COE206 – Principles of Artificial Intelligence

Mustafa MISIR

Istinye University, Department of Computer Engineering

mustafa.misir@istinye.edu.tr

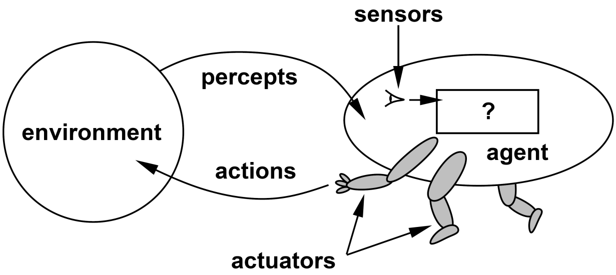
http://mustafamisir.github.io
http://memoryrlab.github.io

L2: Intelligent Agents

Outline

- Agents and Environments
- Rationality
- Nature of Environments
- Structure of Agents

Agents and Environments

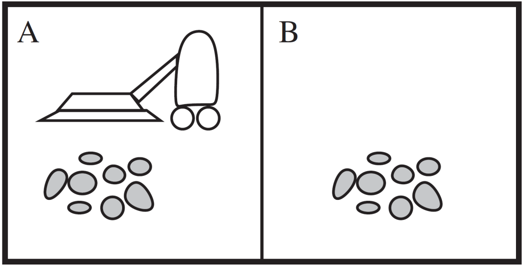


Agents include humans, robots, softbots, thermostats, etc.

- The agent function maps from percept histories to actions:

$$f : P^* \rightarrow A$$

Agents and Environments: e.g. Vacuum Cleaner



- **Percepts**: location and contents, e.g. [A, Dirty]
- **Actions**: Left, Right, Suck, NoOp

Agents and Environments: e.g. Vacuum Cleaner

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
$\vdots$	$\vdots$
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
$\vdots$	$\vdots$

Rationality

What is rational at any given time depends on 4 things:

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

Rational Agent

A **rational agent** should select an action maximizing its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rationality vs. Omniscience[2]

An omniscient agent knows the actual outcome of its actions and can act accordingly; yet omniscience is rather impossible in reality.

A cargo door falls off a passing airliner while crossing the street[1] while it is perfectly rational to pass.

- **Rationality vs. Perfection**: Rationality maximizes expected performance, while perfection maximizes actual performance.

[1] https://www.cbsnews.com/news/door-drops-off-boeing-767-rattles-seattle-suburb/

[2] Omniscience is a state of possessing all the knowledge there is

Rationality

Rationality does not require omniscience as the rational choice depends on the percept sequence to date. Still the agent should avoid engaging in decidedly underintelligent activities.

- e.g. if an agent does not look both ways before crossing a busy road, its percept sequence will not tell that there is a large truck approaching at high speed.

- irrational to cross the road given this uninformative percept sequence: the risk of accident.
- a rational agent should choose the *looking* action (**information gathering**) to maximize the expected performance

Rationality: Beyond Information Gathering

Learn as much as possible from what it perceives.

- The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented.

To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy.

- A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.
- After sufficient experience of its environment, the behavior of a rational agent can become effectively independent of its prior knowledge.

PEAS

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

- Performance measure
- Environment
- Actuators
- Sensors

PEAS: e.g. Taxi[3]

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver				

[3] image source: https://www.thedrive.com/tech/12722/Message

PEAS: e.g. Taxi

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

PEAS: e.g. Medical Diagnosis System[4]

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system				

[4] image source: https://medium.com/future-today/biomind-artificial-intelligence-that-defeats-doctors-in-tumour-diagnosis-5f8ec97298b2

PEAS: e.g. Medical Diagnosis System

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers

PEAS: e.g. Satellite Image Analysis System

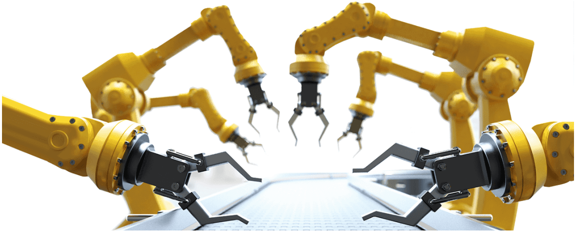
Agent Type	Performance Measure	Environment	Actuators	Sensors
Satellite image analysis system				

PEAS: e.g. Satellite Image Analysis System

Agent Type	Performance Measure	Environment	Actuators	Sensors
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays

PEAS: e.g. Part Picking Robot[5]

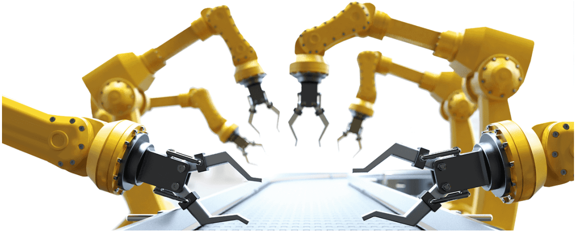
Agent Type	Performance Measure	Environment	Actuators	Sensors
Part-picking robot				



5 image source: https://www.robotics.org/blog-article.cfm/Pick-and-Place-Robots-What-Are-They-Used-For-and-How-Do-They-Benefit-Manufacturers/88

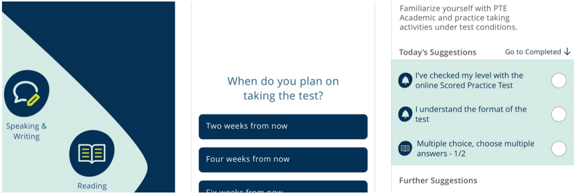
PEAS: e.g. Part Picking Robot

Agent Type	Performance Measure	Environment	Actuators	Sensors
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors



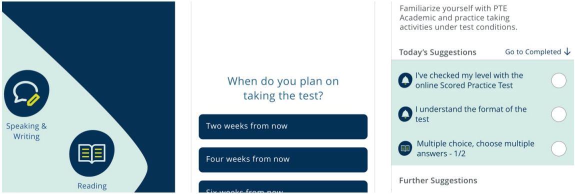
PEAS: e.g. Interactive English Tutor

Agent Type	Performance Measure	Environment	Actuators	Sensors
Interactive English tutor				



PEAS: e.g. Interactive English Tutor

Agent Type	Performance Measure	Environment	Actuators	Sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

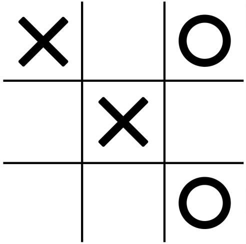


Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image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

PEAS: Tic-Tac-Toe[6]

TASK: List the PEAS description of Tic-Tac-Toe:

- Performance Measure, Environment, Actuators, Sensors



Submit photo of your answer to **Piazza** as a *private message*.
Also, deliver its hard copy.

[6] https://playtictactoe.org/ – Google Search: Tic Tac Toe

Environment Types

- Fully Observable vs. Partially Observable
- Single-agent vs. Multi-agent
- Deterministic vs. Stochastic
- Episodic vs. Sequential
- Static vs. Dynamic
- Discrete vs. Continuous

Environment Types: Fully vs. Partially Observable

- **Fully Observable**: Agent's sensors give it access to the complete state of the environment at each point in time
- **Partially Observable**: Otherwise, because of noisy and inaccurate sensors or parts of the state are simply missing from the sensor data – e.g. a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, missing from the sensor data

Environment Types: Single-agent vs. Multi-agent

e.g. an agent solving a crossword puzzle by itself is clearly in a **single-agent** environment, whereas an agent playing chess is in a **multi-agent** (two-agent) environment.

- A **single-agent** system might still have multiple entities[7] – several actuators, or even several robots. However, if each entity sends its perceptions to and receives its actions from a single central process, then there is only a single agent: the central process.
- In **multi-agent** scenarios, several agents exist which model each other's goals and actions.

[7] https://www.cs.cmu.edu/afs/cs/usr/pstone/public/papers/97MAS-survey/node4.html

Environment Types: Single-agent vs. Multi-agent

Does an agent A (e.g. the taxi driver) have to treat an object B (another vehicle) as an agent, or can it be treated as an object behaving according to the laws of physics?

- The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior.

e.g. 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 – a competitive multi-agent environment. – e.g. In the taxi-driving environment, avoiding collisions maximizes the performance measure of all agents, so it is a partially cooperative multi-agent environment.

- It is also partially competitive because, for example, only one car can occupy a parking space.

Environment Types: Deterministic vs. Stochastic

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.

Environment Types: Episodic vs. Sequential

- In an **episodic** environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. The next episode does not depend on the previous one – e.g. mail sorting system
- In **sequential** environments, the current decision could affect all future decisions.
  - e.g. 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.

Environment Types: Dynamic vs. Static

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

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**.

For example:

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

Environment Types: Discrete vs. Continuous

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.

- e.g. the chess environment has a finite number of distinct states.
  - Chess also has a discrete set of percepts and actions.
- e.g. 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.
  - Taxi-driving actions are also continuous (steering angles, etc.).

Known vs. Unknown

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.

- if the environment is unknown, the agent will have to learn how it works in order to make good decisions – e.g. 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.

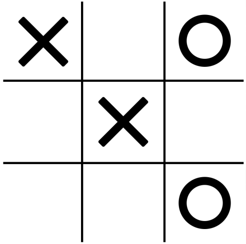
Environment Types

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

Environment Types: Tic-Tac-Toe[8]

TASK: Determine the Tic Tac Toe environment in terms of:

- Fully Observable / Partially Observable; Deterministic / Stochastic; Episodic / Sequential; Static / Dynamic; Discrete / Continuous



Submit photo of your answer to **Piazza** as a *private message*.
Also, deliver its hard copy.

[8] https://playtictactoe.org/ – Google Search: Tic Tac Toe

Agents

The job of AI is to design an agent program that implements the agent function—the mapping from percepts to actions.

We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the architecture:

$$agent = architecture + program$$

The program we choose has to be one that is appropriate for the architecture. e.g. if the program is going to recommend actions like Walk, the architecture had better have legs.

Agents

Anything problematic?

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*

Agents

Let $\mathcal{P}$ be the set of possible percepts and $\mathcal{T}$ is the lifetime of the agent, i.e. the total number of percepts it will receive.

- The lookup table will contain $\sum_{t=1}^{\mathcal{T}} |\mathcal{P}|^t$ entries, of possibly increasing numbers...

183	190	191	215	195	206	224	166	154	173	192	198	167	180	190	222
169	174	139	178	173	150	201	227	210	139	142	193	160	227	209	155
172	152	225	216	197	230	196	187	130	157	182	181	178	202	221	176
221	176	164	137	153	190	233	192	162	141	208	176	199	214	201	158
195	223	200	191	174	217	160	152	189	130	144	133	218	186	215	165
207	189	137	189	167	183	224	169	172	187	151	173	168	186	173	192
206	206	205	148	234	232	157	150	235	139	197	230	199	221	164	199
143	185	144	208	139	182	169	229	228	209	199	216	189	134	187	184

Agents

Still, the table-driven agent does what we want, i.e. implements the desired agent function.

- The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table.

e.g. the huge tables of square roots used prior to the 1970s have now been replaced by a five-line program for Newton's method running on electronic calculators.

- Can AI do for general intelligent behavior what Newton did for square roots?

Agent Types

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

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

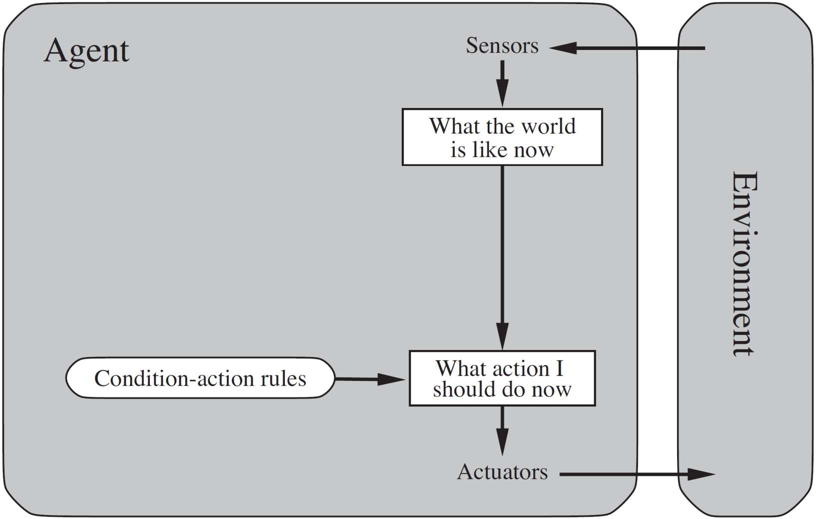
Agent Types: Simple Reflex Agents

Acts according to a rule whose condition matches the current state, as defined by the percept.

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

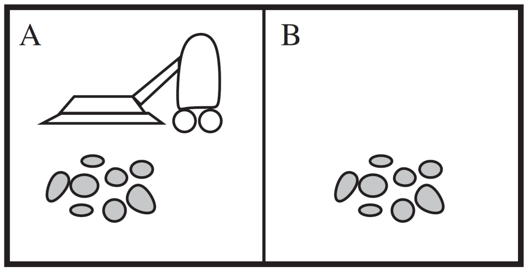
  $state \leftarrow$ INTERPRET-INPUT($percept$)
  $rule \leftarrow$ RULE-MATCH($state, rules$)
  $action \leftarrow rule$.ACTION
  return *action*

Agent Types: Simple Reflex Agents



Agent Types: Simple Reflex Agents

TASK : Implement a Simple Reflex Agent for a Vacuum Cleaner



Submit your code to **Piazza** as a *private message*.

Agent Types: Simple Reflex Agents

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

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*

The most obvious reduction comes from ignoring the percept history. A further, small reduction comes from the fact that when the current square is dirty, the action does not depend on the location.

Agent Types: Simple Reflex Agents

Imagine yourself as the driver of the automated taxi. If the car in front brakes and its brake lights come on, then you should notice this and initiate braking – **condition-action** rule:

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

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

- car breaking example works if the car in front has a centrally mounted brake light. A simple reflex agent driving behind such a car would either brake continuously and unnecessarily, or, worse, never brake at all.

Agent Types: Simple Reflex Agents

Referring to the vacuum cleaner example

- Moving Left fails (forever) if it happens to start in square A, and moving Right fails (forever) if it happens to start in square B. Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments.
- Escape from infinite loops is possible if the agent can randomize its actions.

Agent Types: Model-based 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.

- Simple Reflex Agents rely on the **current percept** in the fully observable environments.

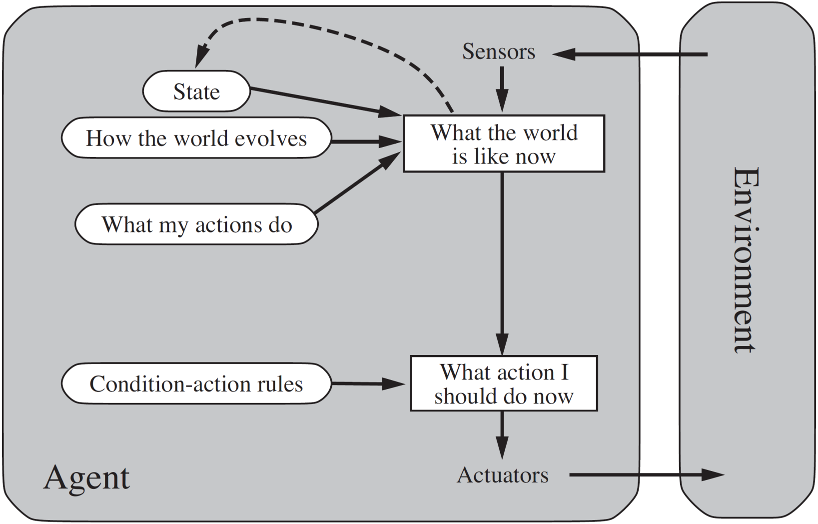
Agent Types: Model-based Agents

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

- How the world evolves independently of the agent, e.g. that an overtaking car generally will be closer behind than it was a moment ago.
- How the agent's own actions affect the world, e.g. that when the agent turns the steering wheel clockwise, the car turns to the right.

This knowledge about "how the world works" called a **model** of the world. An agent that uses such a model is called a **model-based agent**.

Agent Types: Model-based Agents



Agent Types: Model-based Agents

Keeps track of the current state of the world, using an internal model. 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
    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

  $state \leftarrow$ UPDATE-STATE($state, action, percept, model$)
  $rule \leftarrow$ RULE-MATCH($state, rules$)
  $action \leftarrow rule$.ACTION
  return *action*

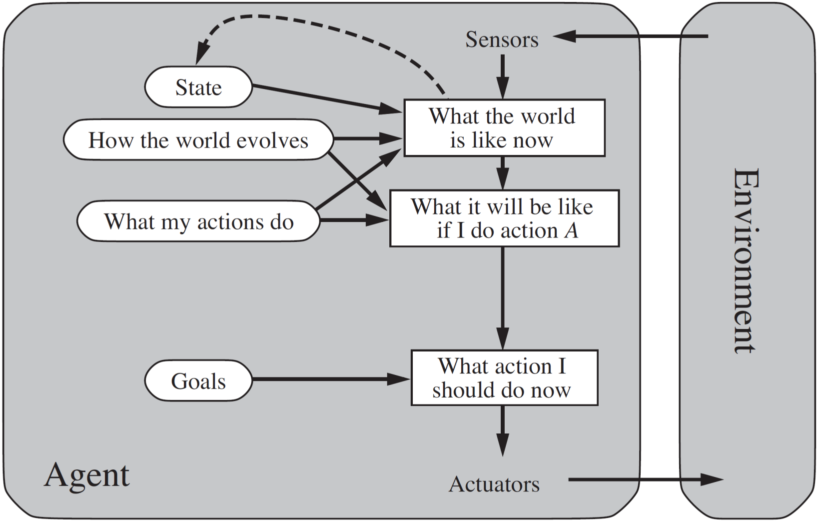
Agent Types: Goal-based Agents

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

- as well as a current state description, the agent needs some sort of goal information that describes situations that are desirable – e.g. being at the passenger's destination.

The agent program can combine this with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal.

Agent Types: Goal-based Agents



Agent Types: Utility-based Agents

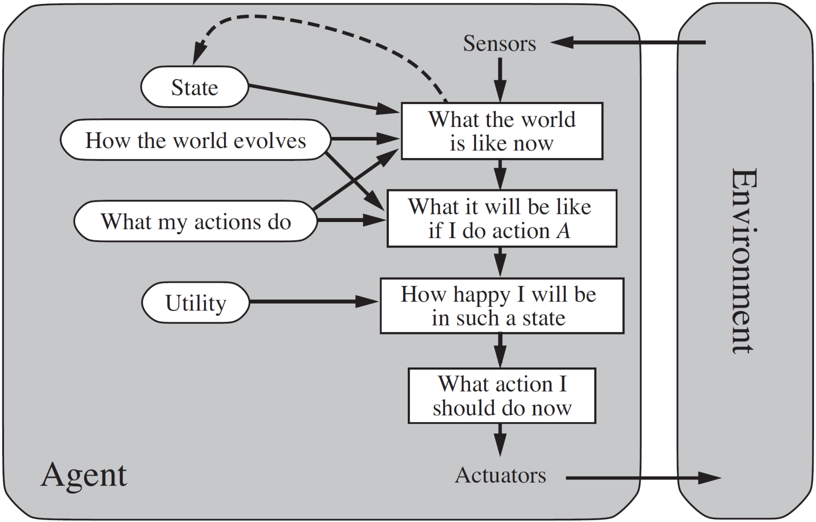
Goals alone are not enough to generate high-quality behavior in most environments – e.g. example, many action sequences will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others.

- Goals just provide a crude binary distinction between "happy" and "unhappy" states.

Instead, prefer **utility** which is the quality of being useful, calculated by a **utility function** as a performance measure.

- Choose the action that leads to the best expected utility.

Agent Types: Utility-based Agents



Agent Types: General Learning Agent

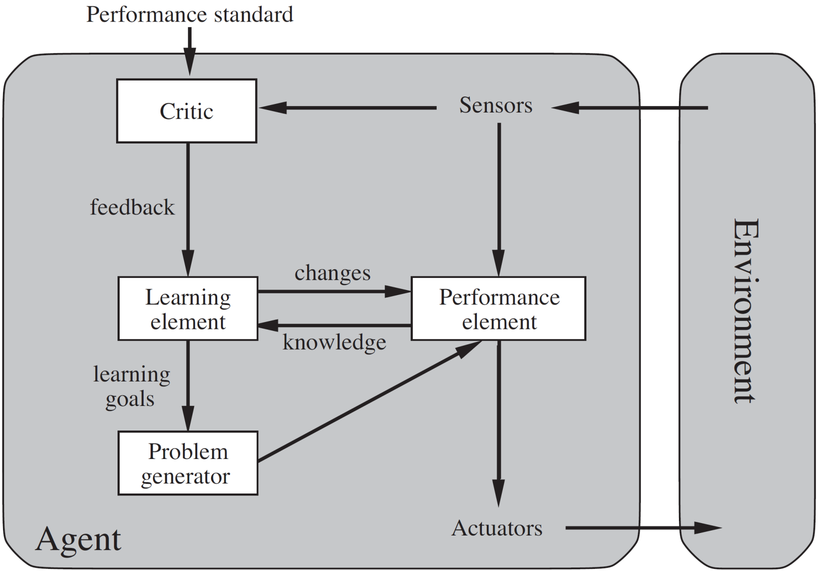
A learning agent can be divided into **four conceptual components**:

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

Agent Types: General Learning Agent – e.g. Automated Taxi

- **learning element**: if the taxi exerts a certain braking pressure when driving on a wet road, then it will soon find out how much deceleration is actually achieved.
- **performance element**: goes out on the road and drives
- **critic**: the taxi makes a left turn across three lanes of traffic, the critic observes the bad language used by other drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by installation of the new rule.
- **problem generator**: trying out the brakes on different road surfaces under different conditions

Agent Types: General Learning Agent



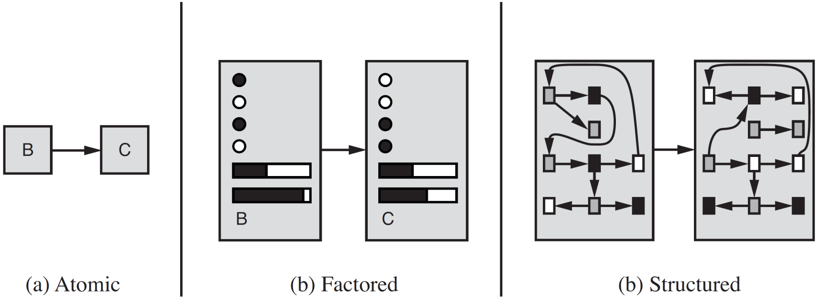
Agent Programs

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, realvalued, 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.



(a) Atomic (b) Factored (b) Structured

Summary

- An **agent** is something that perceives and acts in an environment.
- The **performance measure** evaluates the behavior of the agent in an environment. A rational agent acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far.
- A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors. They can be fully or partially observable, single-agent or multiagent, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.
- The **agent program** implements the agent function.
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept. **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.