Agents

LECTURE 1 - part II

Agents • Environment classification

Reactive Agent • Model-Based Agent • Planning Agent
Utility-based agent • Game Playing Agent • Learning Agent
Agent Programs and Architectures

UNSW SYDNEY

Agents situated in Environments

Agents • Environment classification

UNSW
SYDNEY

What is agent?

Artificial Intelligence is the synthesis and analysis of **computational agents** that **act intelligently**.

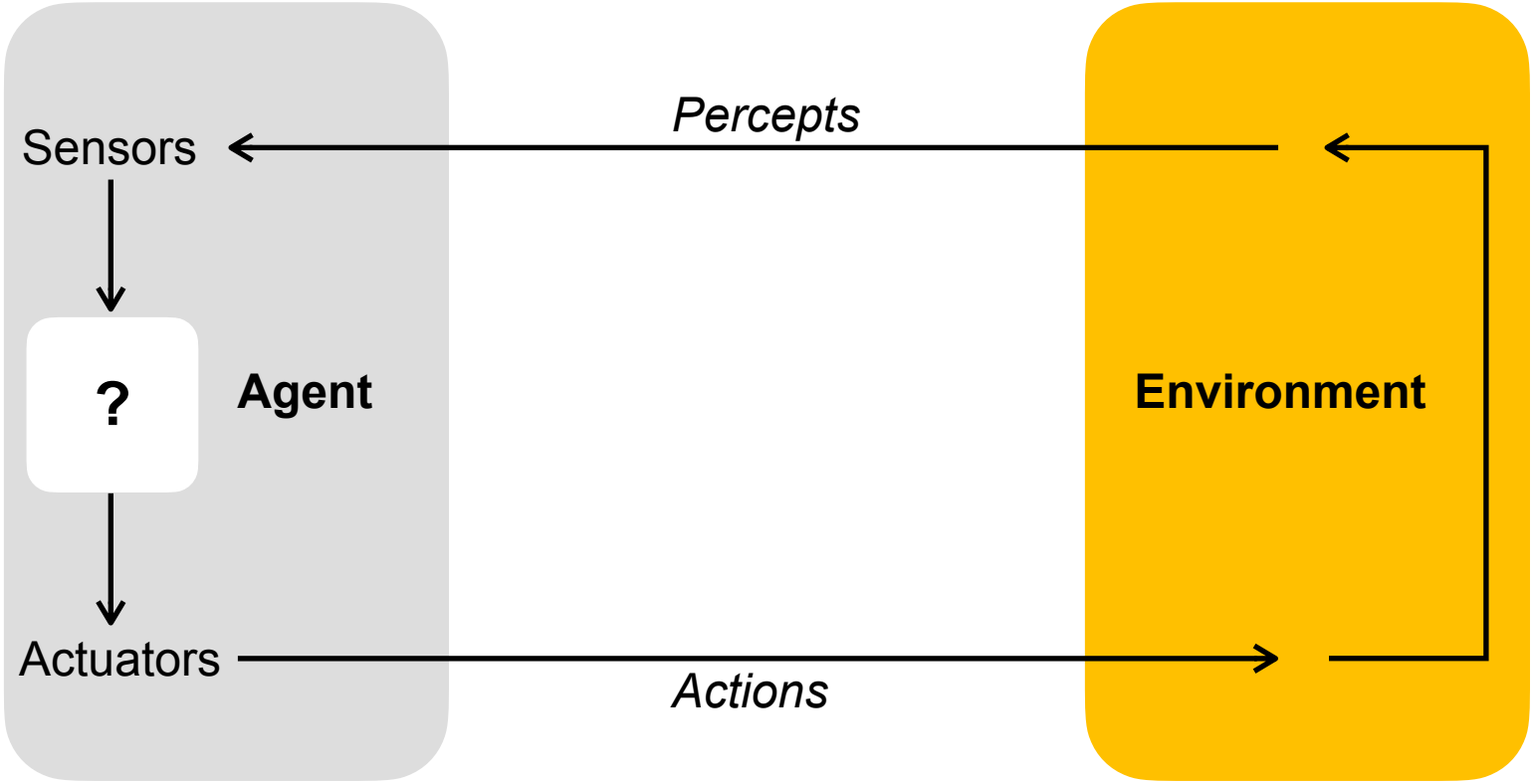
An **agent** is something that acts in an environment.

Agent as **actor:**

- Acts autonomously in the world to achieve goals
- Rational – may have beliefs, desires and intentions

Agent is an entity that perceives its **environment** through **sensors** and acts on its environment through **effectors/actuators**

Agent Model



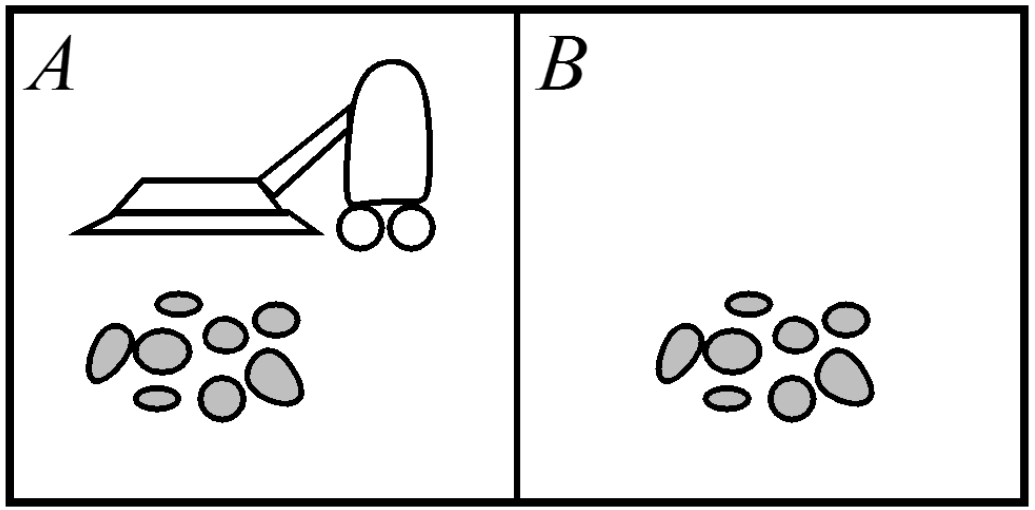
Agents as mappings

An agent can be seen as a mapping between percept sequences and actions.

f : Percept* → ***Action***

The agent program runs on a physical architecture to produce *f*

Example: Vacuum cleaner
- percepts location and and content *[A, Dirty]*
- perform actions *Left, Right, Suck, NoOp*



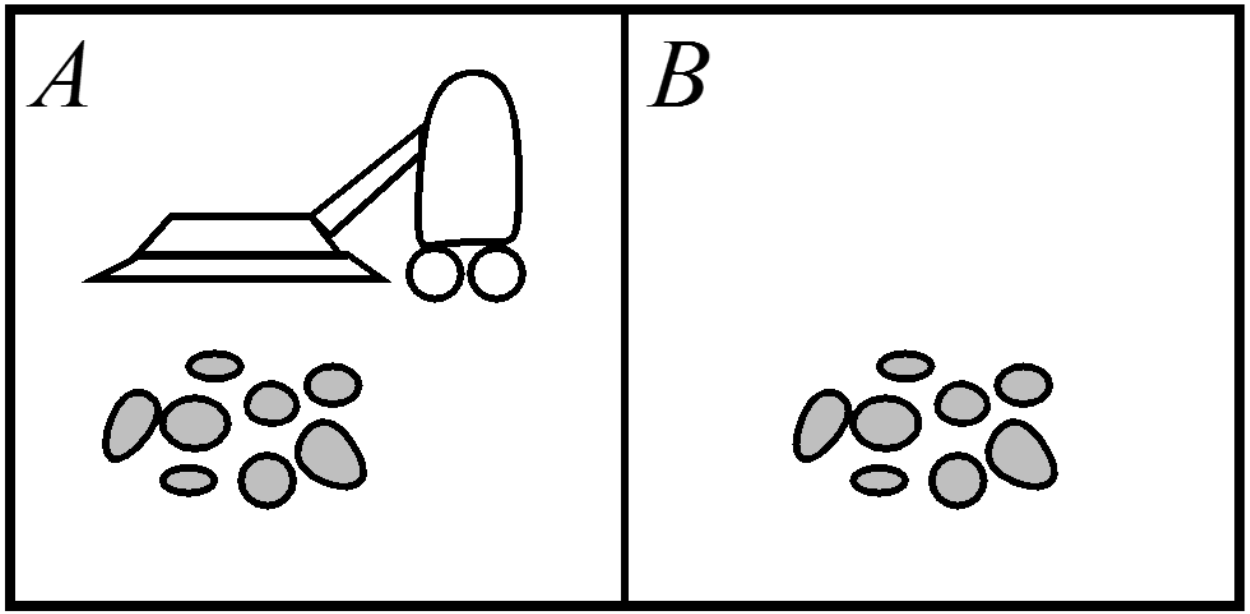
Vacuum cleaner world with 2 locations

The less an agents relies on its built-in knowledge, as opposed to the current percept sequence, the more autonomous it is.

Vacuum cleaner world with 2 locations

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

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
…	…



Agents in the environment

Before we design an agent program, we must have a good idea of the possible **percepts** and **actions**, what **goals** or **performance measure** the agent is supposed to achieve, and what sort of **environment** it will operate in.

- These come in a wide variety.

知觉

PAGE (Percepts, Actions, Goals, Environment) description.

Note: that the goals do not necessarily have to be represented within the agent; they simply describe the performance measure by which the agent design will be judged.

Examples of PAGE descriptions

Agent	Percepts	Actions	Goal	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Ask questions, run tests, prescribe treatments	Healthy patients, minimise cost	Patient, hospital
Chess playing system	Position of the chess pieces	Move a chess pieces	Score	Chess board, chess pieces
Movie recommendation system	Preferences, watched movies, feedback	Suggest/rank movies, gather feedback	Client's happiness, watched movies	Movie streaming service
Refinery controller	Temperature, pressure readings	Open and close valves, adjust temperatures	Maximise purity, safety	Refinery

Rational agents

The rationality of an agent depends on

- the performance measure defining the agent's degree of success
- the percept sequence, the sequence of all the things perceived by the agent
- the agent's knowledge of the environment
- the actions that the agent can perform

For each possible percept sequence, an **ideal rational agent** does whatever possible to maximise its performance, based on the percept sequence and its built-in knowledge.

Rationality

What is the right function?

Fixed performance measure evaluates any given **sequence of environment states**.

- one point per square cleaned up in time T?
- one point per clean square per time step, minus one per move?
- penalise for dirty squares?

Rational ≠ omniscient
percepts may not supply all relevant information

Rational ≠ clairvoyant
action outcomes may not be as expected

Rational ≠ successful

Rational ⇒ exploration, learning, autonomy

From "Doctor Who" episode "Smile" - episode 2, series 10

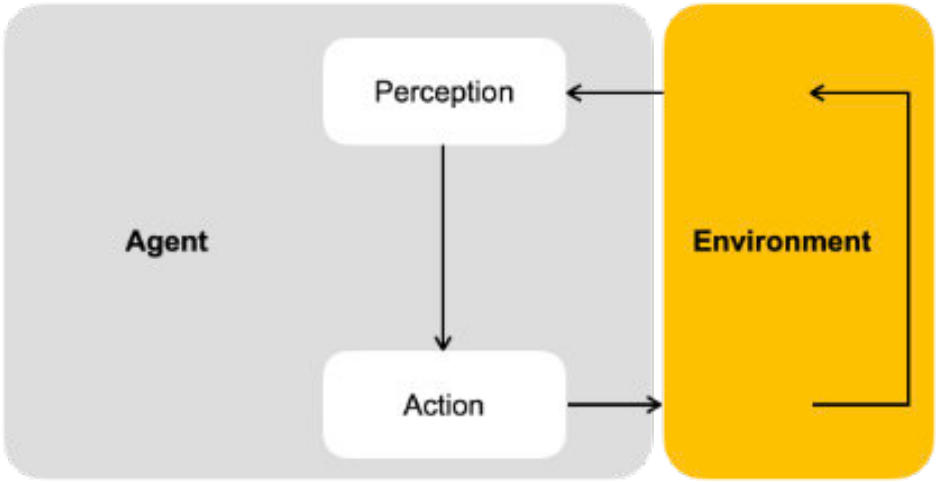
Specifying and classifying task

We want a unified framework that can be used to specify, characterise, compare, and contrast different AI tasks.

Tasks are the problems to which rational agents are the solutions.

To design a rational agent, we must specify the task environment:

- **P**erformance measure
- **E**nvironment
- **A**ctuators
- **S**ensors



PEAS model of an agent

- Performance measure
  How can we tell if an agent is doing a good or bad job?
  What are the qualities of performance that we would like to measure?
- Environment
  What are the components/attributes of the environment (which are relevant to the agent)?
- Actuators
  What are the outputs that enable action upon an environment?
- Sensors
  What are the inputs that sense and provide data for the agent?

Example: chess playing agent

Performance measure:

- +1 for a Win,
- +0.5 for a Draw
- 0 for a Loss.

Environment: board, pieces

Actuators: move piece to new square

Sensors: which piece is on which square

Example: “Smile” robot

Performance measure:

- +1 for increasing % of happy people,
- -1 for decreasing % of happy people,
- **- 100 increasing death rate**

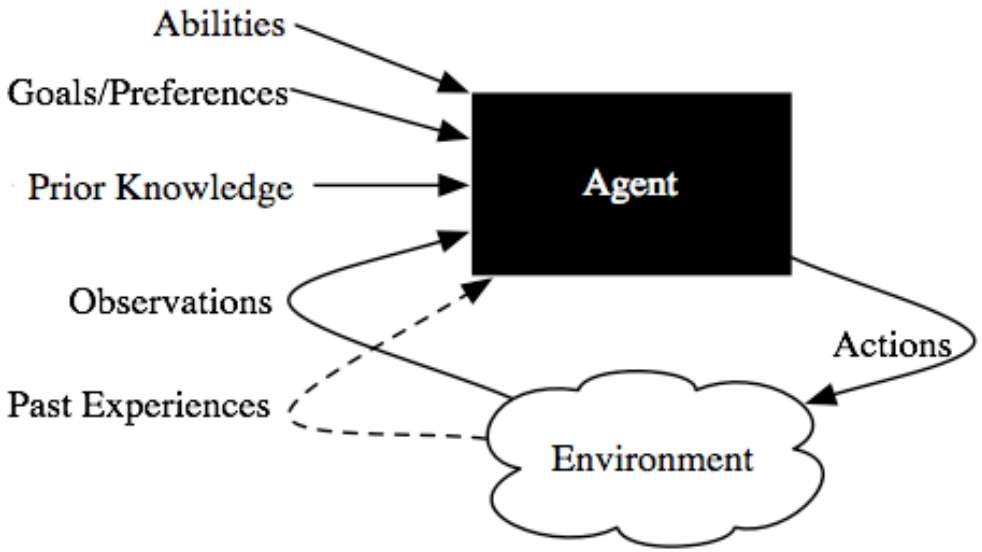
Environment: off-earth colony, people

Actuators: speakers, arms, legs, head, …

Sensors: cameras, microphones, …

Agent interacting with an environment

AI is about practical reasoning: reasoning in order to do something.



A coupling of perception, reasoning, and acting comprises an **agent**.

An agent acts in an **environment**.
An agent's environment may well include other agents.

An agent together with its environment is called a **world**.

Agent interacting with an environment

An Agent - a black box in terms of its inputs and outputs. At any time, what an agent does depends on:

- **prior knowledge** about the agent and the environment
- **history** of interaction with the environment, which is composed of
  - **stimuli** received from the current environment, which can include **observations** about the environment, as well as actions that the environment imposes on the agent and
  - **past experiences** of previous actions and stimuli, or other data, from which it can learn
- **goals** that it must try to achieve or **preferences** over states of the world
- **abilities**, the primitive actions the agent is capable of carrying out.

Example: “Smile” robot

abilities: movement, grippers, speech, facial expressions, and etc.

goals: entertain, heal, social skills, and etc.

prior knowledge: how to detect mood, personal data

stimuli: vision, speech recognition, gesture recognition, face recognition.

past experiences: effect from actions, history of person’s mood

Classifying environment types

模拟的

Simulated vs Situated/Embodied (有没有一个现实实体)

Static vs Dynamic

Discrete vs Continuous

Fully Observable vs Partially Observable

确定的 vs 随机的

Deterministic vs Stochastic

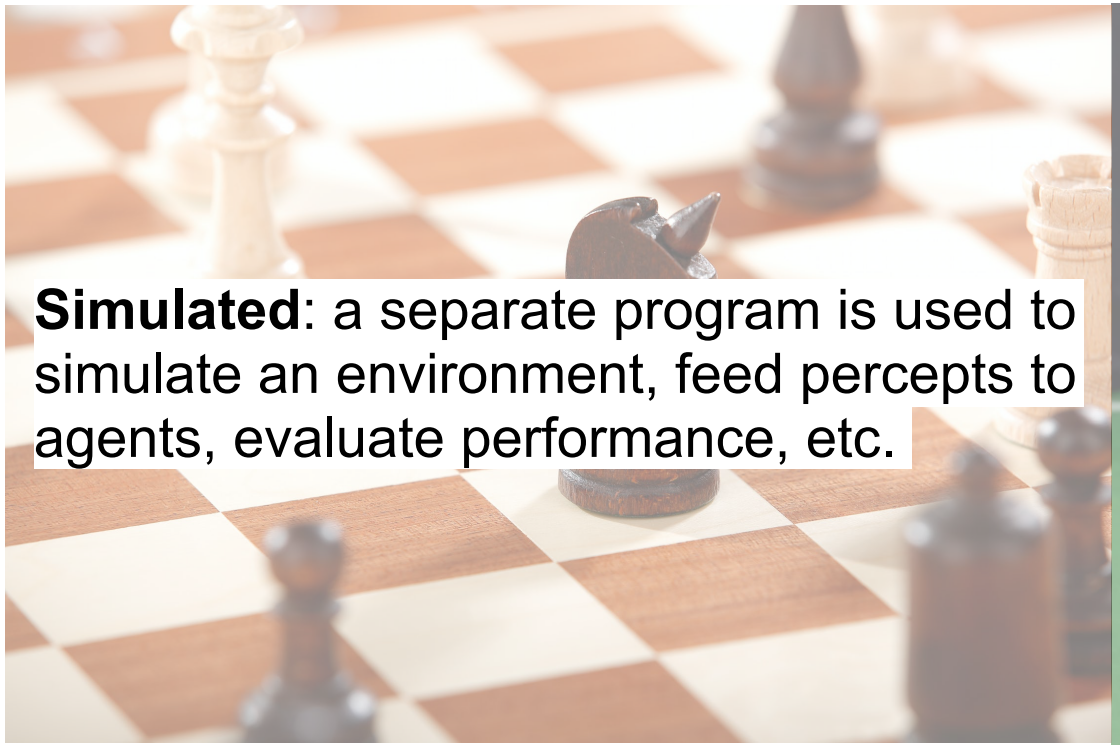
不定期的 vs 序列的

Episodic vs Sequential

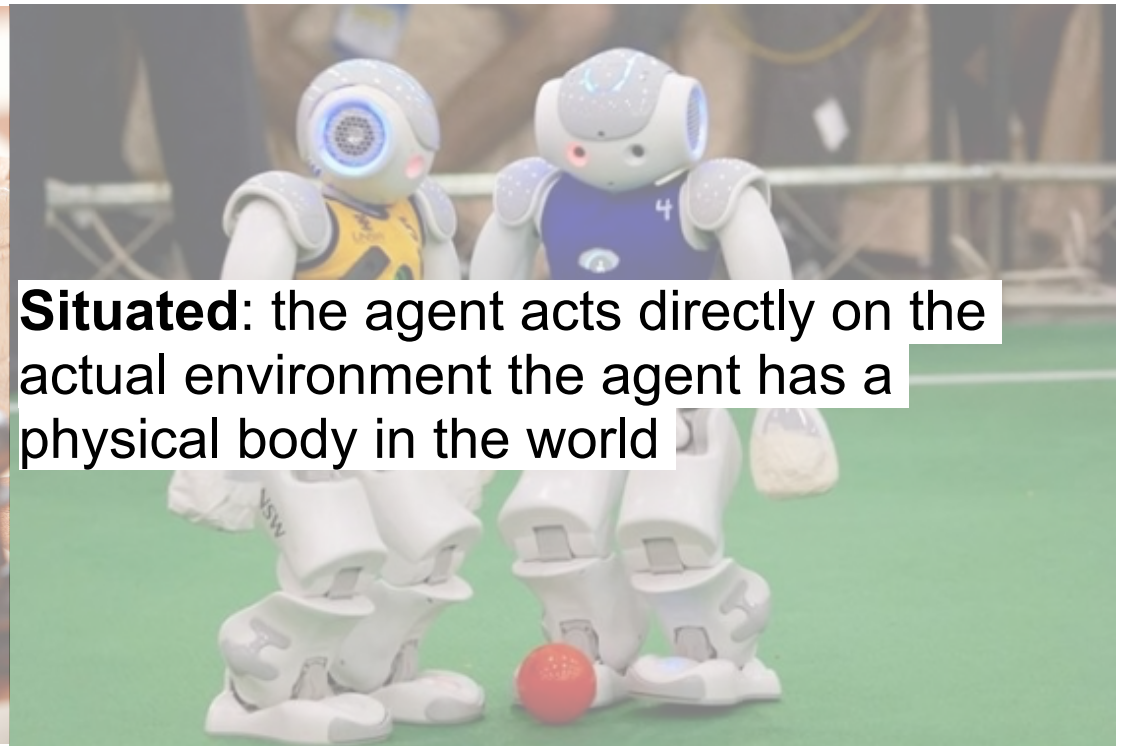
Known vs Unknown

Single-Agent vs Multi-Agent

Classifying environment types



Simulated: a separate program is used to simulate an environment, feed percepts to agents, evaluate performance, etc.



Situated: the agent acts directly on the actual environment the agent has a physical body in the world

Classifying environment types

Rodney Brooks 1991:

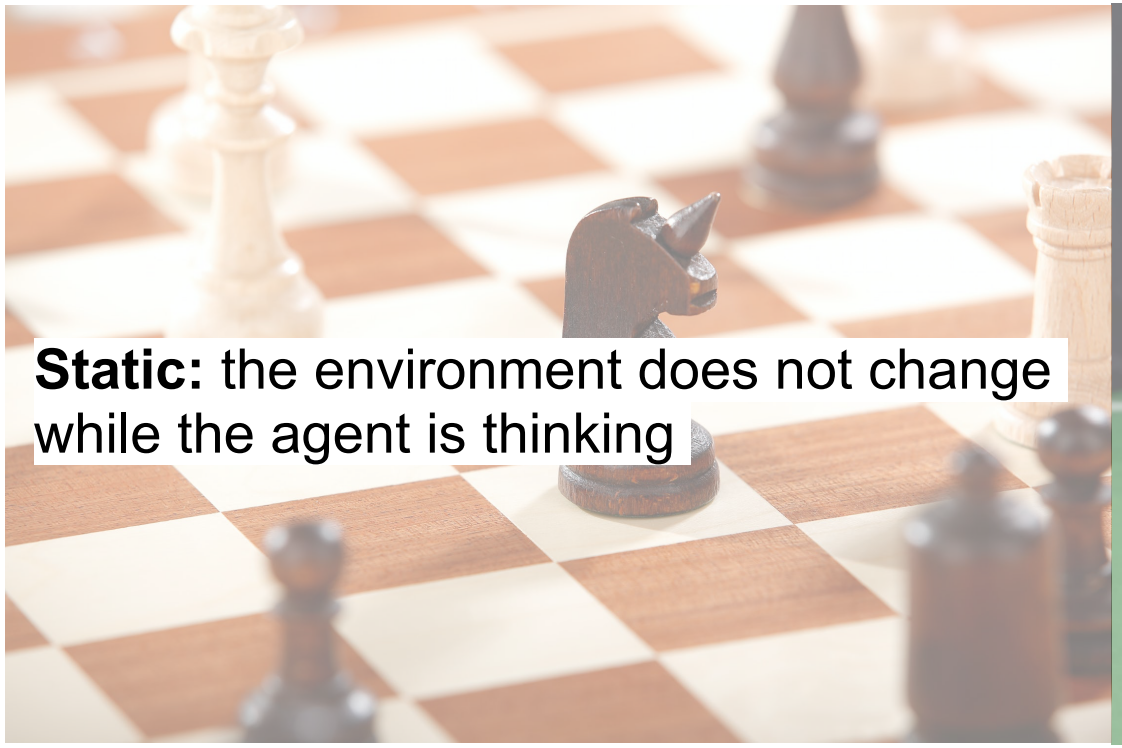
Situatedness: The robots are situated in the world – they do not deal with abstract descriptions, but with the “here” and “now” of the environment which directly influences the behaviour of the system.

Embodiment: The robots have bodies and experience the world directly – their actions are part of a dynamics with the world, and actions have immediate feedback on the robot’s own sensations.

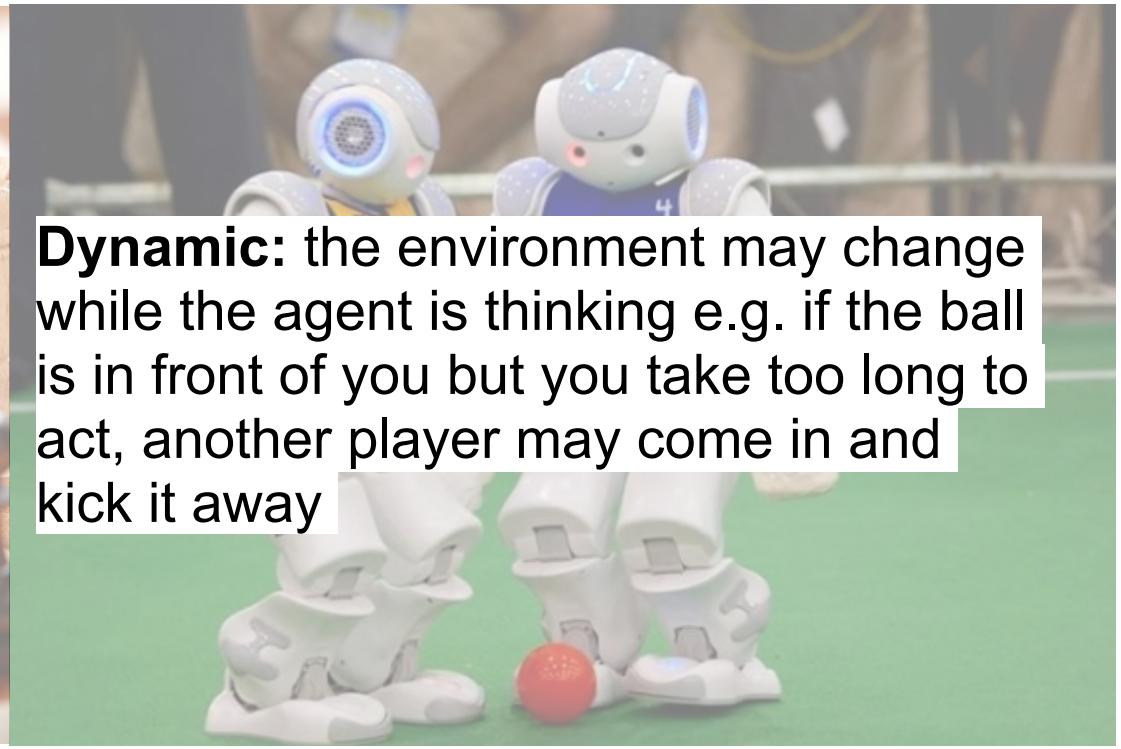
Situated but not Embodied: high frequency stock trading system

Embodied but not Situated: an industrial spray painting robot

Classifying environment types

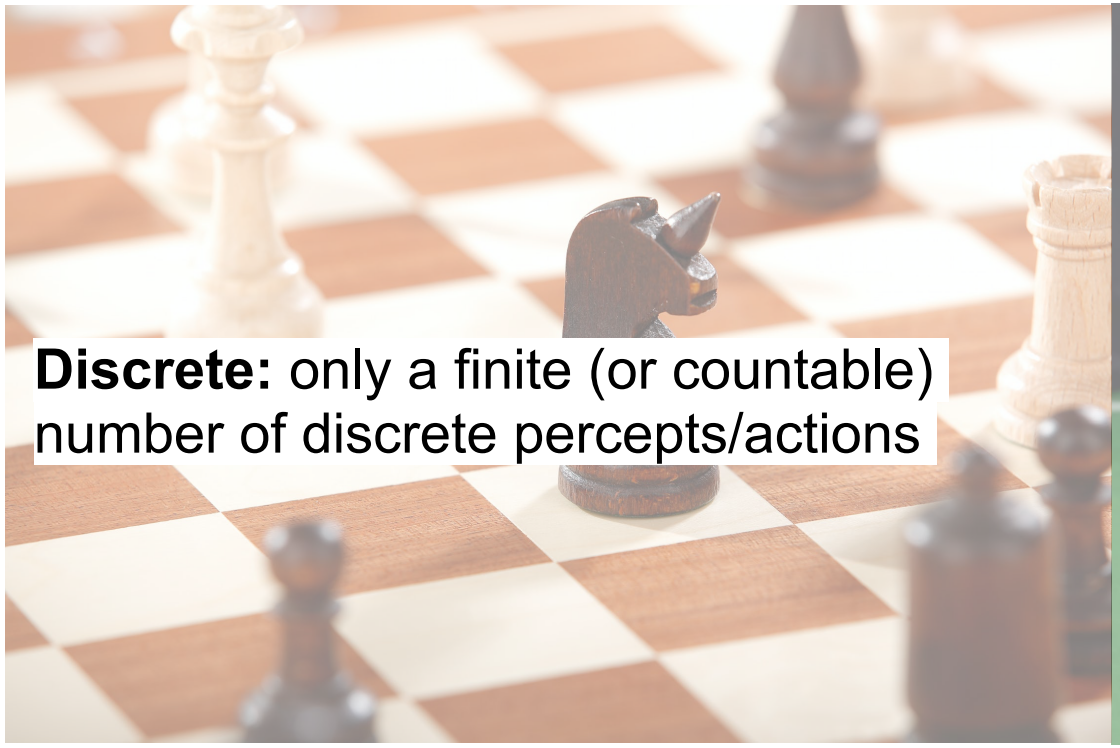


Static: the environment does not change while the agent is thinking

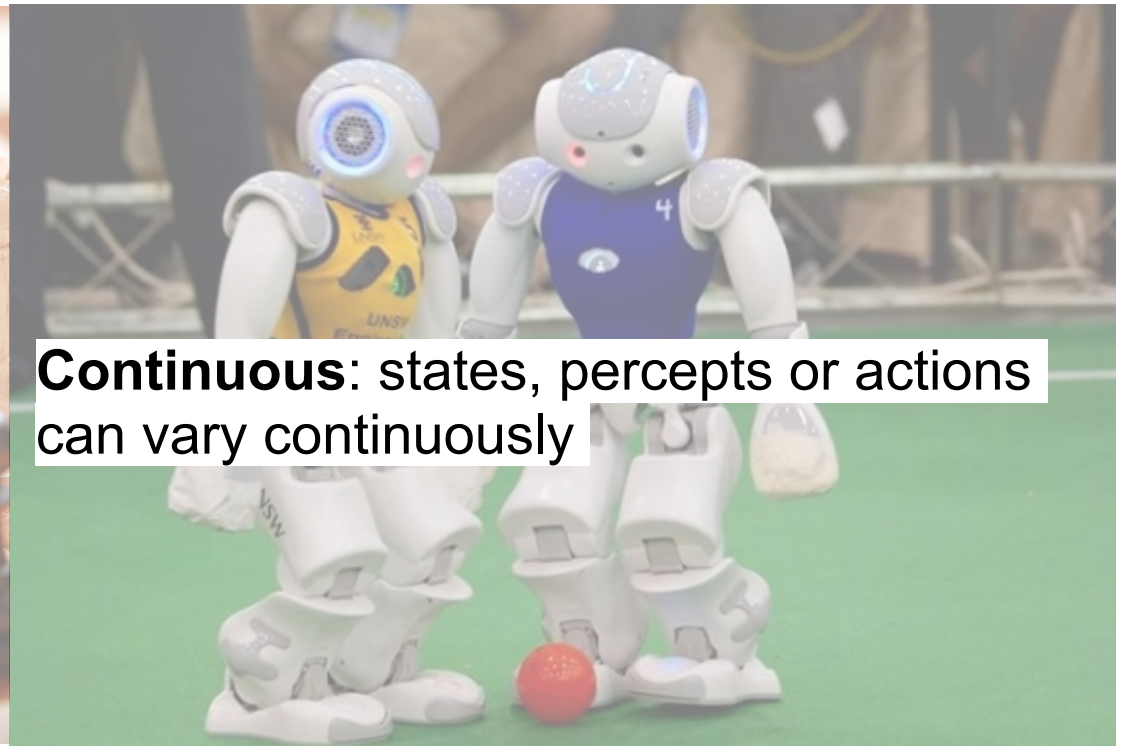


Dynamic: the environment may change while the agent is thinking e.g. if the ball is in front of you but you take too long to act, another player may come in and kick it away

Classifying environment types

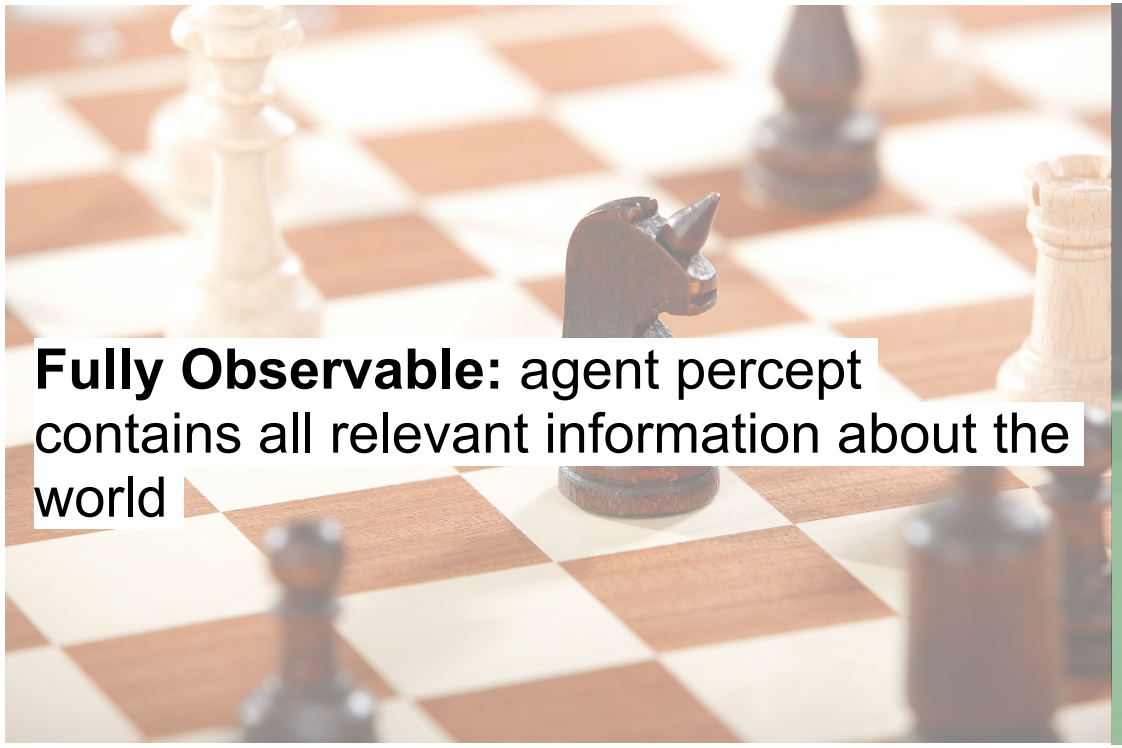


Discrete: only a finite (or countable) number of discrete percepts/actions

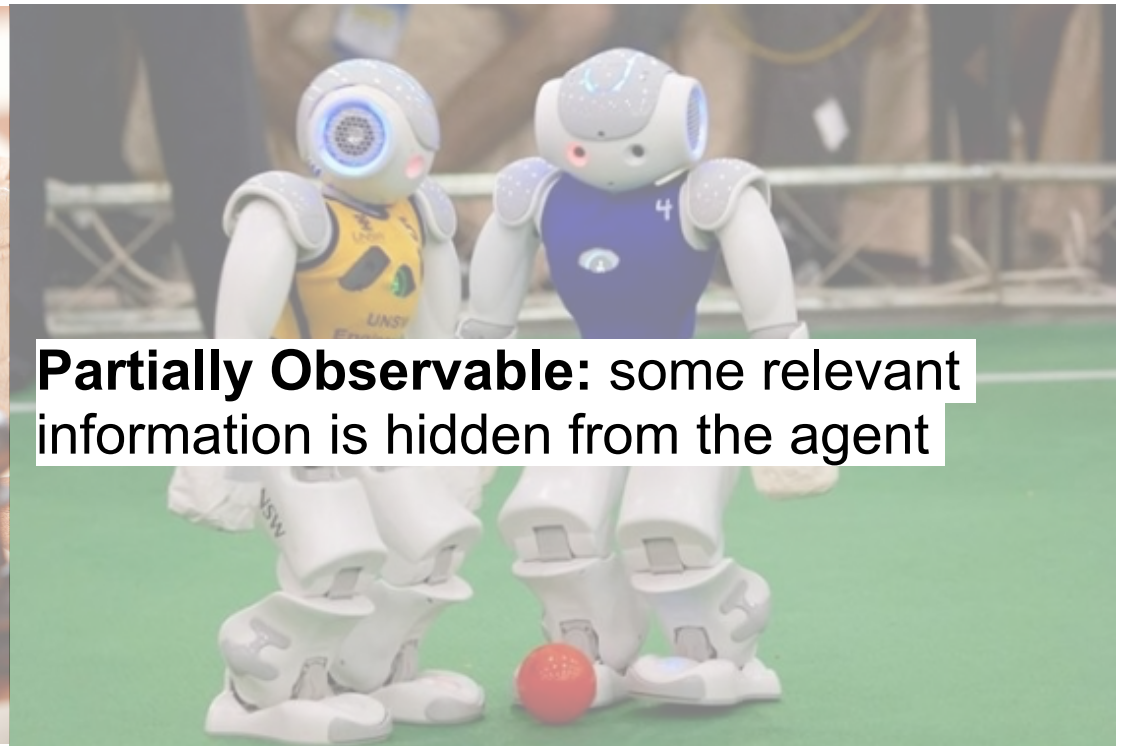


Continuous: states, percepts or actions can vary continuously

Classifying environment types

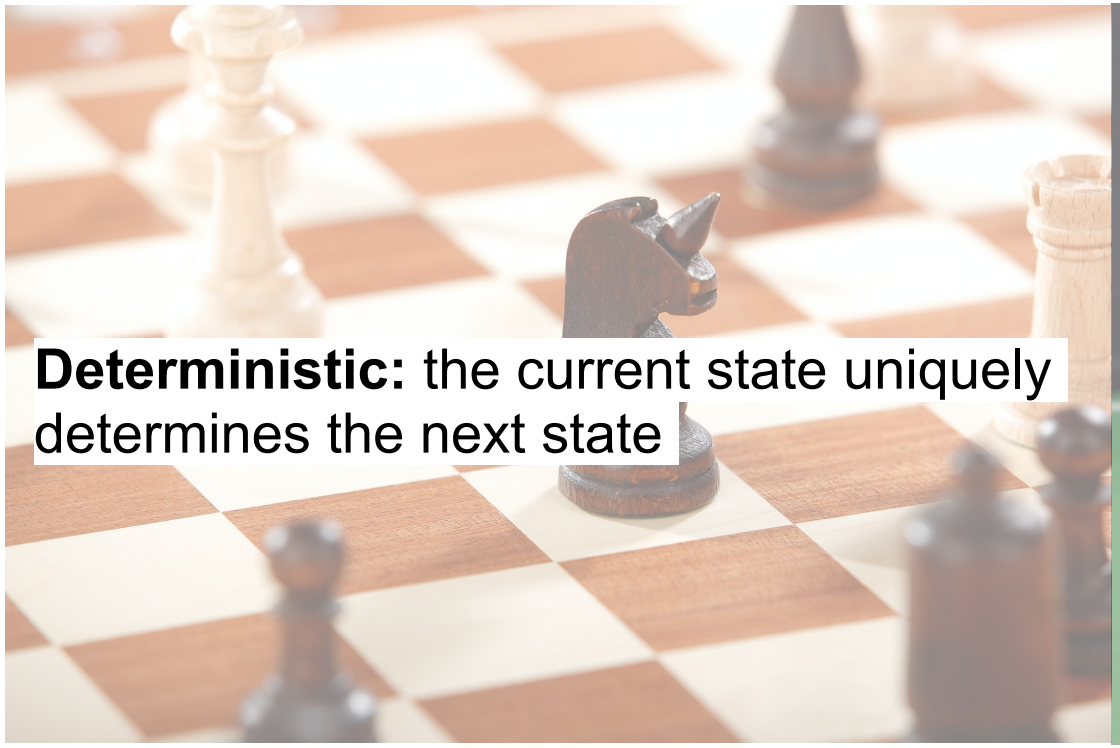


Fully Observable: agent percept contains all relevant information about the world

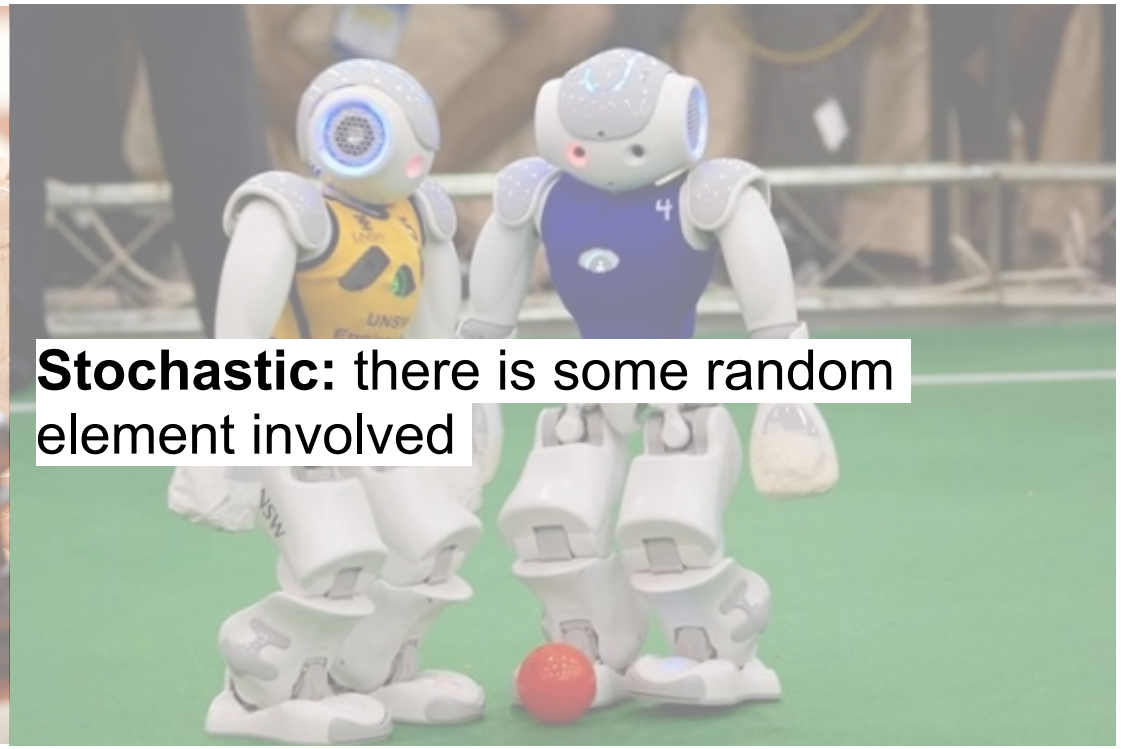


Partially Observable: some relevant information is hidden from the agent

Classifying environment types



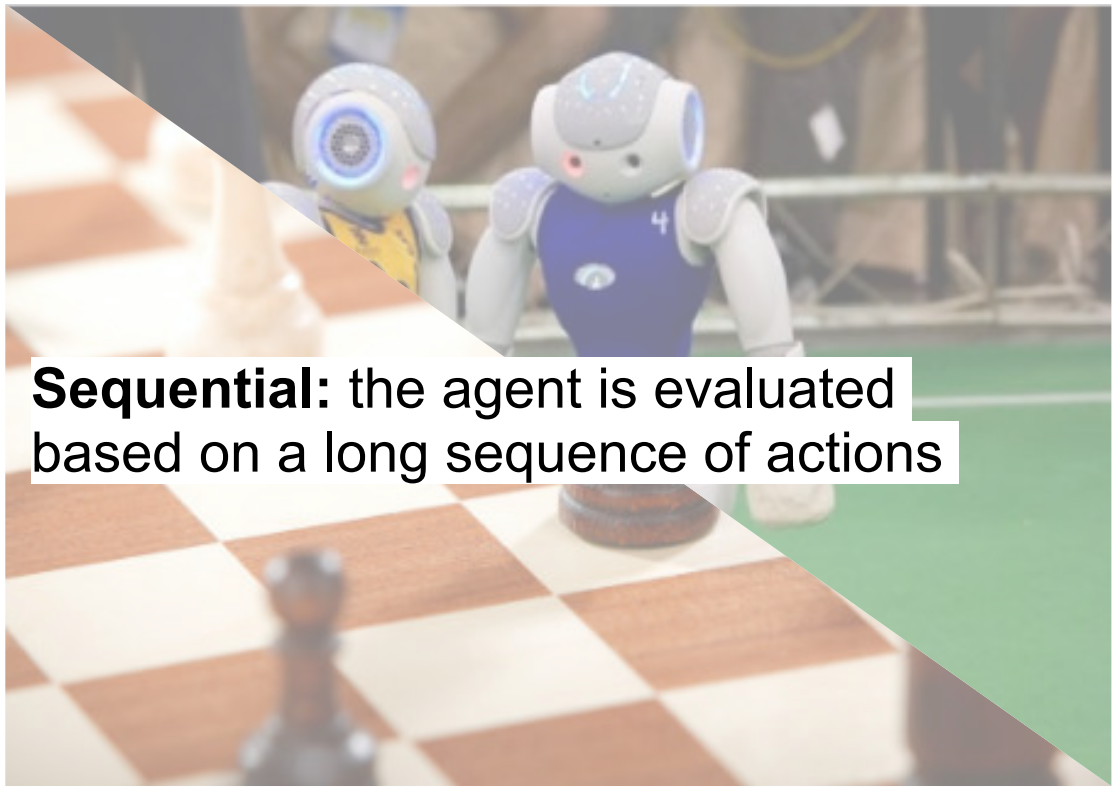
Deterministic: the current state uniquely determines the next state



Stochastic: there is some random element involved

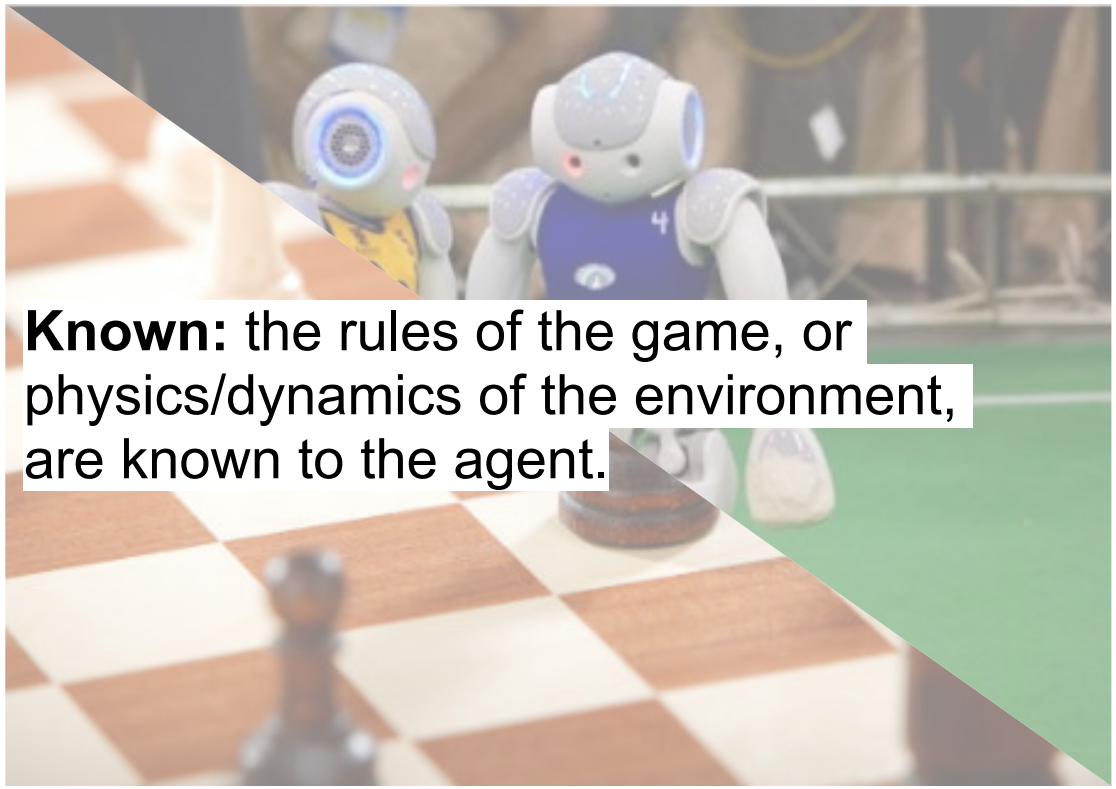
Classifying environment types

Episodic: every action by the agent is evaluated independently



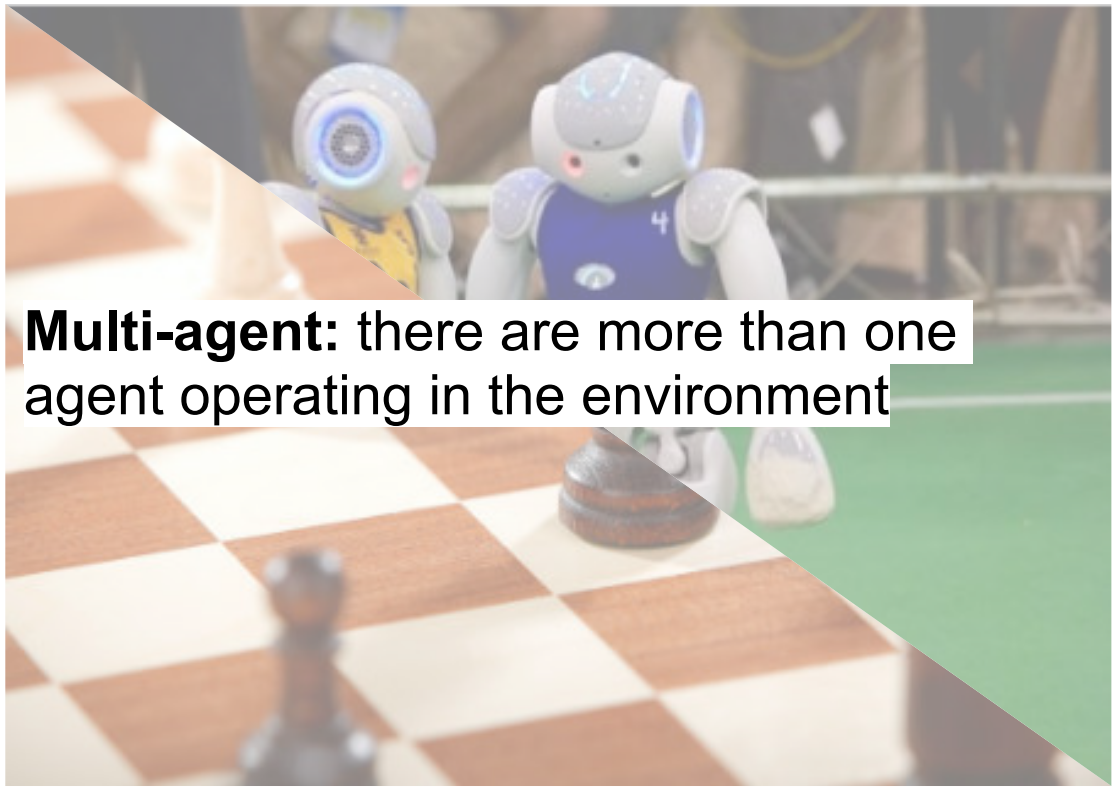
Classifying environment types

Unknown: the rules of the game, or physics/dynamics of the environment, are (partially) unknown to the agent.



Classifying environment types

Single-agent: there is only one agent operating in the environment



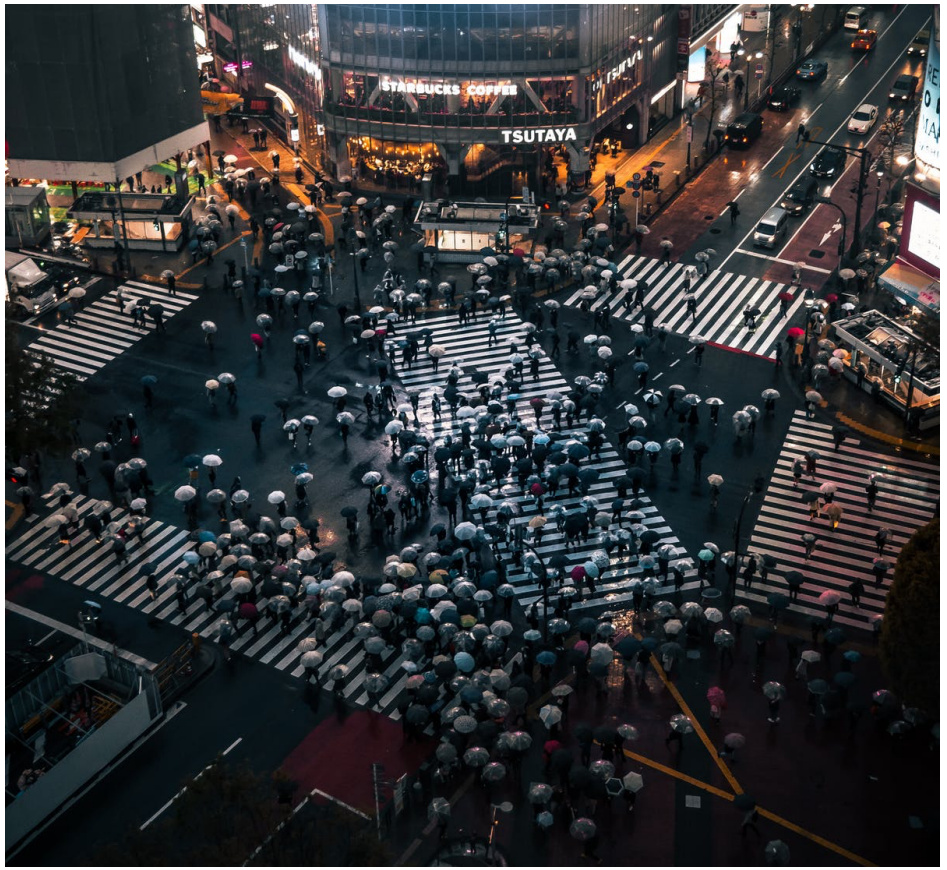
Multi-agent: there are more than one agent operating in the environment

Environments types

The environment type largely determines the agent design

The real world is (of course)

- partially observable,
- stochastic (instead of deterministic),
- sequential (instead of episodic),
- dynamic (instead of static),
- continuous (instead of discrete),
- multi-agent (instead of single-agent).



Summary

AI tasks or environments can be classified in terms of whether they are simulated, static, discrete, fully observable, deterministic, episodic, known, single- or multi- agent.

The environment type strongly influences the agent design (discussed in the next section)

- Capabilities those needed to survive in the environment
- Need not be “over-engineered” to handle more complexity
- Agent + Environment can be thought of as a coupled system

References:

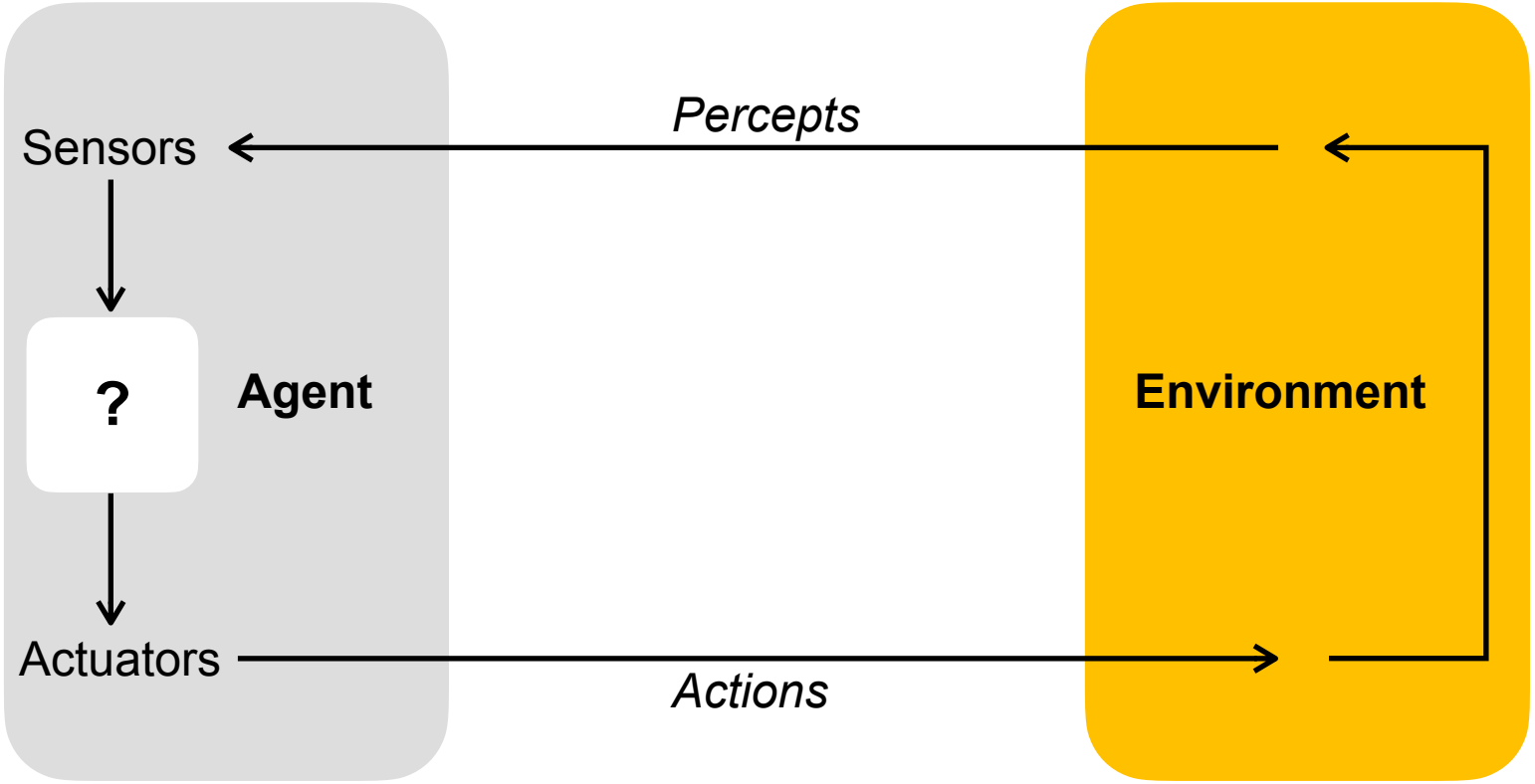
- Poole &Mackworth, Artificial Intelligence: Foundations of Computational Agents, Chapter 1
- Russell & Norvig, Artificial Intelligence: a Modern Approach, Chapter 2.

Agent types

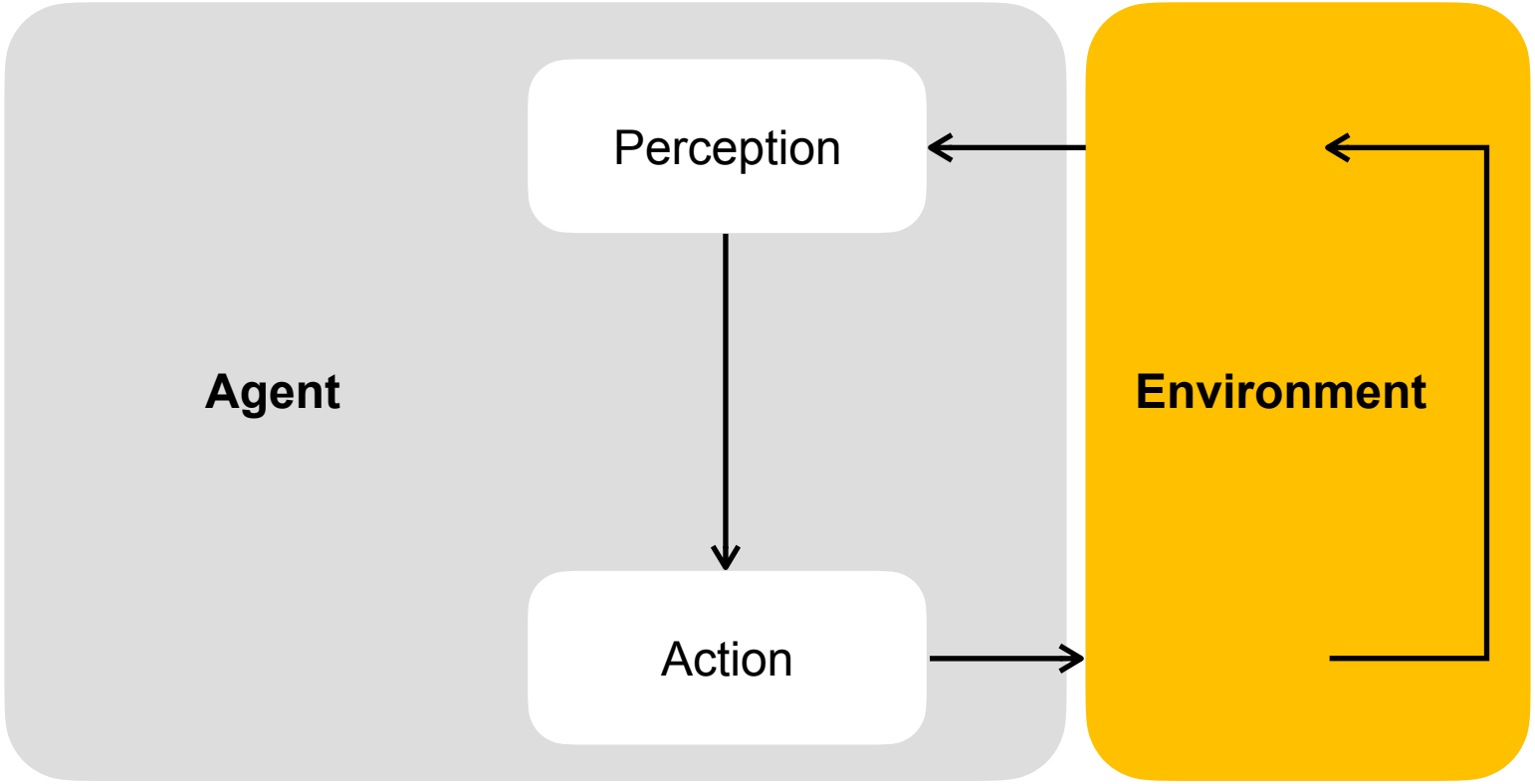
Reactive Agent • Model-Based Agent • Planning Agent
Utility-based agent • Game Playing Agent • Learning Agent
Agent Programs and Architectures

UNSW
SYDNEY

Agent Model



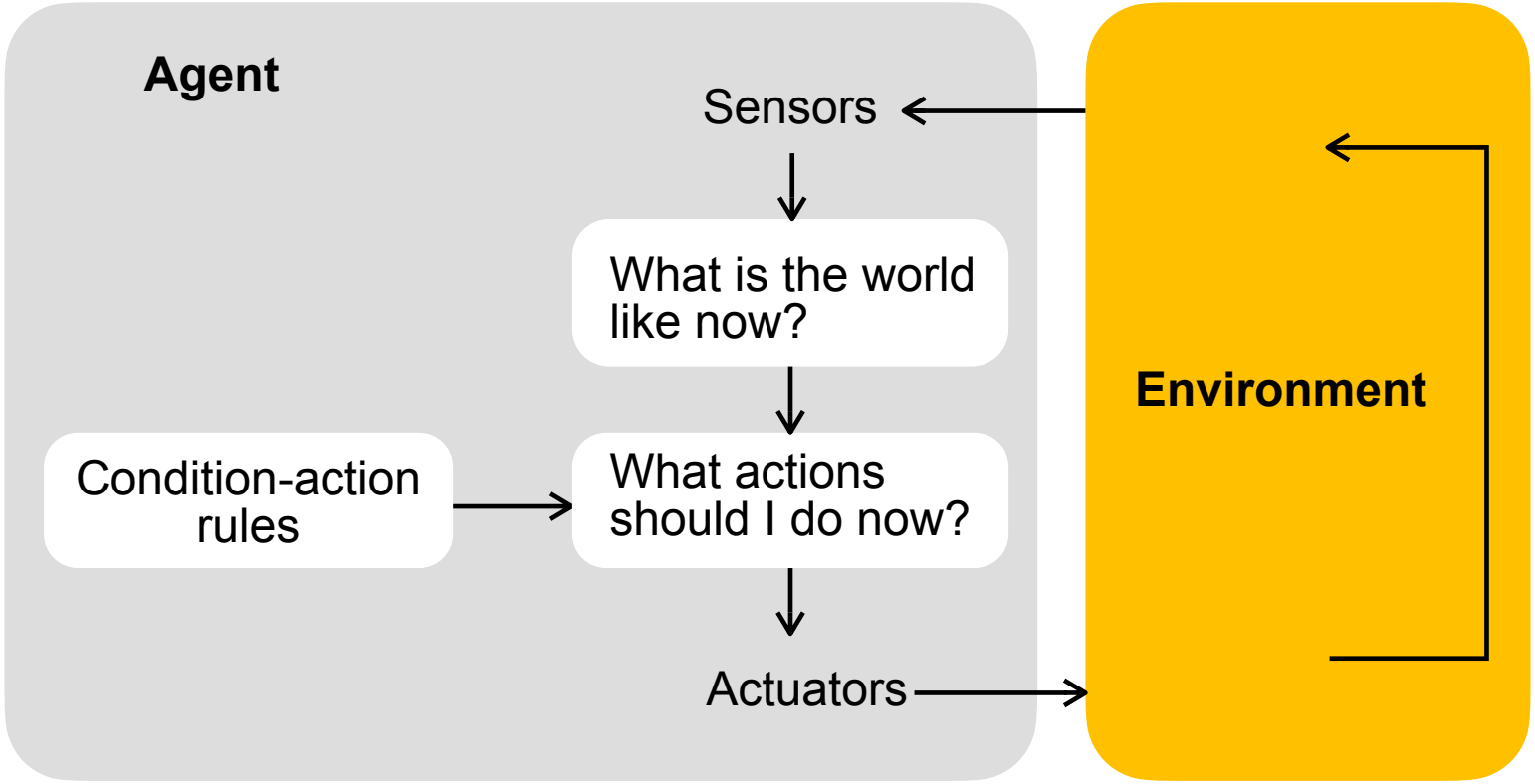
Reactive agent



Reactive agent

- Choose the next action based only on what they currently perceive, using a “policy” or set of rules which are simple to apply
- Sometimes called “simple reflex agents” – but they can do surprisingly sophisticated things!
  - Lego Mindstorms
  - Pool balancing
  - Automatic Climate Control
  - Swiss robots (https://www.youtube.com/watch?v=B0wM-eKSxhk)

Reflex (reactive) agent

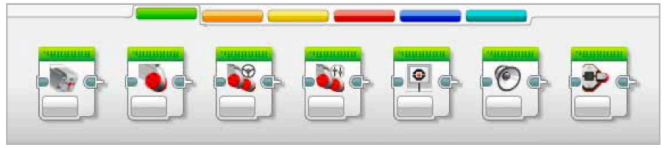


Reactive Agents - Lego Mindstorms

Action Blocks

(In order from left to right)

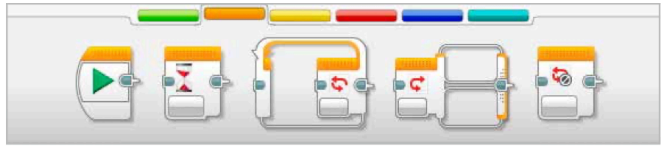
+ Medium Motor
+ Large Motor
+ Move Steering
+ Move Tank
+ Display
+ Sound
+ Brick Status Light



Flow Blocks

(In order from left to right)

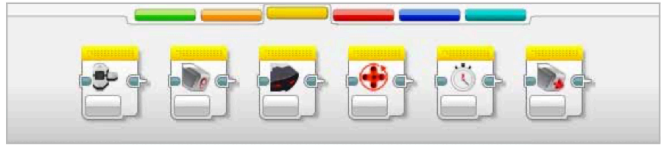
+ Start
+ Wait
+ Loop
+ Switch
+ Loop Interrupt



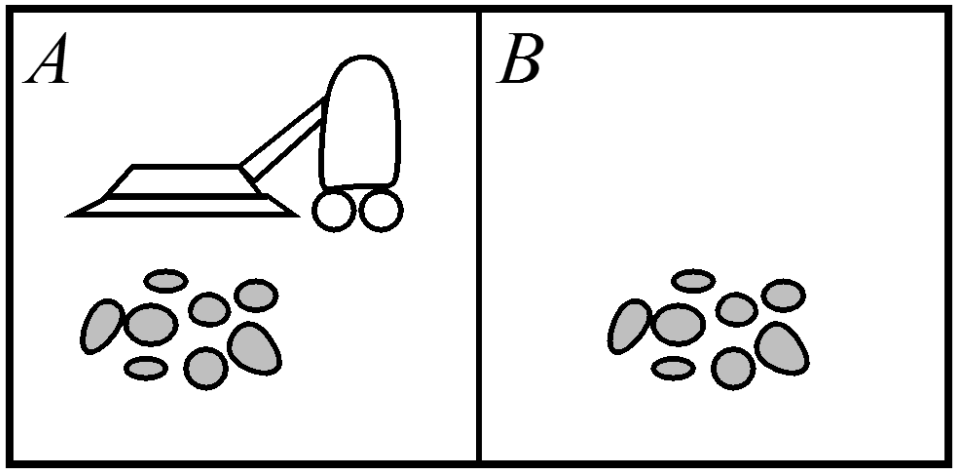
Sensor Blocks

(In order from left to right)

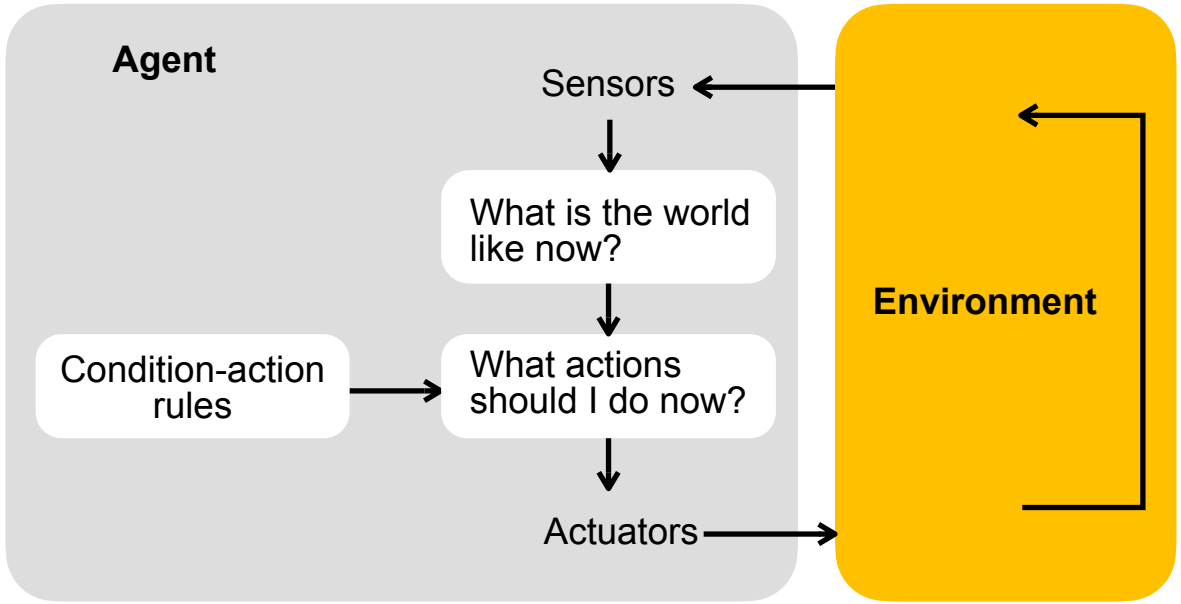
+ Brick Buttons
+ Color Sensor
+ Infrared Sensor
+ Motor Rotation
+ Timer
+ Touch Sensor



Vacuum-cleaner world

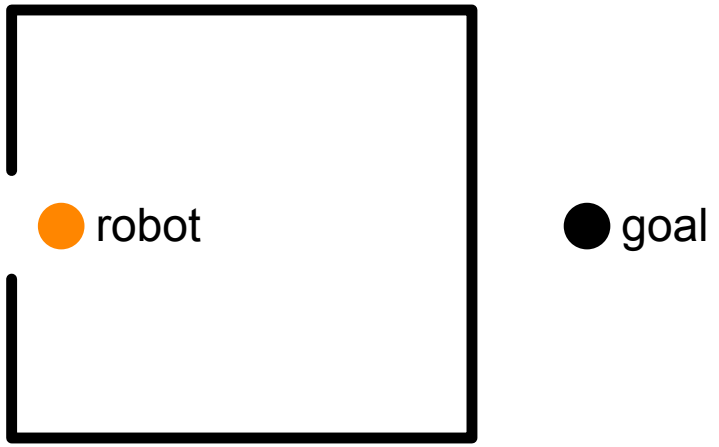


Percepts: location and contents
Actions: Left, Right, Suck, NoOp



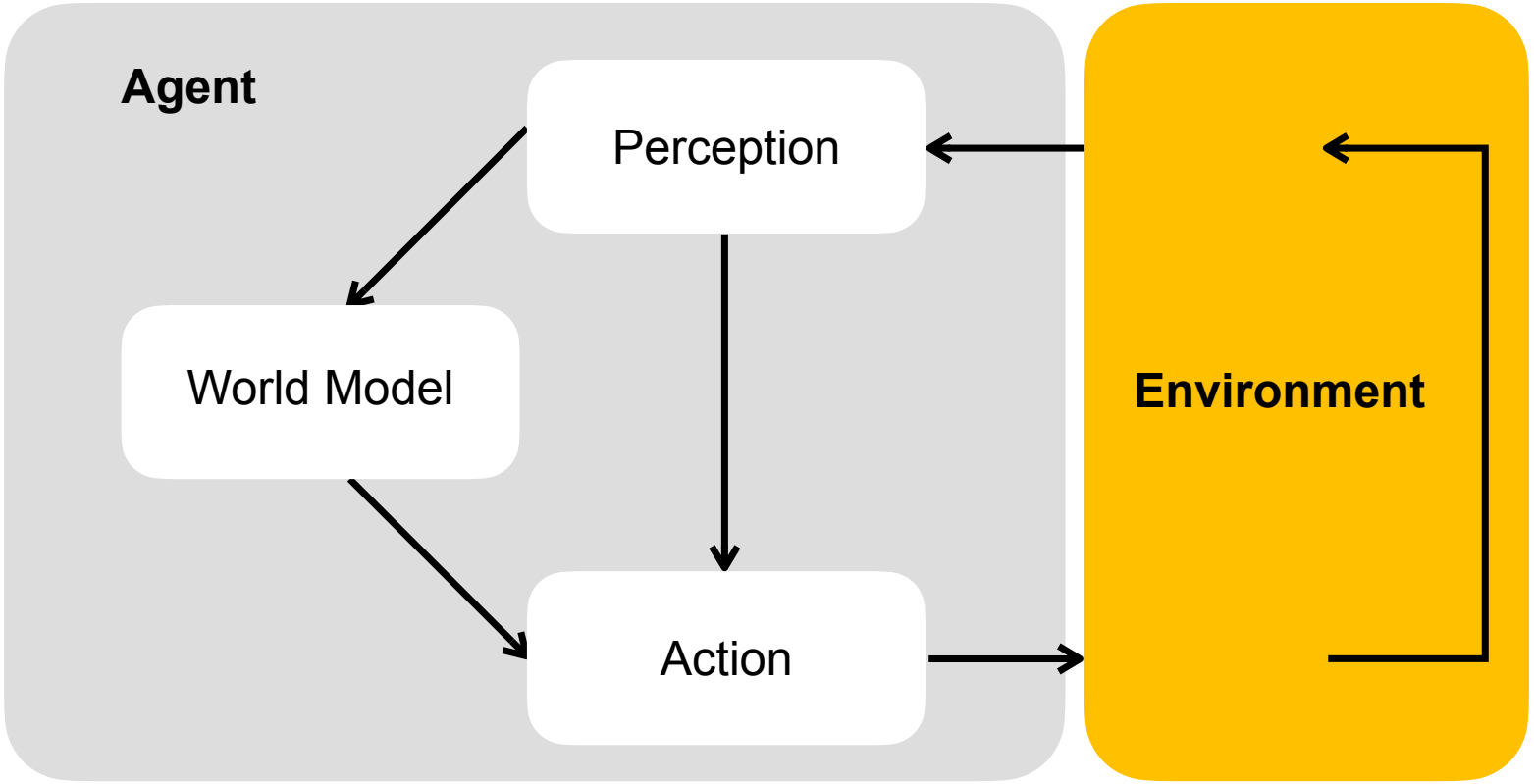
Limitations of Reactive Agents

- Reactive Agents have no memory or “state”
  - unable to base decision on previous observations
  - may repeat the same sequence of actions over and over



Escape from infinite loops is possible if the agent can randomise its actions.

Model-Based Agent

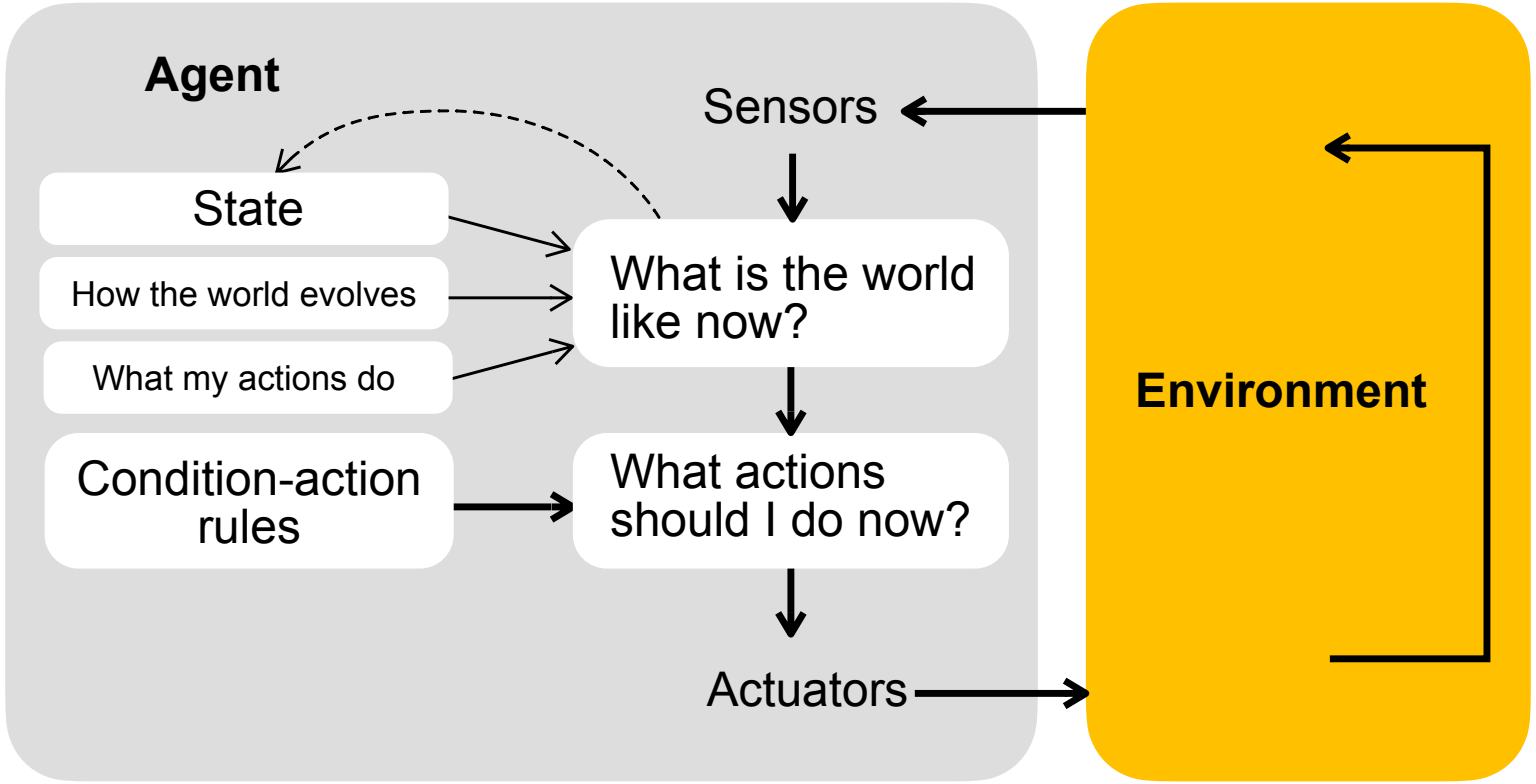


Model-Based Agent

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.
- 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.
- Knowledge about “how the world works”—is called a model of the world. An agent that uses such a model is called a model-based agent.

Model-Based Agent



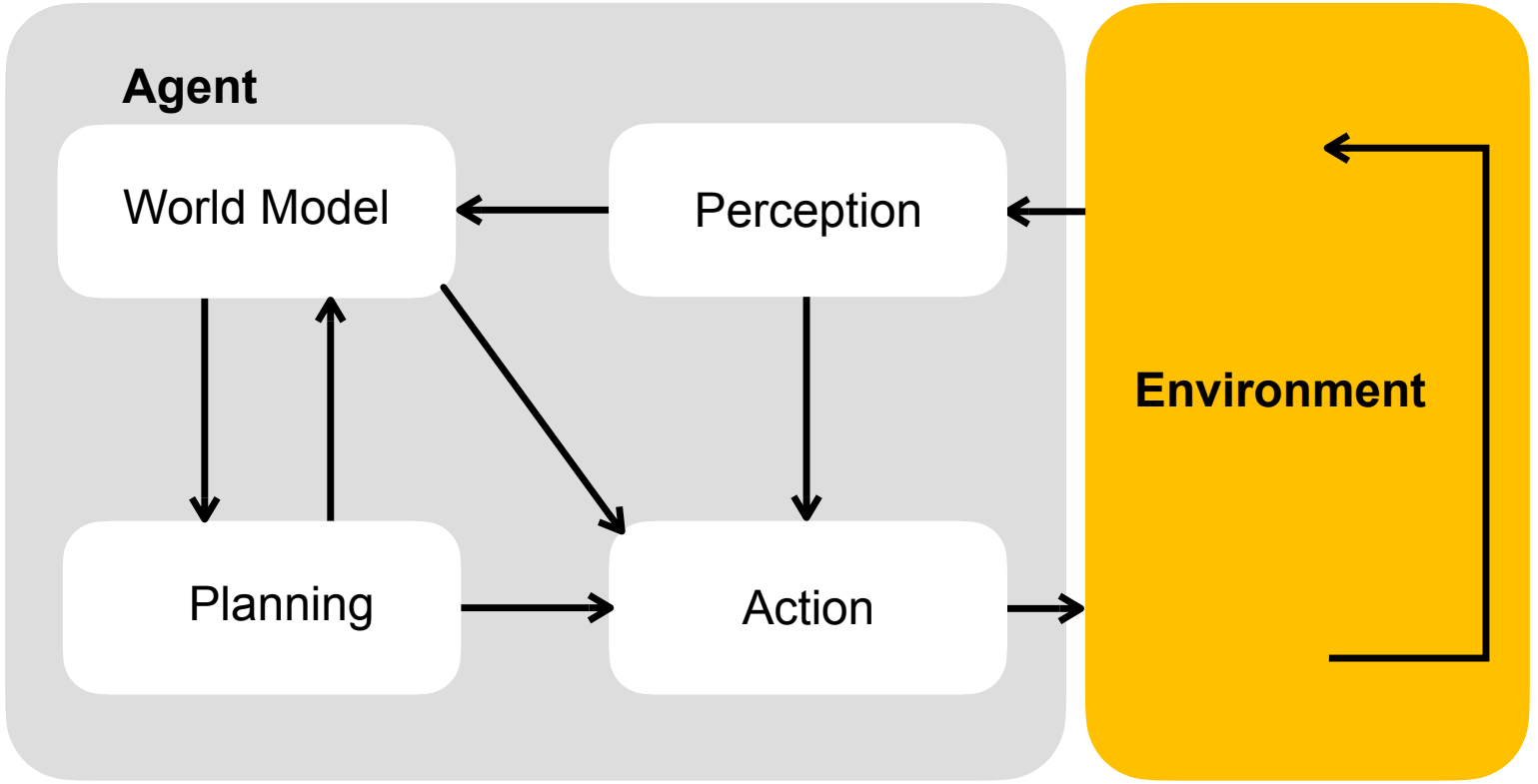
A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Limitations of Model-Based Agent

An agent with a world model but no planning can look into the past, but not into the future; it will perform poorly when the task requires any of the following:

- searching several moves ahead
  - Chess, Rubik's cube
- complex tasks requiring many individual step
  - cooking a meal, assembling a watch
- logical reasoning to achieve goals
  - travel to New York

Planning Agent

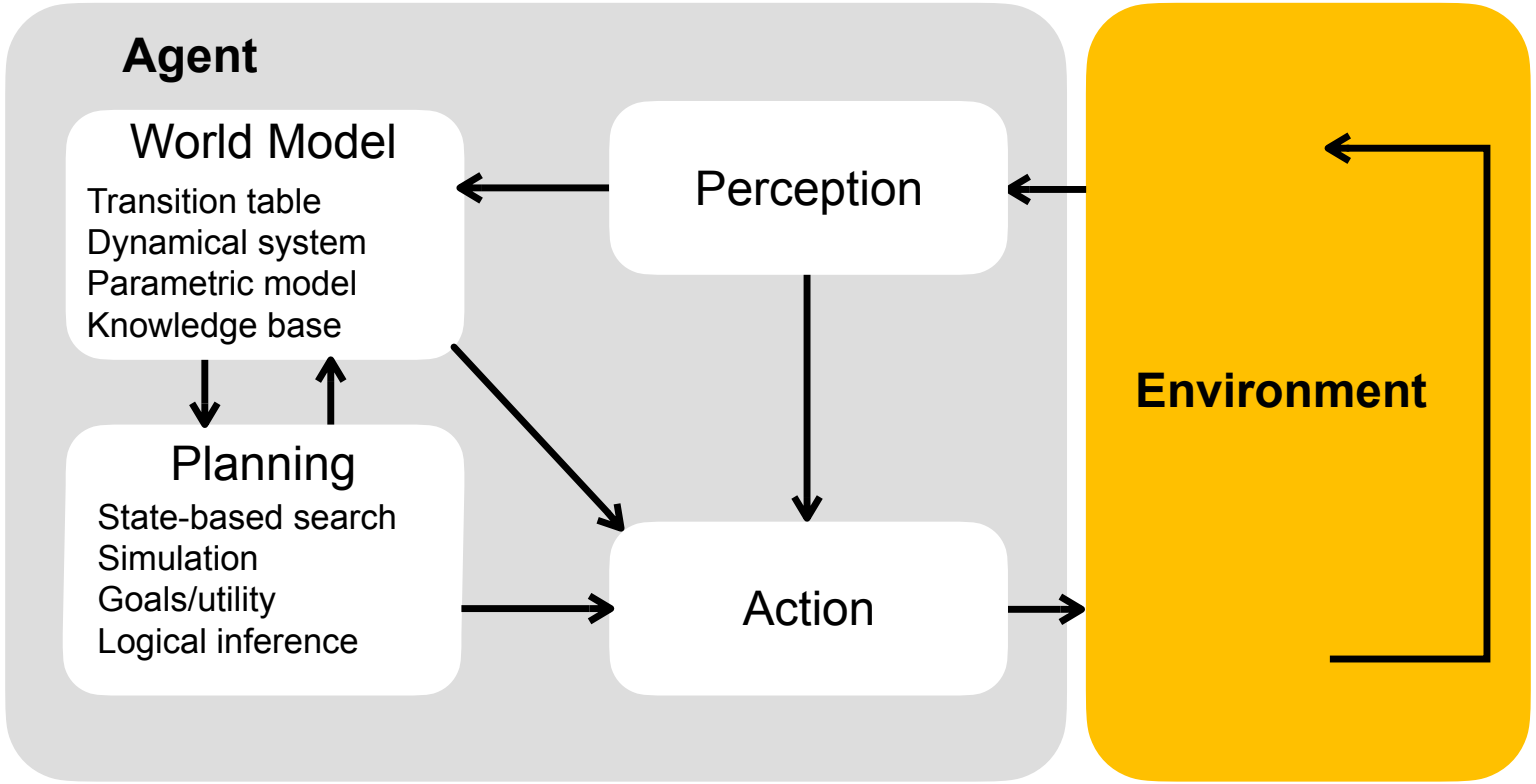


Planning Agent

- Decision making of this kind is fundamentally different from the condition–action rules
- It involves consideration of the future
  - “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 states to actions

Planning Agent

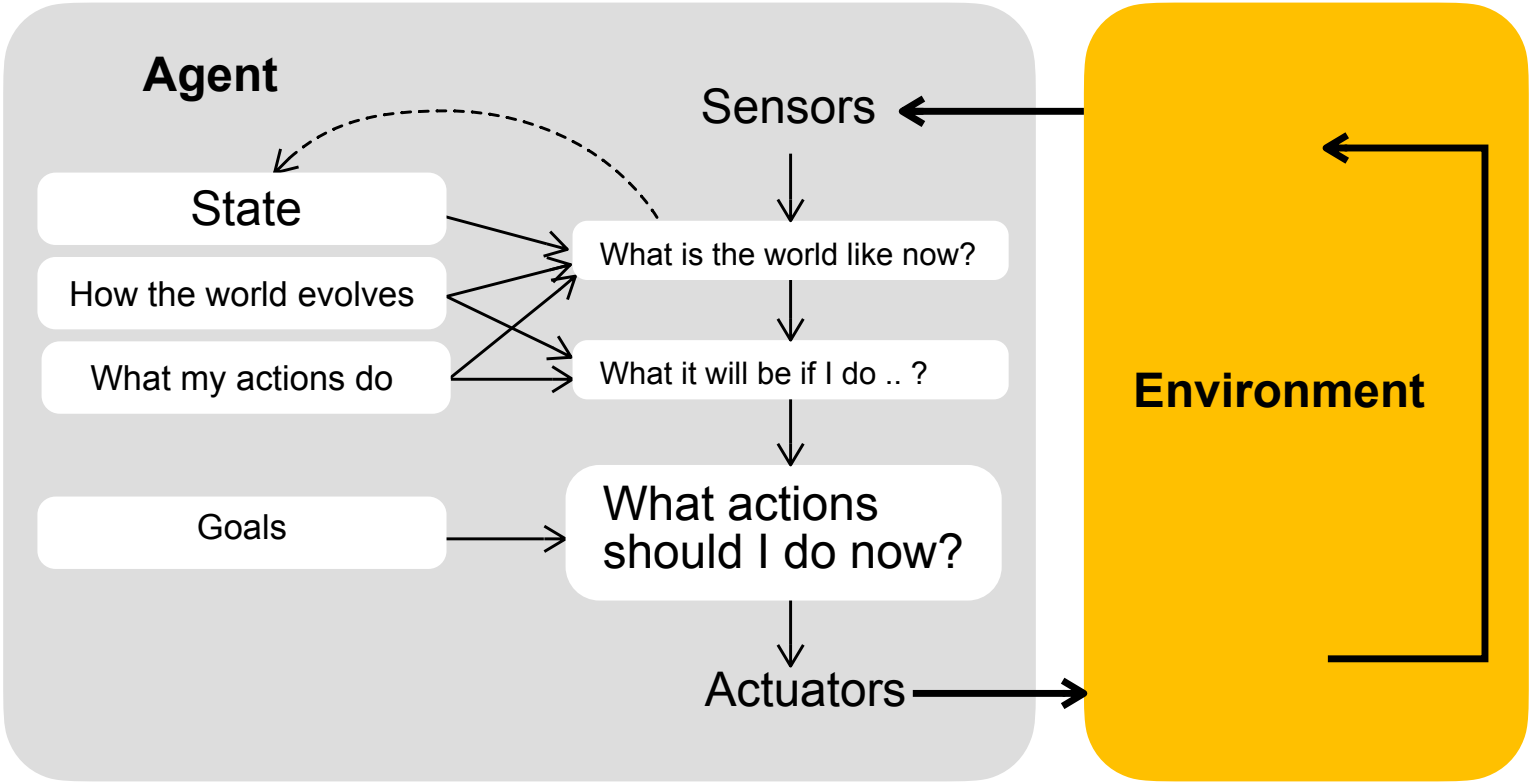


Reasoning about feature states

- What is the best action in this situation?

- Faking it
  - Sometimes an agent may appear to be planning ahead but is actually just applying reactive rules.
  - These rules can be hand-coded, or learned from experience.
  - Agent may appear intelligent, but is not flexible in adapting to new situations.

Goal-based (teleological) Agent



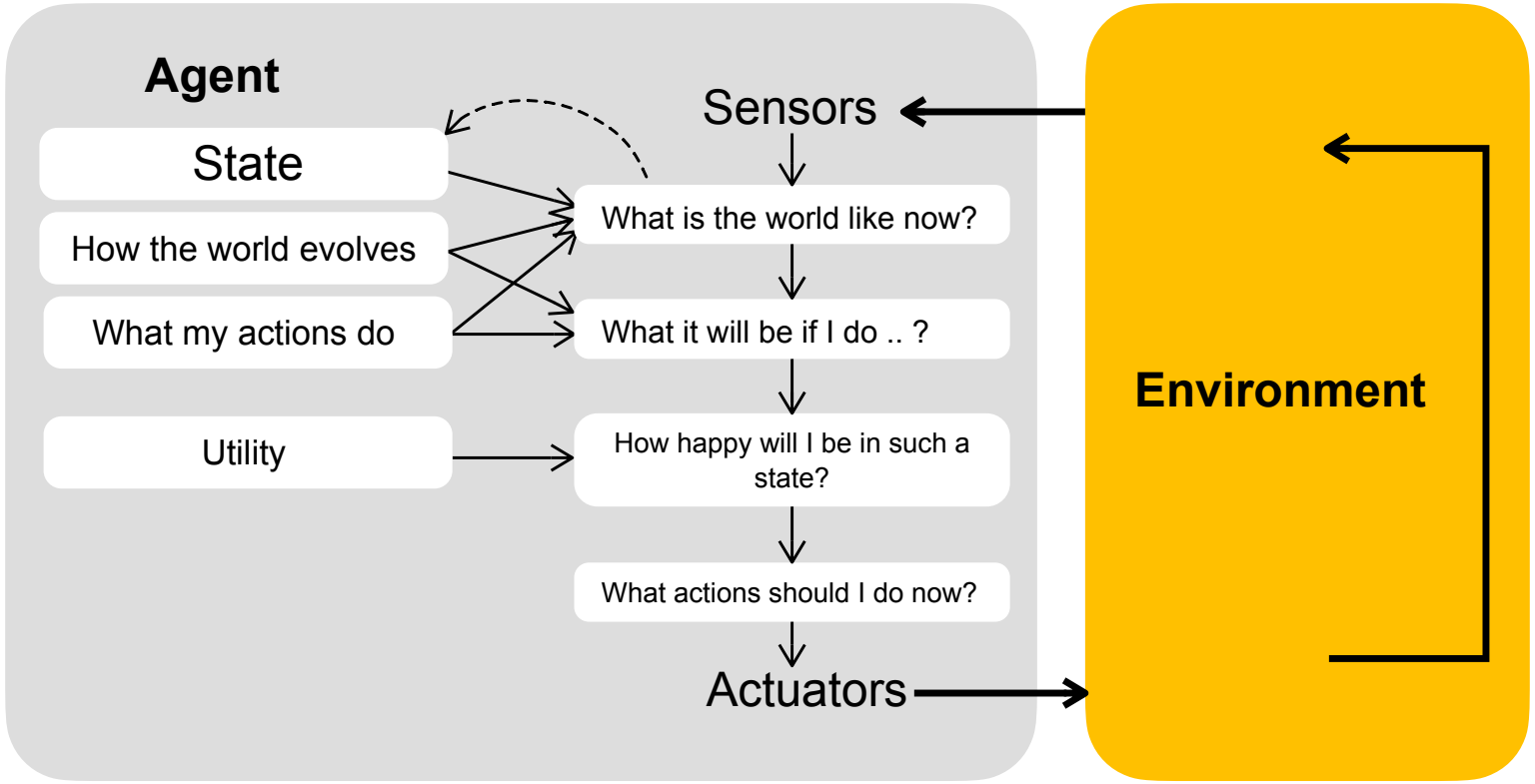
Goal-based (teleological) Agent

Goal-based (teleological) agent — state description often not sufficient for agent to decide what to do so it needs to consider its goals (may involve searching and planning)

- A planning, goal-based agent is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.

- The agent's behaviour can easily be changed.

Utility-based Agent

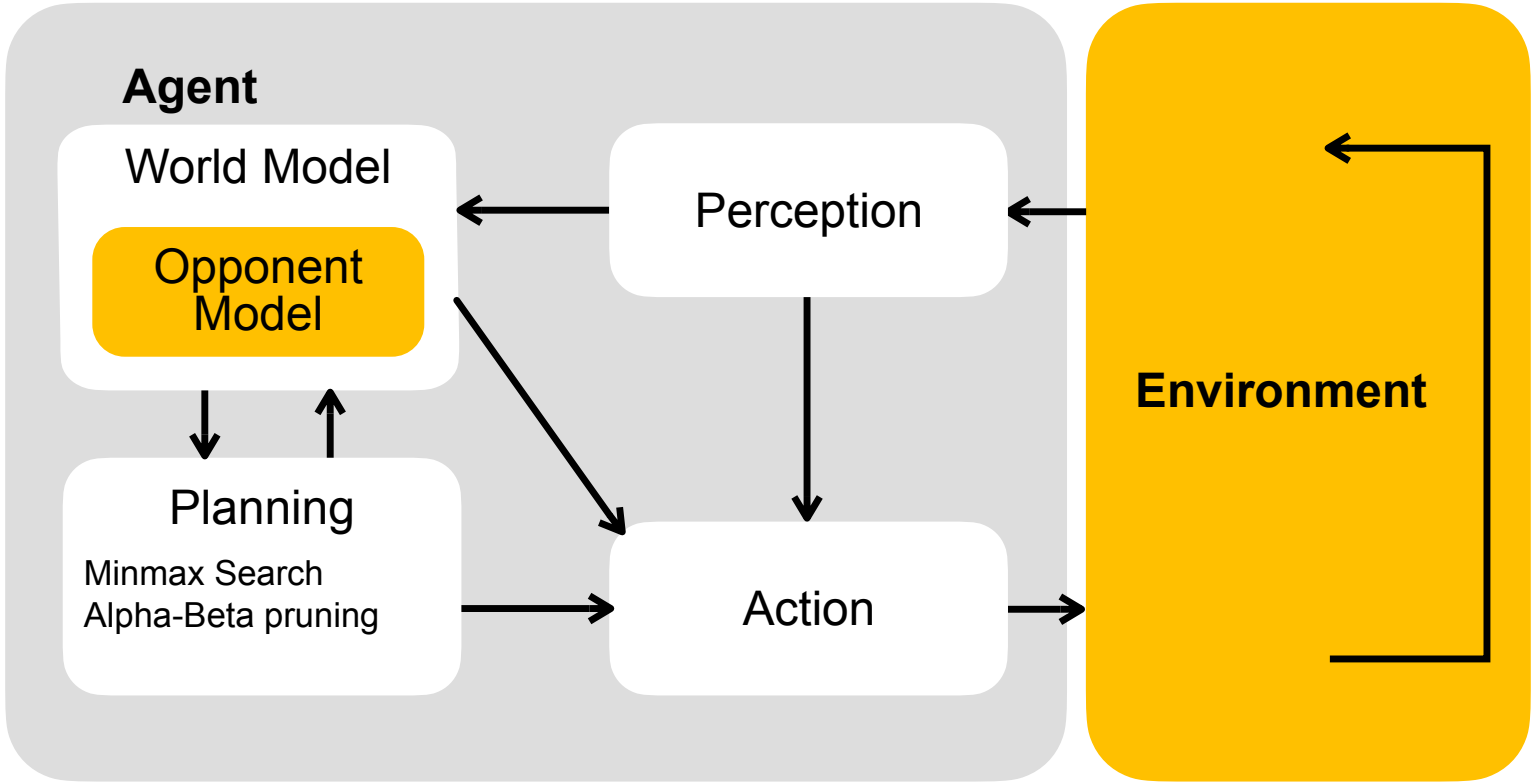


Utility-based Agent

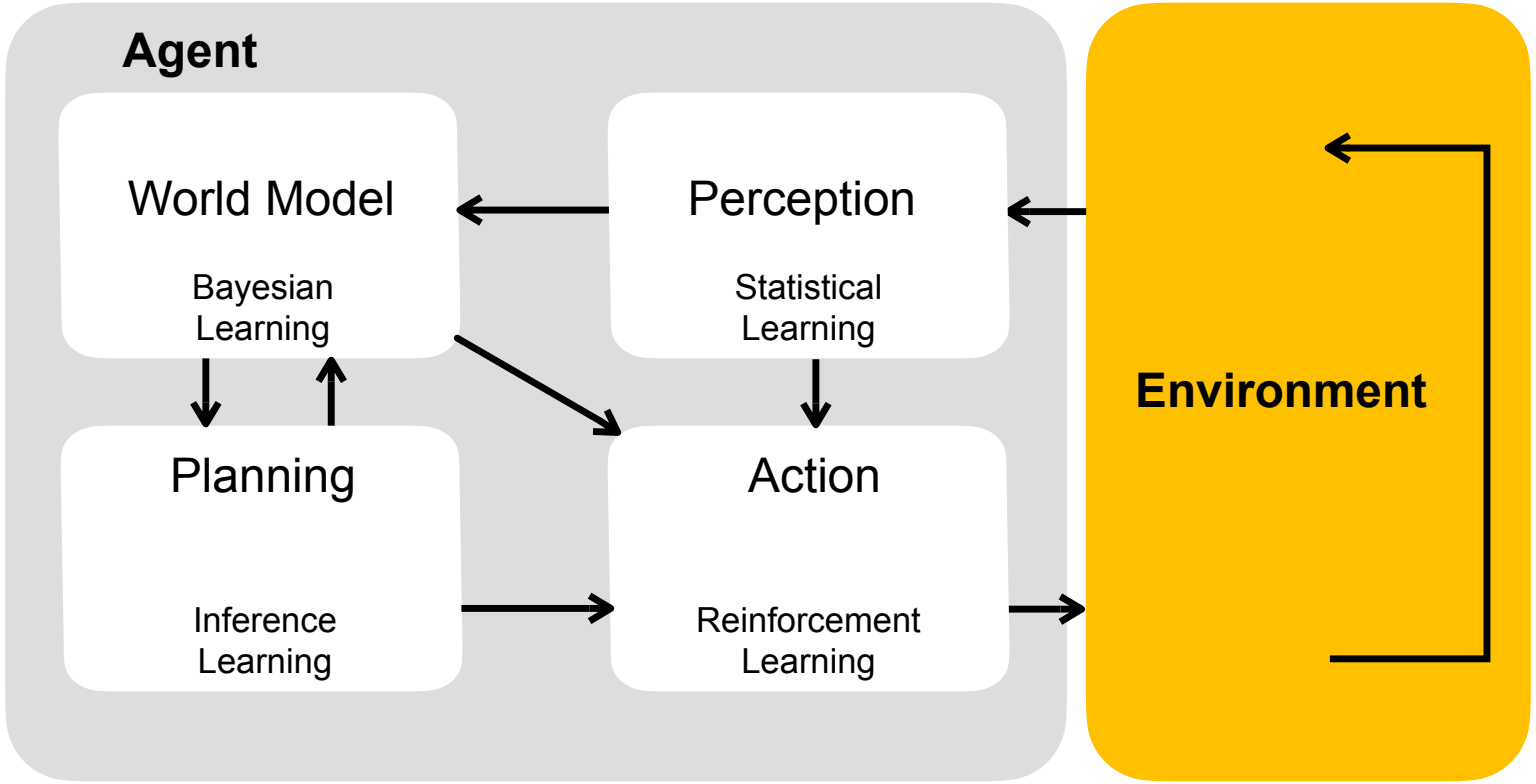
- A model-based, utility-based agent uses a model of the world, along with a utility function that measures its preferences among states of the world. It chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted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

- The utility-based agent is not easy to implement
  - It has to model and keep track of its environment
  - Tasks involved a great deal of research on perception, representation, reasoning, and learning.
  - It can be implemented as a Decision-making agent that must handle the uncertainty inherent in stochastic or partially observable environments.

Game Playing Agent



Learning Agent



Learning Agent

- Learning is not a separate module, but rather a set of techniques for improving the existing modules
- Learning is necessary because:
  - may be difficult or even impossible for a human to design all aspects of the system by hand
  - the agent may need to adapt to new situations without being re-programmed by a human

Agents - Summary

- The agent program implements the agent function.
- A variety of basic agent-program designs reflecting the kind of information made explicit and used in the decision process.
  - The designs vary in efficiency, compactness, and flexibility.
  - The appropriate design of the agent program depends on the nature of the environment.

- Simple reflex agents respond directly to percepts,
- model-based reflex agents maintain internal state to track aspects of the world that are not evident in the current percept.
- Planning (goal-based) agents act to achieve their goals, and
- Utility-based agents try to maximize their own expected “happiness.”
- All agents can improve their performance through learning.

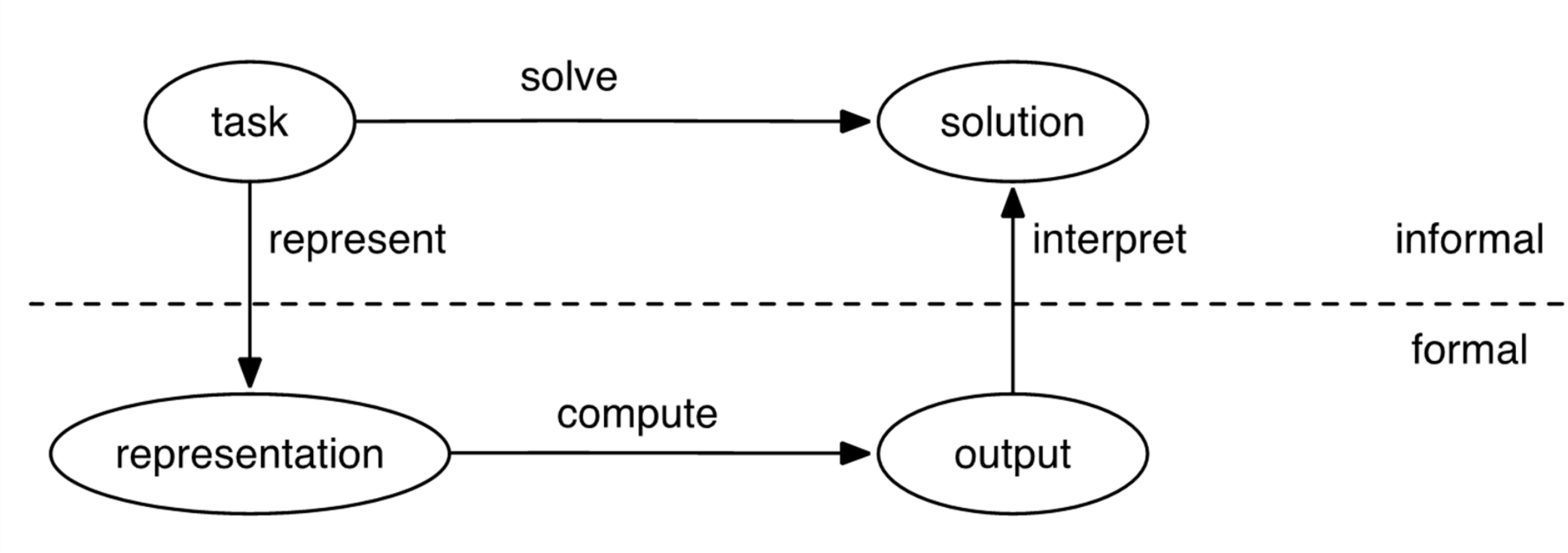
Tasks

One way that AI representations differ from computer programs in traditional languages that an AI representation typically specifies what **needs to be computed**, not how it is to be computed.

Much AI reasoning involves searching through the space of possibilities to determine how to complete a task.

Typically, a task is only given informally, such as “deliver parcels promptly when they arrive” or “fix whatever is wrong with the electrical system of the house.”

The role of representations in solving tasks



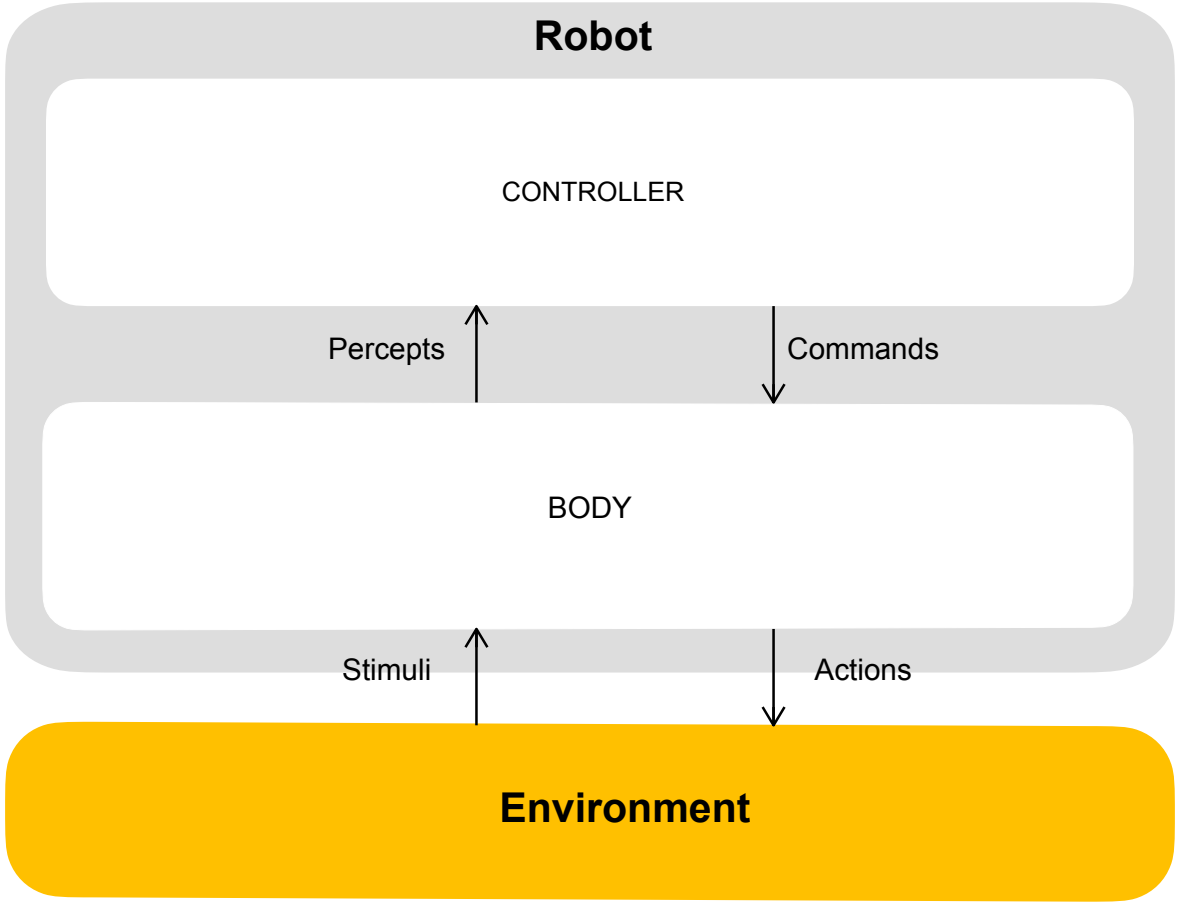
The general framework of solving tasks

- To solve a task, the designer of a system must:
  - determine what constitutes a solution
  - represent the task in a way a computer can reason about
  - use the computer to compute an output, which is answers presented to a user or actions to be carried out in the environment, and
  - interpret the output as a solution to the task.

Representation

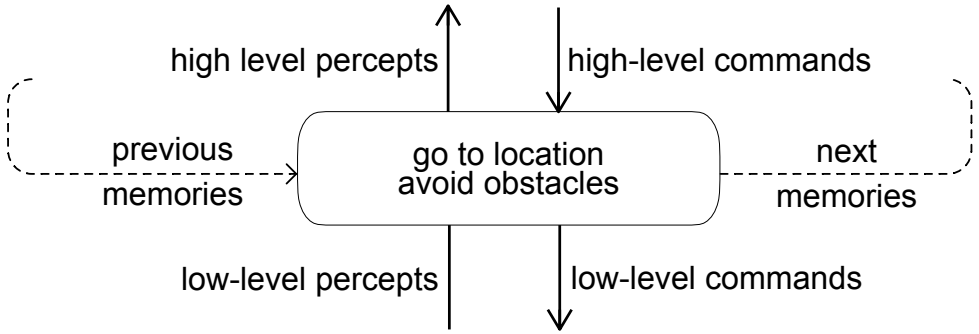
- Rich enough to express knowledge needed (to solve the problem)
- As close to the problem as possible: compact, natural, maintainable
- Amenable to efficient computation
  - Able to express features of the problem that can be exploited for computational gain
  - Able to trade off accuracy and computation time and/or space
  - Able to be acquired from people, data and past experiences

Agent system



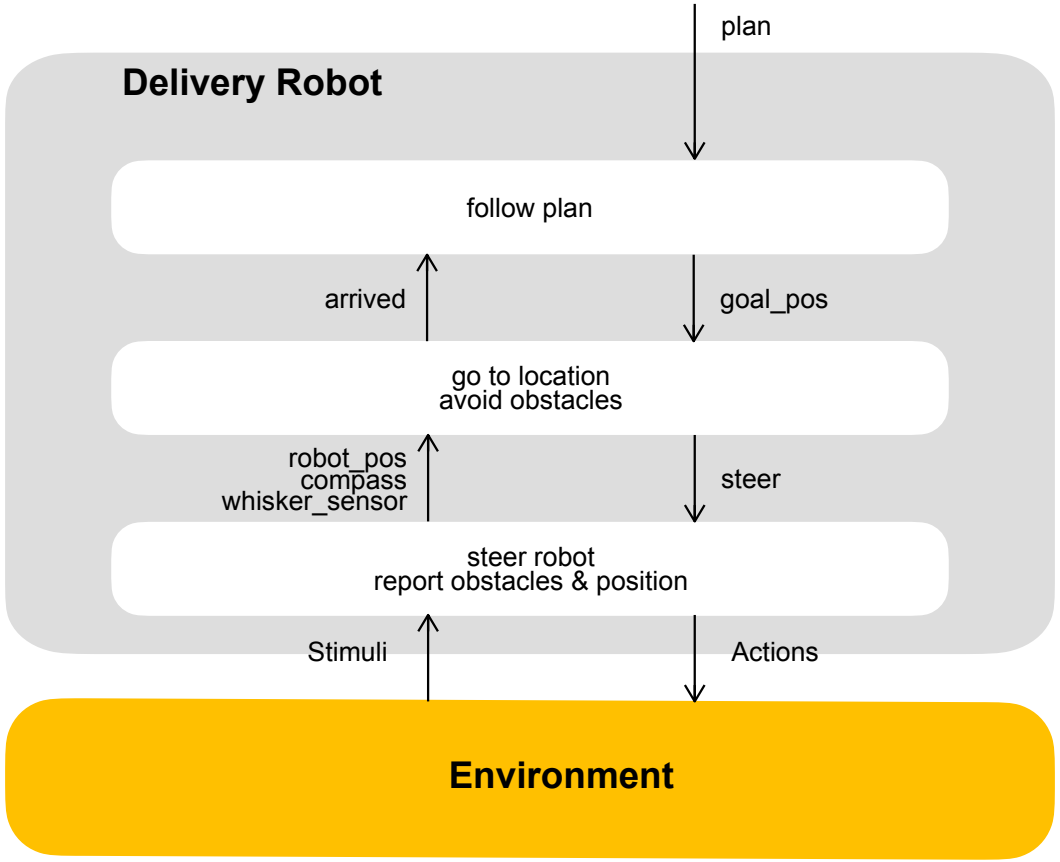
Layered architectures

- Hierarchy of controllers
- Controller gets percepts from and sends commands to the lower layer
  - Abstracts low level features into higher level (perception)
  - Translates high level commands into actuator instructions (action)

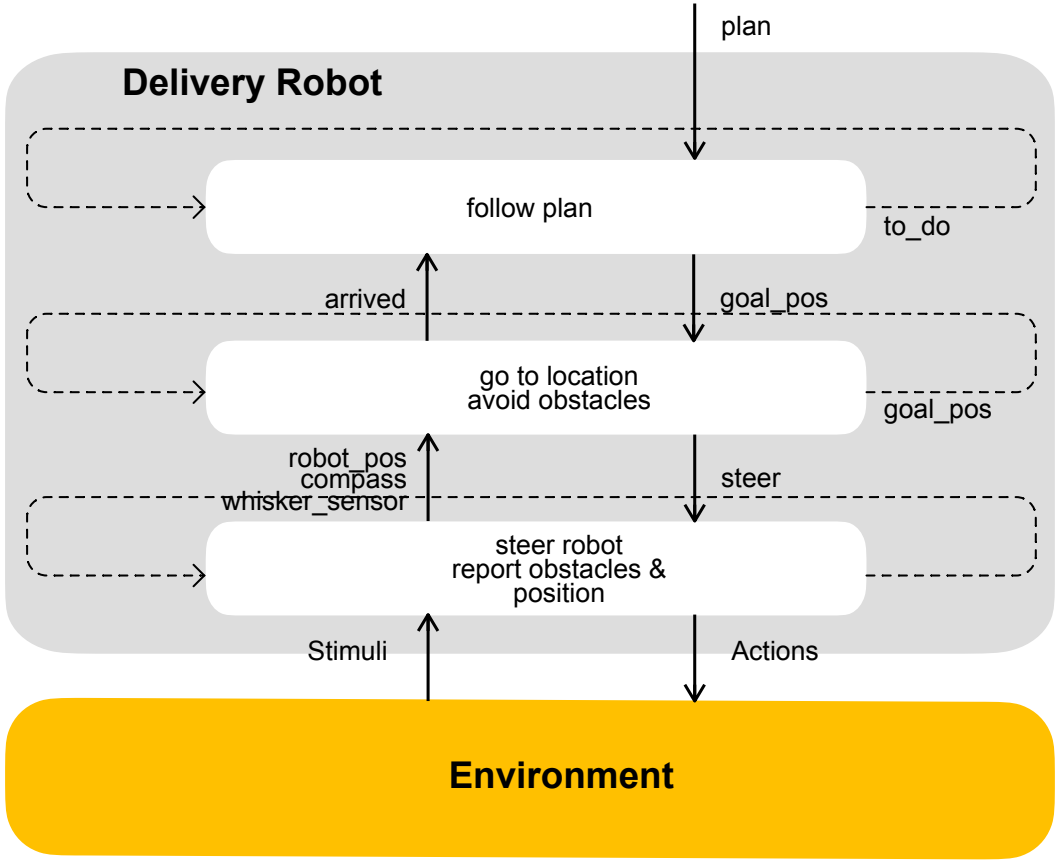


- The controllers have different representations, programs
- The controllers operate at different time scales
- A lower-level controller can override its commands

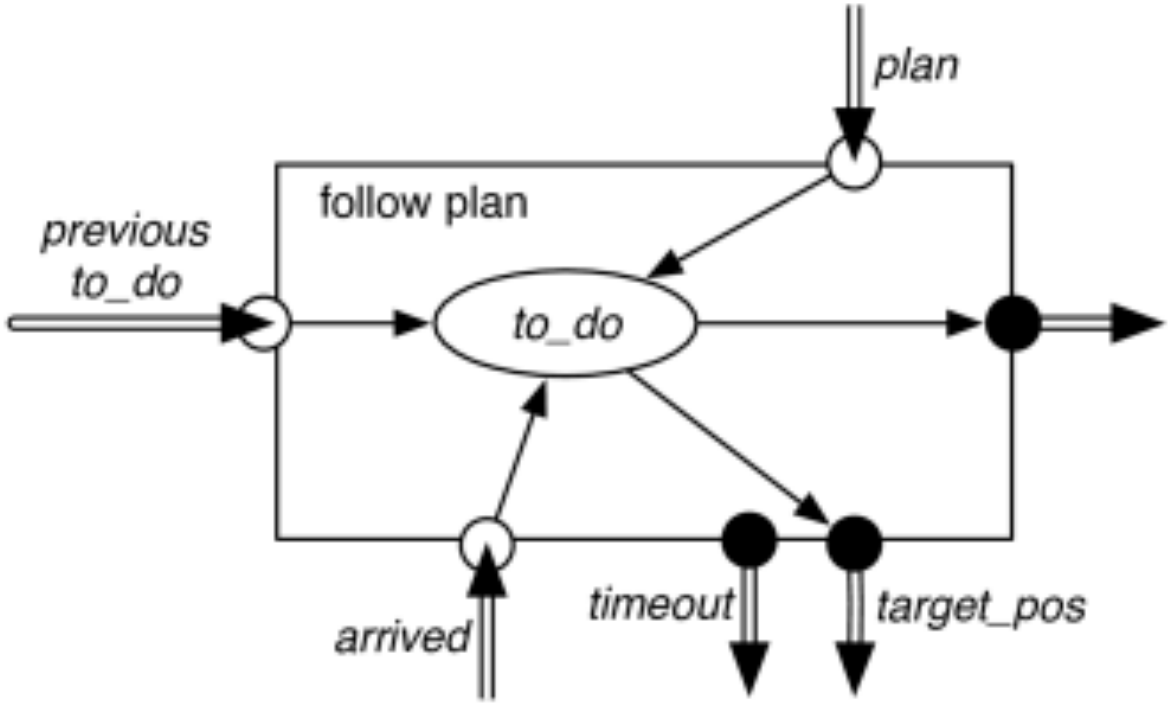
Example: delivery robot



Example: delivery robot



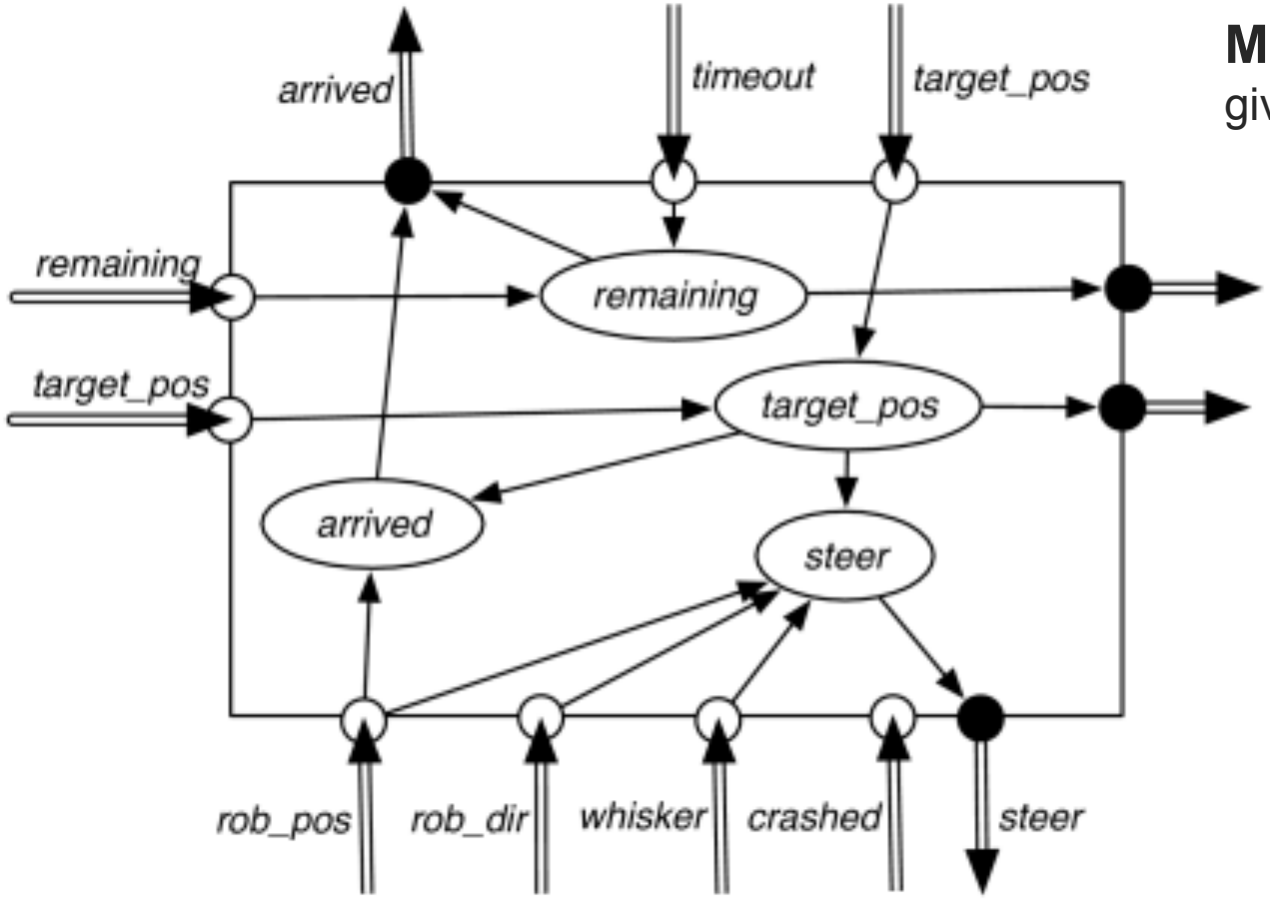
Delivery robot: the top layer



Top Layer Code

```
given plan:
    to do := plan
    timeout := 200
    while not empty(to do)
        target pos := coordinates(first(to do))
        do(timeout; target pos)
        to do := rest(to do)
```

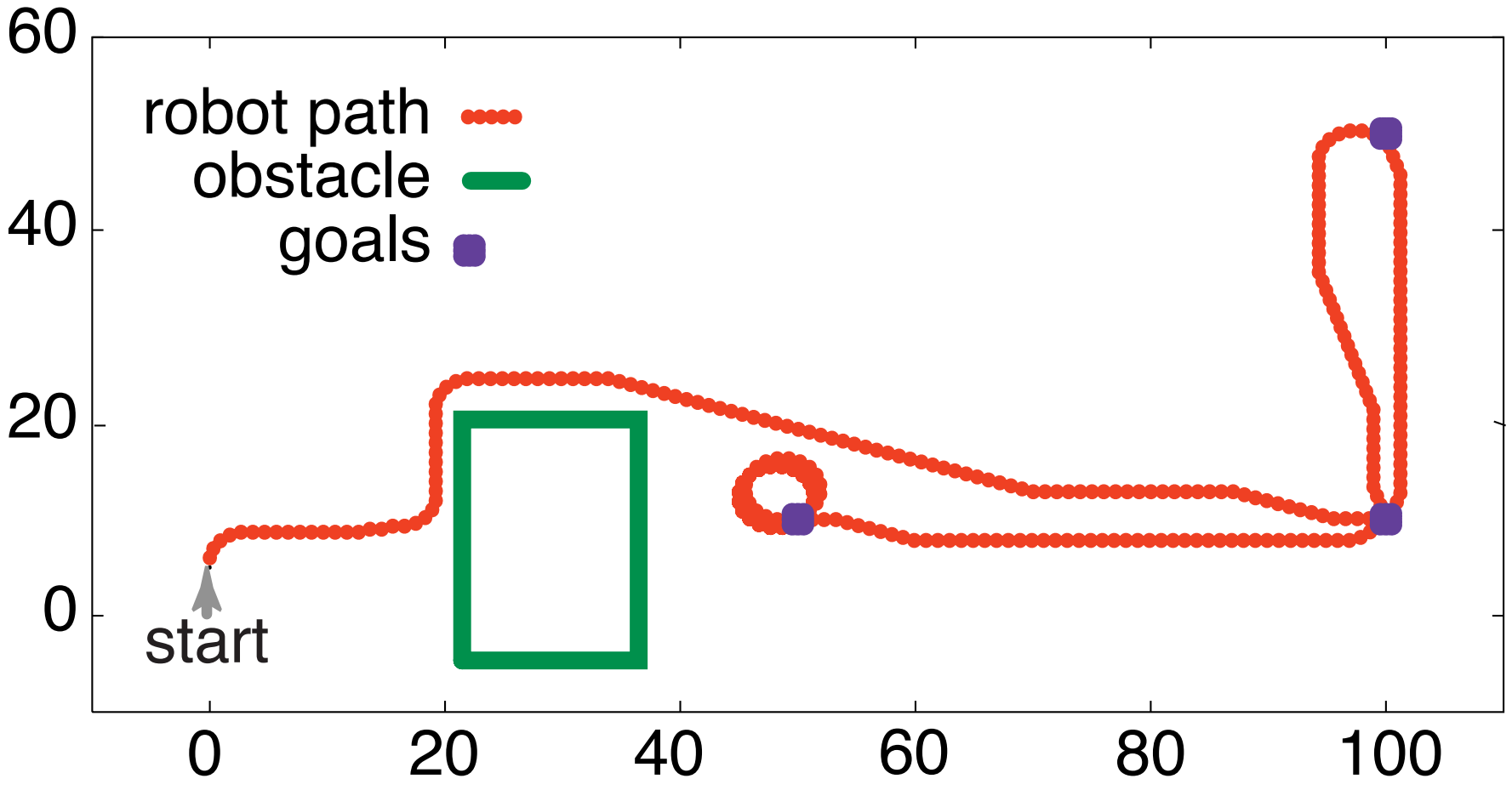
Delivery robot: the middle layer



Middle Layer Code

given timeout and target pos:
  remaining := timeout
  while not arrived() and remaining > 0
    if whisker sensor = on
      then steer := left
    else
      if straight ahead(rob pos; robot dir; target pos)
        then steer := straight
    else
      if left of (rob pos; robot dir; target pos)
        then steer := left
      else steer := right
      do(steer)
      remaining := remaining – 1
  tell upper layer arrived()

Delivery robot: simulation



Summary

- The agent program implements the agent function.
- A variety of basic agent-program designs reflecting the kind of information made explicit and used in the decision process.
  - The designs vary in efficiency, compactness, and flexibility.
  - The appropriate design of the agent program depends on the nature of the environment.

- Simple reflex agents respond directly to percepts,
- Model-based reflex agents maintain internal state to track aspects of the world that are not evident in the current percept.
- Planning (Goal-base) agents act to achieve their goals, and
- Utility-based agents try to maximise their own expected “happiness.”
- All agents can improve their performance through learning.