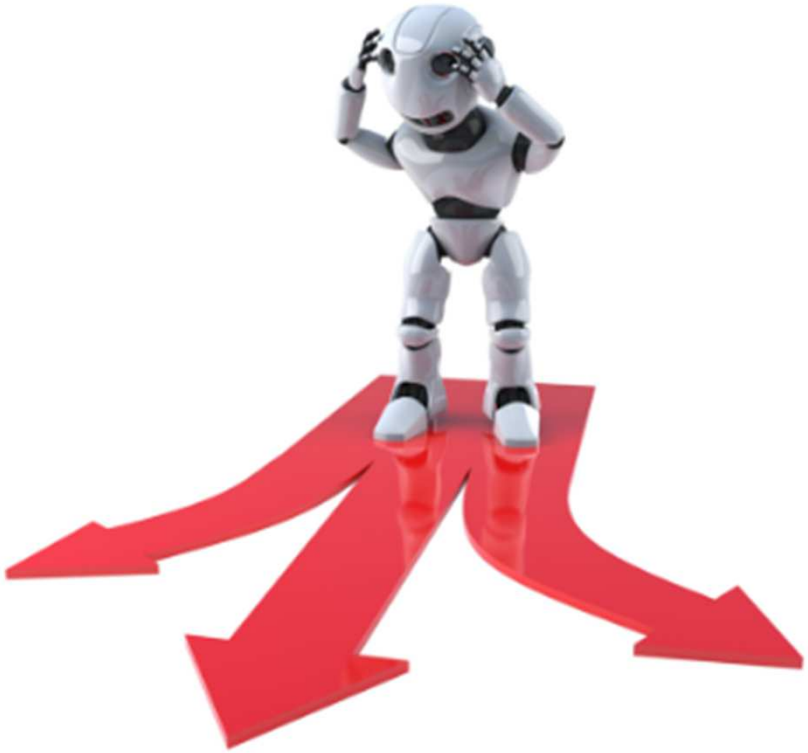
- Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not (see previous function table). **Is this a rational agent?**
- That depends! First, we need to say what the performance measure is, what is known about the environment, and what sensors and actuators the agent has. Let us assume the following:
  - The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
  - The "geography" of the environment is known a priori, but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square.
  - The Right and Left actions move the agent one square except when this would take the agent outside the environment, in which case the agent remains where it is.
  - The only available actions are Right, Left, and Suck.
  - The agent correctly perceives its location and whether that location contains dirt.

"*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.*"

Learning and autonomy

- A **rational agent** should not only to gather information from its sensors/inputs, but also to **learn** from what it perceives
  - The prior knowledge of the environment may be modified and augmented as the agent gains experience
  - There are extreme cases in which the environment is completely known a priori and completely predictable

- A rational agent should be **autonomous** and learn what it can to compensate for partial or incorrect prior knowledge.
  - An agent lacks autonomy if it relies only on the prior knowledge rather than on its own percepts and learning processes
  - For example, a vacuum-cleaning agent that learns to predict where and when additional dirt will appear will do better than one that does not.

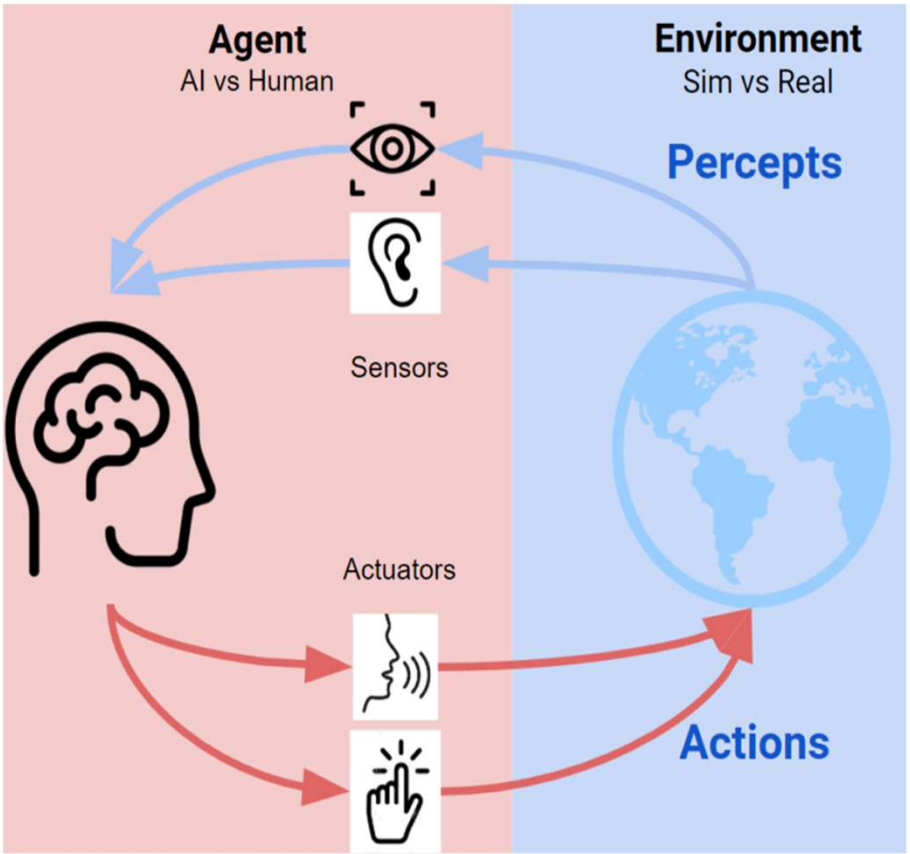


Retrieved on 2022.08.25 from http://www.scottoverton.ca/blog/2016/6/2/can-we-program-robots-to-make-ethical-decisions

Task environment

- Consists of the performance measure, the environment, and the agent's actuators and sensors (**PEAS)**:
  - **(P)erformance measures** need to be objective. Usually, several goals are combined, and tradeoffs are required if some of these goals conflict.
  - **(E)nvironment** includes the objects and constraints that the agent has to deal with. The environment of an automated car driver need operate in many distinct conditions The more restricted the environment, the easier the design problem.
  - **(A)ctuators** consist of the agent's components responsible for moving or controlling the mechanism or system
  - **(S)ensors** are the components that detects events or changes in its environment

Retrieved on 2022.08.25 from
https://siminsights.com/machine-learning-and-artificial-intelligence-ml-ai-services/



Examples of agent types and their PEAS descriptions.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable, maximize profits, minimize the risk of accidents	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	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

Properties of task environments

Dimensions that help categorize the task environment and decide the appropriate agent design.

- **Fully observable** vs. **partially observable** vs. **Unobservable**
  - **Fully observable** – If an agent's sensors give it access to the complete state of the environment at each point in time. The agent do not need to maintain any internal state to keep track of the world;
  - **Partially observable** – the agent can not access to the complete state of the environment
  - **Unobservable** – The agent know nothing about the environment state.

- **Single-agent** vs. **multiagent**
  - **Single-agent** – Perceives its environment through sensors and acts autonomously on it through actuators.
  - **Multi-agent** – System that contains two or more agents that interact to work cooperatively.
  - For example, 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.
  - A performance measure, criterion that defines the degree of success of the actions, must be associated with any agent or multi-agent system

Properties of task environments (2)

- **Deterministic** vs. **nondeterministic** vs. **stochastic**
  - **Deterministic** – If the next state of the environment is completely determined by the current state and the action executed by the agent(s).
  - **Nondeterministic** – The next state of the environment is not determined by the current state and the action executed by the agent
  - **Stochastic** – The nondeterministic scenarios are converted into stochastic by dealing with probabilities.
  - Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as nondeterministic

- **Episodic** vs. **sequential**
  - In an episodic task environment, the agent's experience is divided into atomic episodes. 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.
  - In sequential environments, the current decision could affect all future decisions.
  - Episodic environments are much simpler than sequential environments because the agent does not need to think ahead. Chess and taxi driving are sequential.

Properties of task environments (3)

- **Static vs. dynamic**
  - **Static** – the environment does not change while an agent is deliberating.
  - **Dynamic** – the environment can change while an agent is deliberating.
  - **Semi-dynamic** – the environment itself does not change with the passage of time but time is important for the agent's performance score.
  - 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.

- **Discrete vs. continuous**
  - It applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
  - For example, chess has a finite number of distinct states and a **discrete** set of percepts and actions.
  - Taxi driving is a **continuous**-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time.

Properties of task environments (4)

- **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 for all actions are given.
  - If the environment is **unknown**, the agent will have to learn how it works in order to make good decisions.

- The hardest case is
  - partially observable (we do no considered unobservable), multiagent, nondeterministic, sequential, dynamic, continuous, and unknown.
  - Taxi driving is hard in all these senses, except that the driver's environment is mostly known. Driving a rented car in a new country with unfamiliar geography, different traffic laws, and nervous passengers is a lot more exciting.

- The simplest environments are: episodic, single agent, fully observable, deterministic, static, discrete, and known.

Examples of task environments and their characteristics

Task environment	Observable	Agents	Deterministic vs Stochastic	Episodic vs Sequencial	Static vs Dynamic	Discrete vs Continuous
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi-dynamic	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-dynamic	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

The TABLE-DRIVEN-AGENT program

- The following pseudocode describes 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 represents explicitly the agent function that the agent program embodies.

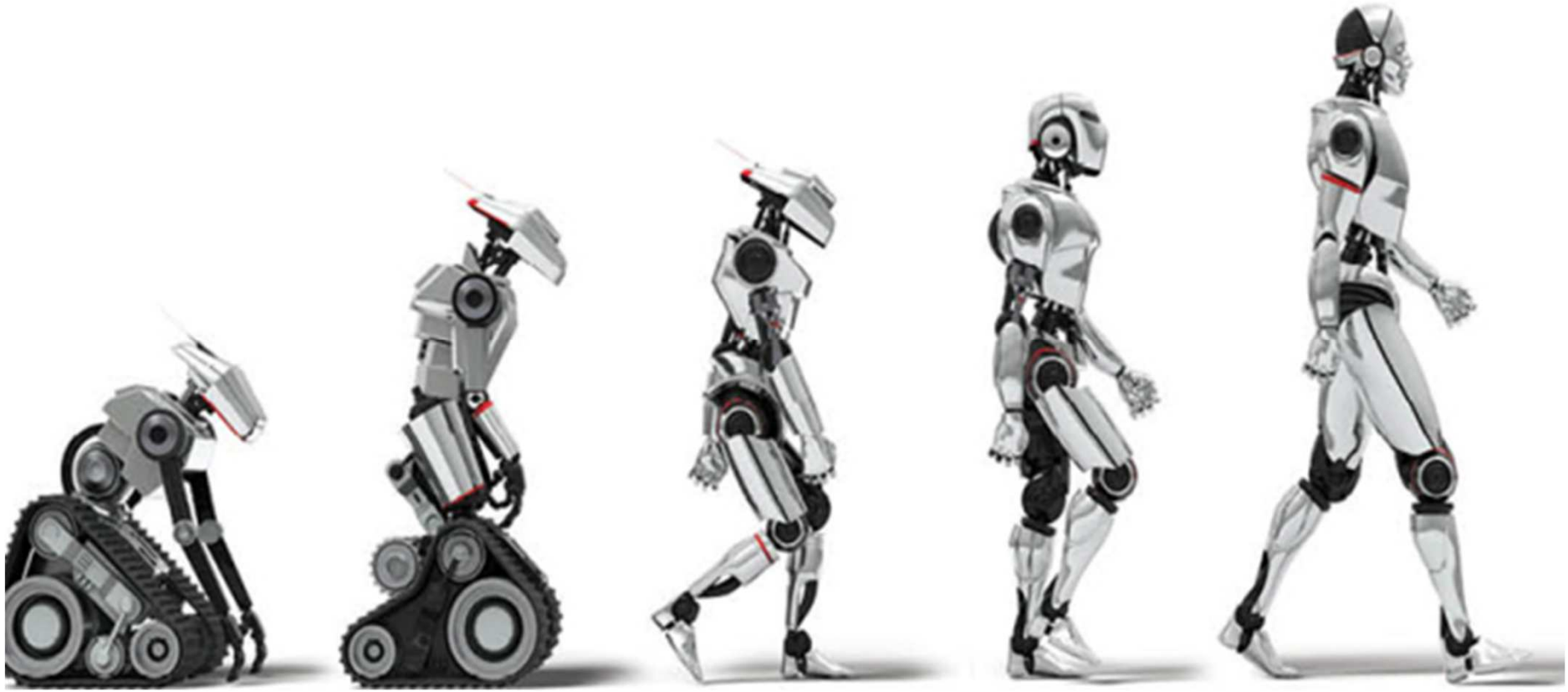
```
function TABLE-DRIVEN-AGENT(percept) returns an action
    persistent:  percepts, a sequence, initially empty
                 table, a table of actions, indexed by percept sequences, initially fully specified
    append percept to the end of percepts
    action ← LOOKUP(percepts, table)
    return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

- To build a rational agent in this way, we must construct a table that contains the appropriate action for every possible percept sequence.
- This approach is not applied to most of the environments, such as chess or taxi driver agents, because it is not possible to store and manage all the table entries.

Agent Program Types

Agent Program Types

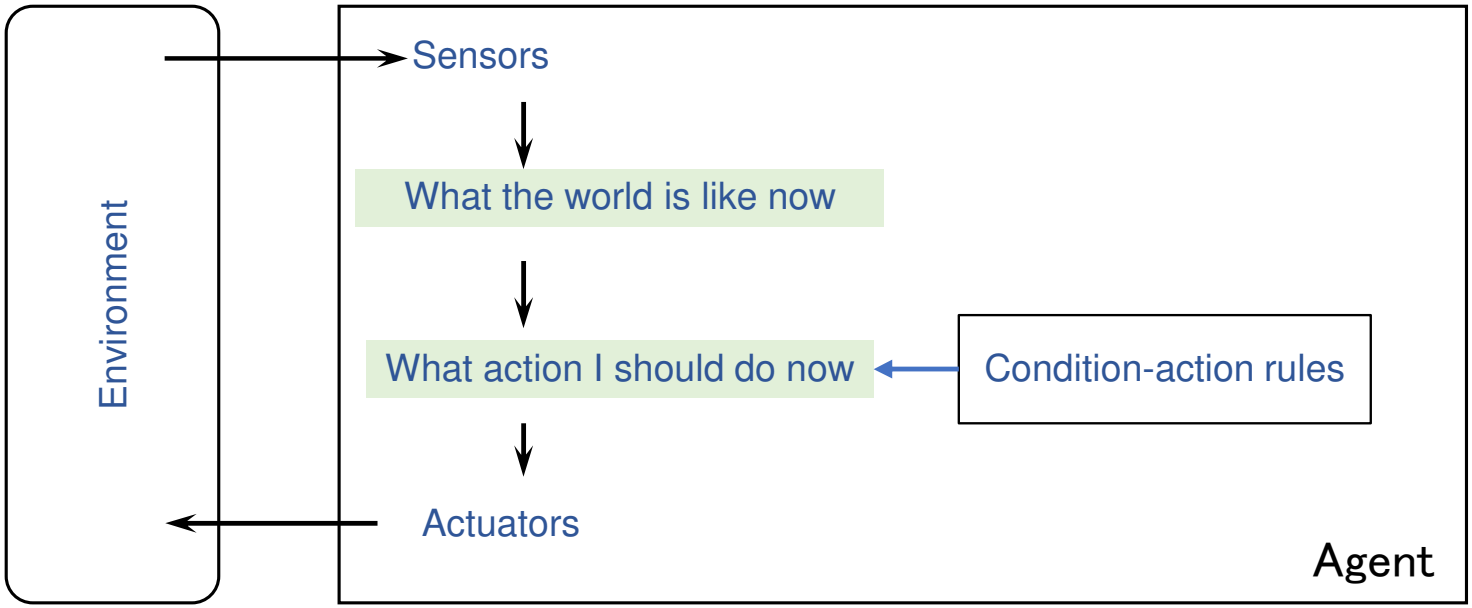
Retrieved on 2022.08.25 from
https://skilllx.com/types-of-agents-in-artificial-intelligence/



Simple reflex **Model-based reflex** **Goal-based** **Utility-based** **Learning agent**

Simple Reflex

Condition-action rules (if-then rules) make a direct link between current perception and action.



*The agent can be successful if the **environment is completely observable** and the correct decision can be made based only on **current perception**.*

Select actions on the basis of the current percept, ignoring the rest of the percept history

Simple reflex agent

- 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.

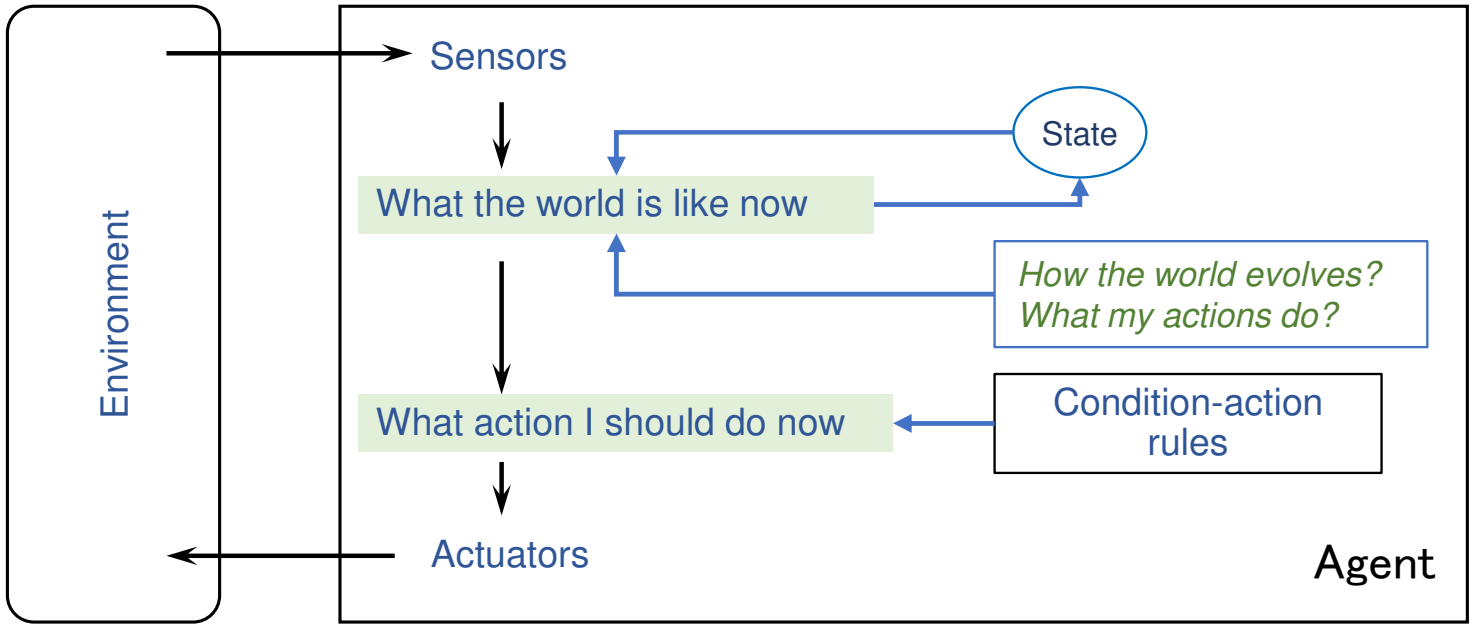
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    persistent: rules, a set of condition–action rules

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

- A **simple reflex agent** acts according to a rule whose condition matches the current state
  - In a car driver agent, if “the car in front is braking”, then this triggers a connection in the agent program to “initiate braking.”
  - This is a condition–action rule: ***if <condition> then <action>***

Model-based Reflex

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.



The agent should maintain some sort of **internal state** that depends on the **percept history** and thereby reflects some of the **unobserved aspects** of the current state.

Disadvantage: little autonomy, no goal defined, and no chaining of rules
Environments: **deterministic** and not very complex

Model-based reflex agents

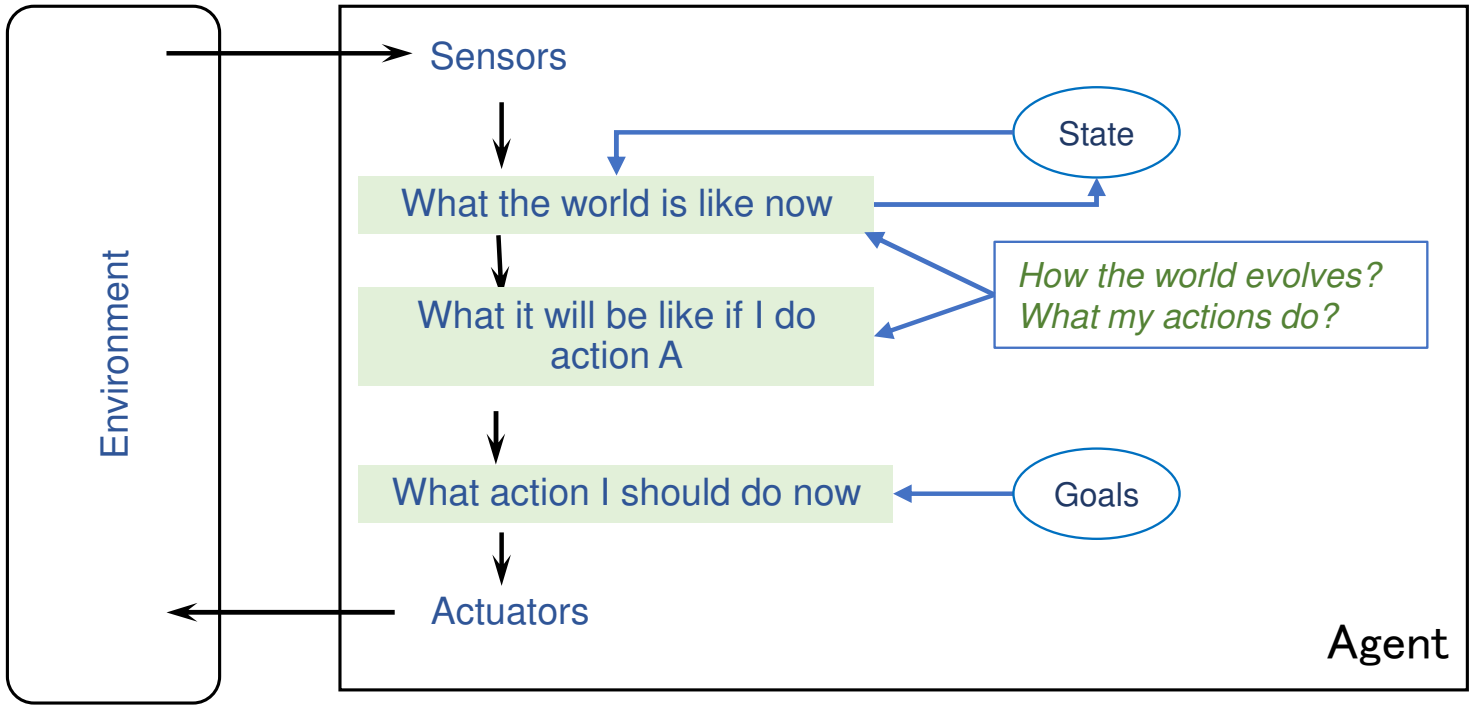
- It requires two kinds of knowledge to be encoded in the agent program
  - Information about (a) how the world changes over time by the effects of the agent's action and (b) how the world evolves independently of the agent. That is a **transition model** of the world.
  - Information about how the state of the world is reflected in the **Sensor model**. For example, identifying car braking when the camera gets wet and droplet-shaped objects appear in the image
- A model-based reflex **agent 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.

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
    persistent: state, the agent's current conception of the world state
    transition model, a description of how the next state depends on the current state and 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←UPDATE-STATE(state, action, percept, transition model, sensor model)
    rule←RULE-MATCH(state, rules)
    action←rule.ACTION
    return action
```

Goal-based Agents

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



As well as a current state description, the agent may need goal information that describes situations that are desirable — for example, for a car driver agent, what is the destination.

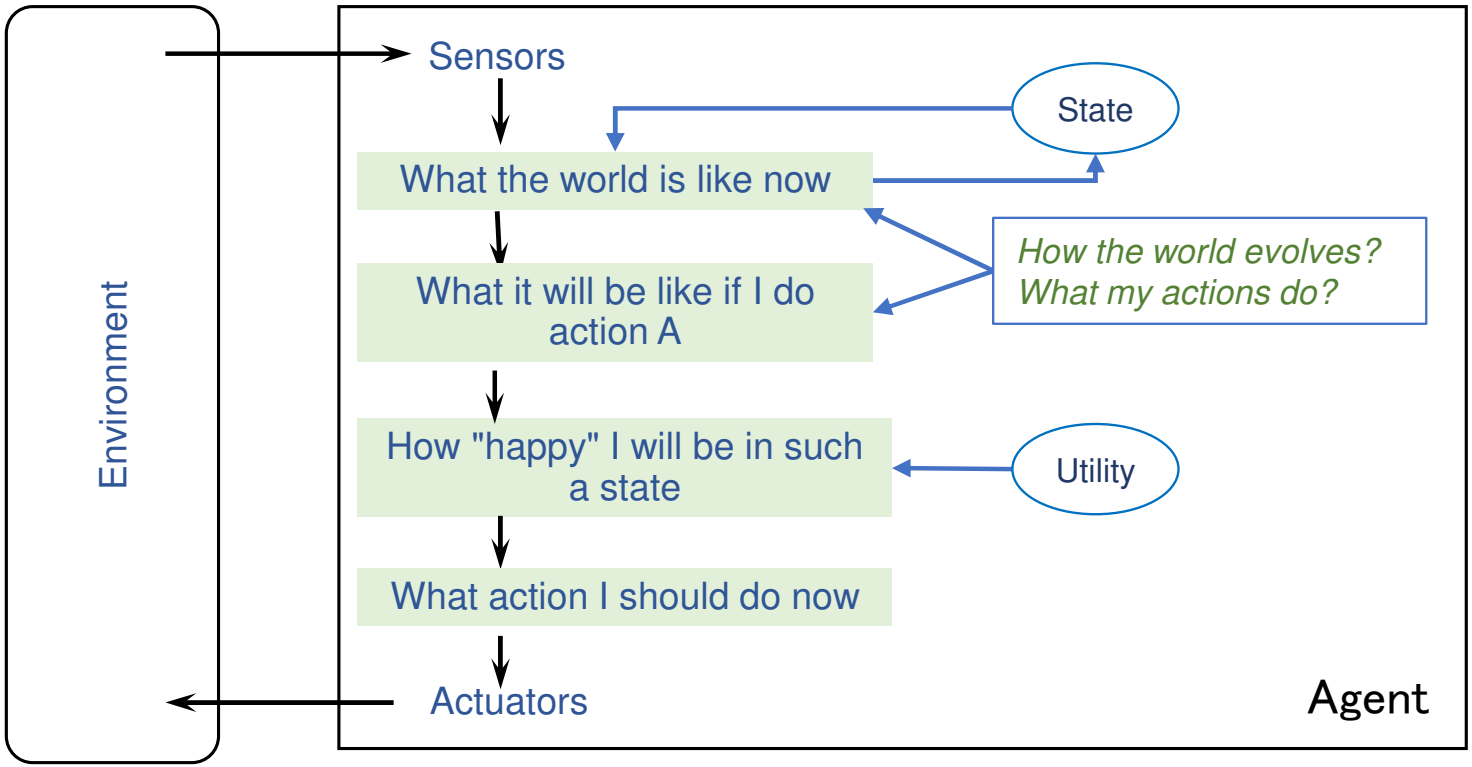
Goal-based agents (1)

- Goal-based action selection is
  - straightforward — for example, when goal satisfaction results immediately from a single action.
  - Complex — for example, when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal.
- Decision making of this kind is different from the condition–action rules,
  - It involves consideration of the future — both “What will happen if I do such-and-such?” and “Will that make me happy?”
- In the reflex agent designs, this information is not explicitly represented, because the built-in rules map directly from percepts to actions.
  - The reflex agent brakes when it sees brake lights, without having no idea why.
  - A goal-based agent brakes when it sees brake lights because that’s the only action that it predicts will achieve its goal of not hitting other cars.
- Goal-based agent appears less efficient, but it is more flexible because the **knowledge that supports its decisions is represented explicitly and can be modified**.
- Goal-based agents require **deterministic** task environments.

Utility-based Agent

A utility-based agent has to model and keep track of its environment, tasks that have involved a great deal of research on perception, representation, reasoning, and learning.

A utility-based agent chooses the action that **maximizes the expected utility** of the outcomes



A utility-based agent must model and **keep track of its environment**. This task involve a great deal of research on perception, representation, reasoning, and learning.

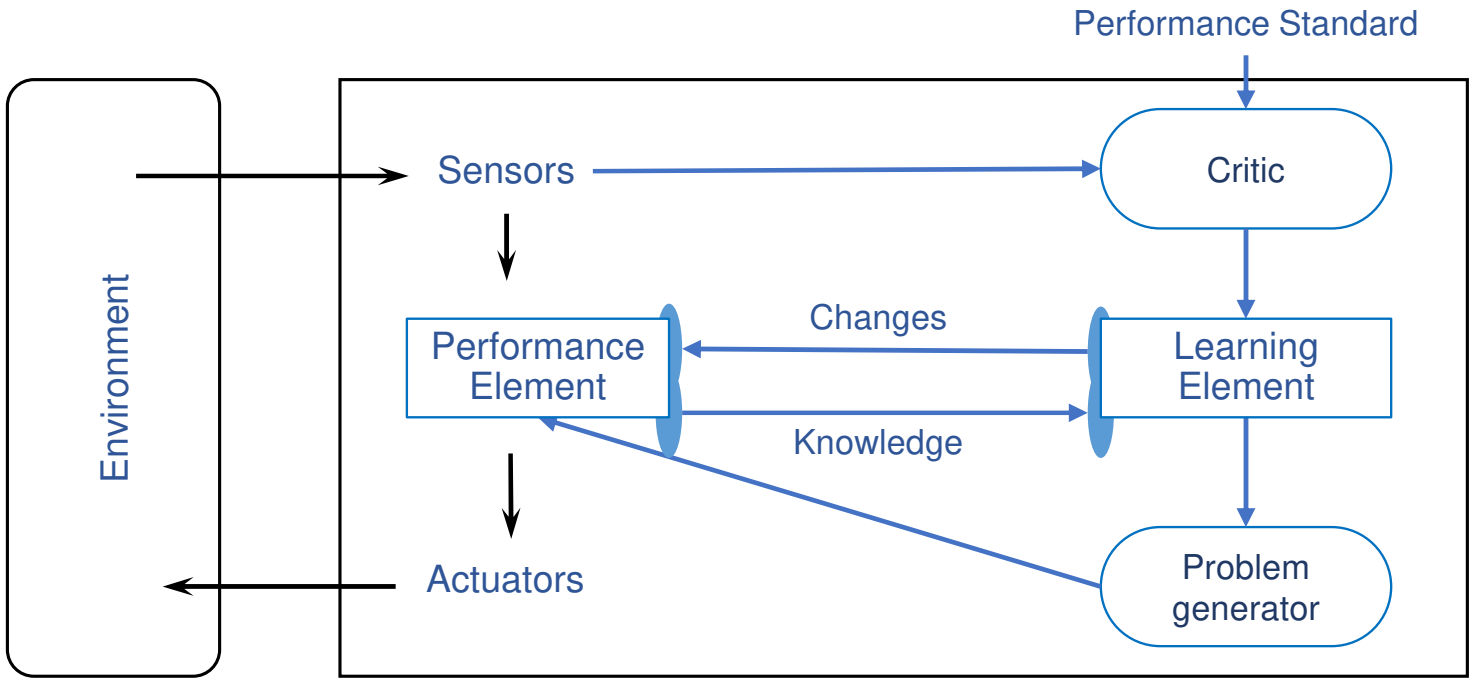
Utility-based agents (1)

- Goals are not enough to generate high-quality behavior in most environments. For example, many action sequences will get the taxi to its destination, but some are quicker, safer, or cheaper than others.
  - they just provide a crude binary distinction between “happy” and “unhappy” states.
  - A more general performance measure should allow a **comparison of different world states** according to exactly how "happy" they would make the agent. "Utility" is the term used instead.
- An agent’s utility function is essentially an **internalization of the performance measure**
  - The internal utility function and the external performance measure should agree
  - An agent that maximizes its utility will be rational according to the external performance measure.
- A **utility-based agent** has many advantages in terms of flexibility and learning. Furthermore, in two kinds of cases, goals are inadequate, but a utility-based agent can still make rational decisions
  - When there are conflicting goals – for example, speed and safety –, the utility function specifies the appropriate tradeoff.
  - When there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed against the importance of the goals
- Disadvantage: the agents **is not able to adapt to changing environments**.

Learning Agent

A learning agent can be divided into four conceptual components:

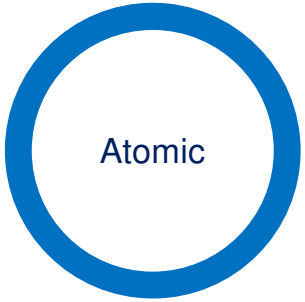
- the **learning element** is responsible for making improvements;
- the **performance element** is responsible for selecting external actions;
- the **critic** gives feedback on how the agent is doing and determines how the performance element should be modified to do better in the future;
- the **problem generator** is responsible for suggesting actions that will lead to new and informative experiences.



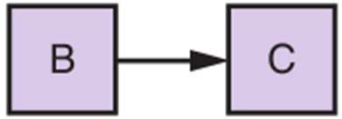
Learning Agent

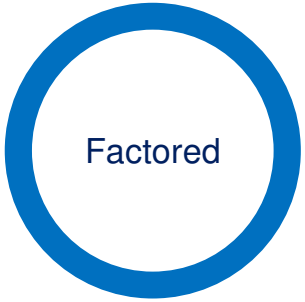
- The knowledge of the previous agent types has been determined by whoever developed it
  - A learning element uses feedback on how an agent is performing and determines how the performance element should be modified to perform better, in the future.
- The learning element can **make changes to any of the four "knowledge" components**.
  - Improving the model components of a model-based agent so that they conform better with reality is almost always a good idea, regardless of the external performance standard.
- The simplest cases involve learning directly from the percept sequence
  - Observation of pairs of successive states of the environment can allow the agent to learn "What my actions do" and "How the world evolves" in response to its actions.
  - For example, if the automated taxi exerts a certain braking pressure when driving on a wet road, then it will soon find out how much deceleration is achieved, and whether it skids off the road.
- The **problem generator** might identify certain parts of the model that need **improvement**
  - Suggest experiments, such as trying out the brakes on different road surfaces under different conditions.
  - Information from the external standard is needed when trying to learn a reflex component or a utility function.

States & States Transitions

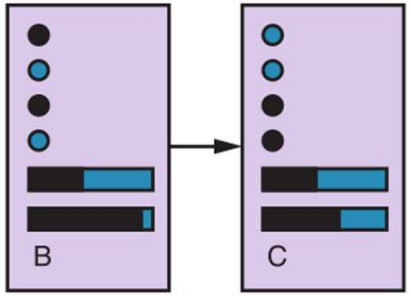


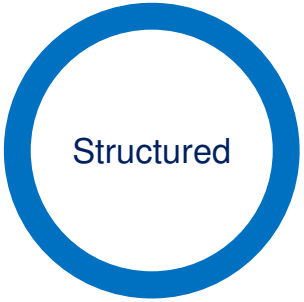
Each state is indivisible and and behaves as a black box with no internal structure



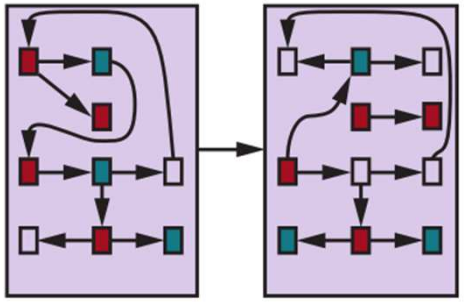


Each state is divided into a set of variables or attributes where each can have a value. Values can be Boolean, real-valued, or one of a fixed set of symbols





A state includes objects, each of which may have attributes of its own as well as relationships with other objects



Agents: development methodology

- Break the **problem** down into
  - perceptions, actions, goals, and
  - environment (and other agents)

- Break down the **type of knowledge** into
  - What are the relevant properties of the world?
  - How does the world evolve?
  - How to identify desirable states of the world?
  - How to interpret your perceptions?
  - What are the consequences of your actions in the world?
  - How to measure the success of your actions?
  - How to evaluate your own knowledge?

- Indicate the architecture and method of problem solving

Exercise – Room Temperature Control Agent (TempControl)

- Present a diagram and the pseudo-code for a simple reactive agent to control the temperature of a room.
- Suppose you have the perceptions
  - T1 and T2 corresponding to the room temperature and the outside temperature
- The available actions are:
  - ACH - turn on the hot air
  - ACC - turn on the cold air
  - NAC - Turn off the Air Conditioning
  - OW - open the windows
  - CW - close the windows.

- The room temperature is intended to be between 22 and 24 degrees.
  - Whenever it is possible, the windows should be used to control the temperature
  - Whenever the temperature is more than 2 degrees away from the desired range (T1<20 or T1>26), the windows should be closed, and the heater or cold air used to bring the temperature back up.

Exercise solution

- **Perceptions**:
  - T1 - Indoor Temperature
  - T2 - Outdoor Temperature.
- **Actions** (6) – see previous slide
- **Goal**: To keep the room temperature between 22 and 24 ºC
- **Perception Interpretation**
  - HOT+ = T1 > 26
  - HOT = T1 > 24 and T1<= 26
  - NORMAL = T1 >= 22 and T1<= 24
  - COLD = T1 >= 20 and T1< 22
  - COOL+ = T1 < 20
  - Windows = T2 < 24 and HOT or T2 > 22 and COLD

Condition-Action Rules:

- If NORMAL
  Then CW, NAC
- If (HOT or COLD) and Windows
  Then OW, NAC
- If HOT and not (Windows) or HOT+
  Then CW, ACC
- If COLD and not (Windows) or COLD+
  Then CW, ACH

Thank you!