Foundations of Artificial Intelligence
Chapter 02: **Intelligent Agents**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
https://disi.unitn.it/rseba/DIDATTICA/fai_2026/

Teaching assistant: Nicola Debole, nicola.debole@unitn.it,

M.S. Course “Artificial Intelligence Systems”, academic year 2026-2027

Last update: Monday 14th September, 2026, 12:24

Outline

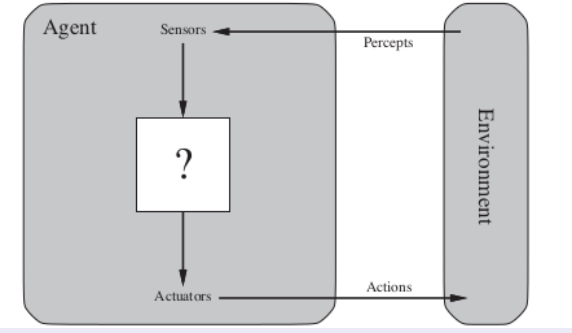
Outline

Agents and Environments

Agents

An agent is any entity that can be viewed as:

- perceiving its environment through sensors, and
- acting upon that environment through actuators.



(© S. Russell & P. Norwig, AIMA)

Agents and Environments: Examples

Agents

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

- human:
  - perceives: with eyes, ears, nose, hands, ...,
  - acts: with voice, hands, arms, legs, ...
- 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
- thermostat:
  - perceives: with heat sensor
  - acts: electric impulses to valves, devices, ...

Key concepts

Percept and Percept sequences

- percept: the collection of agent's perceptual inputs at any given instant
- percept sequence: the complete history of everything the agent has ever perceived

An agent's choice of action at any given instant

- can depend on the entire percept sequence observed to date
- does not depend on anything it hasn't perceived

Remark: An agent can perceive its own actions, but not always it effects.

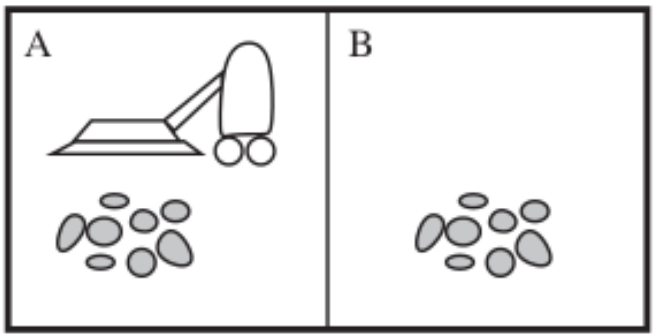
Agent function and agent program

- An agent's behavior is described by the agent function $f : P^* \longmapsto A$
  - maps any given percept sequence into an action
  - ideally, can be seen as a table [*percept sequence*, *action*]
- Internally, the agent function for an artificial agent is implemented by an agent program.

Example

A very-simple vacuum cleaner

- Environment: squares A and B
- Percepts: location ($\{A, B\}$) and content ($\{Dirty, Clean\}$)
  - e.g. [A, Dirty]
- Actions: $\{left, right, suck, no\_op\}$



(© S. Russell & P. Norwig, AIMA)

Example [cont.]

Simple vacuum cleaner's agent function and program

If the current square is dirty, then suck; otherwise, move to the other square.

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

(© S. Russell & P. Norwig, AIMA)

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** $Left$

(© S. Russell & P. Norwig, AIMA)

Rational Agents

Definition of a rational agent

A rational agent, for each possible percept sequence, 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.

Four ingredients:

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

Rational Agents: Example

The simple vacuum-cleaner agent

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*

(© S. Russell & P. Norwig, AIMA)

Under the following assumptions:

- Actions: Left, Right, Suck
- Environment knowledge:
  - "geography" known a priori,
  - dirt distribution and agent initial location unknown
  - [clean squares cannot become dirty again]
- Perception: self location, presence of dirt
- Performance measure:
  one point for each clean square at each time step, over 1000 time steps

Is the above-described agent rational? $\Longrightarrow$ Yes! (provided the given performence measure)

Beware: if a penalty for each move is given, the agent behaves poorly
$\Longrightarrow$ better agent: do nothing once it is sure all the squares are clean

Outline

Task Environments

Description of Task Environments (“PEAS”)

- To design a rational agent we must specify its task environment
  - i.e. the “problems” to which rational agents are the “solutions”
- Task environment described in terms of four elements (“PEAS”):
  - Performance measure
  - Environment
  - Actuators
  - Sensors

Simple Example: Simple Vacuum Cleaner

- Performance measure: 1 point per clean square per time step
- Environment: squares A and B, possibly dirty
- Actuators: move left/right, suck
- Sensors: self location, presence of dirt

Task Environments [cont.]

Complex Example: Autonomous Taxi

- Performance measure: safety, destination, profits, comfort, ...
- Environment: streets/freeways, other traffic, pedestrians, ...
- Actuators: steering, accelerator, brake, horn, speaker/display, ...
- Sensors: video, sonar, speedometer, engine sensors, GPS, ...

Remark

Some goals to be measured may conflict!

- e.g. profits vs. safety, profits vs. comfort, ...
  $\Longrightarrow$ tradeoffs are required

Task Environments: Examples

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

Properties of Task Environments

Task environments can be categorized along six dimensions:

- Single-agent vs. multi-agent
- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous

Properties of Task Environments [cont.]

Single-agent vs. multi-agent

- A task environment is multi-agent iff contains other agents who are also maximizing some performance measure that depends on the current agent's actions
  - latest condition essential
  - distinction between single- and multi-agent sometimes subtle
- Two important cases
  - competitive multi-agent environment:
    other agents' goals conflict with, or even oppose to, the agent's goals
    - Ex: chess, war scenarios, taxi driving (compete for parking lot), ...
  - cooperative multi-agent environment:
    other agents' goals coincide in full, or in part, with the agent's goals
    - Ex: ants' nest, factory, taxi driving (avoid collisions), ...
- Different design problems for multi-agent wrt. single-agent
  - competitive: randomized behaviour often rational (unpredictable)
  - collaborative: communication with other agents often rational

Properties of Task Environments [cont.]

Fully observable vs. partially observable

- A task environment is fully observable iff the sensors detect the complete state of the environment

  $\Longrightarrow$ no need to maintain internal state to keep track of the environment
- A task environment is effectively fully observable iff the sensors detect all aspects of the state of the environments that are relevant to the choice of action
  - ”relevant” depends on the performance measure
- A task environment may be partially observable (Ex: Taxi driving):
  - noisy and inaccurate sensors
  - parts of the state are not accessible for sensors
- A task environment might be even unobservable (no sensors)
  - e.g. fully-deterministic actions

Properties of Task Environments [cont.]

Deterministic vs. stochastic

- A task environment is deterministic iff its next state is completely determined by its current state and by the action of the agent. (Ex: a crossword puzzle).
- If not so:
  - A t.e. is stochastic if uncertainty about outcomes is quantified in terms of probabilities (Ex: dice, poker game, component failure,...)
  - A t.e. is nondeterministic iff actions are characterized by their possible outcomes, but no probabilities are attached to them

In a multi-agent environment we ignore uncertainty that arises from the actions of other agents (Ex: chess is deterministic even though each agent is unable to predict the actions of the others).

A partially observable environment could appear to be nondeterministic/stochastic.
$\Longrightarrow$ for practical purposes, when it is impossible to keep track of all the unobserved aspects, they must be treated as nondeterministic/stochastic.
(Ex: Taxi driving)

Properties of Task Environments [cont.]

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

  $\Longrightarrow$ episodes do not depend on the actions taken in previous episodes, and they do not influence future episodes
  - Ex: an agent that has to spot defective parts on an assembly line,
- In sequential environments the current decision could affect future decisions

  $\Longrightarrow$ actions can have long-term consequences
  - Ex: chess, taxi driving, ...
- Episodic environments are much simpler than sequential ones
  - No need to think ahead!

Properties of Task Environments [cont.]

Static vs. dynamic

- The task environment is dynamic iff it can change while the agent is choosing an action, static otherwise
  - $\Longrightarrow$ agent needs keep looking at the world while deciding an action
    - Ex: crossword puzzles are static, taxi driving is dynamic
- The t.e. is semidynamic if the environment itself does not change with time, but the agent's performance score does
  - Ex: chess with a clock
- Static environments are easier to deal wrt. [semi]dynamic ones

Properties of Task Environments [cont.]

Discrete vs. continuous

- The state of the environment, the way time is handled, and agents percepts & actions can be discrete or continuous
  - Ex: Crossword puzzles: discrete state, time, percepts & actions
  - Ex: Taxi driving: continuous state, time, percepts & actions
  - ...

Properties of Task Environments [cont.]

Note

- The simplest environment is fully observable, single-agent, deterministic, episodic, static and discrete.
  - Ex: simple vacuum cleaner
- Most real-world situations are partially observable, multi-agent, stochastic, sequential, dynamic, and continuous.
  - Ex: taxi driving

(See also other examples in the AIMA book.)

Properties of the Agent's State of Knowledge

Known vs. unknown

- Describes the agent's (or designer's) state of knowledge about the "laws of physics" of the environment
  - if the environment is known, then the outcomes (or outcome probabilities if stochastic) for all actions are given.
  - if the environment is unknown, then the agent will have to learn how it works in order to make good decisions
- Orthogonal wrt. task-environment properties

Known $\neq$ Fully observable

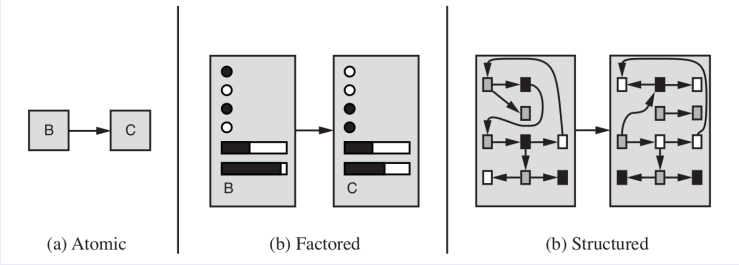
- a known environment can be partially observable
  (Ex: a solitaire card games)
- an unknown environment can be fully observable
  (Ex: a game I don't know the rules of)

Outline

Representations

Representations of states and transitions

- Three ways to represent states and transitions between them:
  - atomic: a state is a black box with no internal structure
  - factored: a state consists of a vector of attribute values
  - structured: a state includes objects, each of which may have attributes of its own as well as relationships to other objects
- increasing expressive power and computational complexity
- reality represented at different levels of abstraction



Representations [cont.]

Atomic Representations

- each state of the world is indivisible
  - no internal structure
- state: one among a collection of discrete state values
  - Ex: find driving routes: {*Trento*, *Rovereto*, *Verona*, ...}

  $\Longrightarrow$ only property: be identical to or different from another state
- very high level of abstraction

  $\Longrightarrow$ lots of details ignored
- The algorithms underlying
  - search and game-playing
  - hidden Markov models
  - Markov decision processes

  all work with atomic representations (or treat it as such)

Representations [cont.]

Factored Representation

- Each state represented in terms of a vector of attribute values
  - Ex: $\langle zone, \{dirty, clean\}\rangle$, $\langle town, speed\rangle$
- State: combination of attribute values
  - Ex: $\langle A, dirty\rangle$, $\langle Trento, 40kmh\rangle$
- Distinct states may share the values of some attribute
  - Ex: $\langle Trento, 40kmh\rangle$ and $\langle Trento, 47kmh\rangle$
  - identical iff all attribute have the same values
    $\Longrightarrow$ must differ for at least one value to be different
- Can represent uncertainty (e.g., ignorance about the amount of gas in the tank represented by leaving that attribute blank)
- Lower level of abstraction $\Longrightarrow$ less details ignored
- Many areas of AI based on factored representations
  - constraint satisfaction and propositional logic
  - planning
  - Bayesian networks
  - (most of) machine learning

Representations [cont.]

Structured Representation

- States represents in terms of objects and relations over them
  - Ex $\forall x.(Men(x) \rightarrow Mortal(x))$,
    $Woman(Maria)$, $Mother \equiv Woman \cap \exists hasChild.Person$
- Lowest level of abstraction $\Longrightarrow$ can represent reality in details
- Many areas of Ai based on factored representations
  - relational databases
  - first-order logic
  - first-order probability models
  - knowledge-based learning
  - natural language understanding