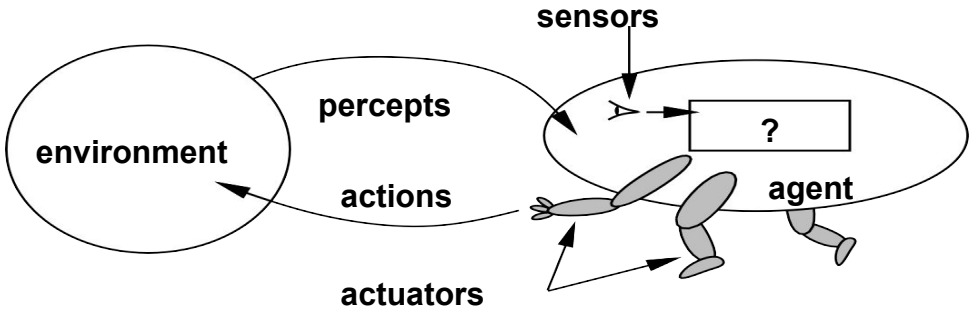
Intelligent Agents

Jacopo Mauro

Slide Based on Slides of the Artificial Intelligence: A Modern Approach book

Agents and environments



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

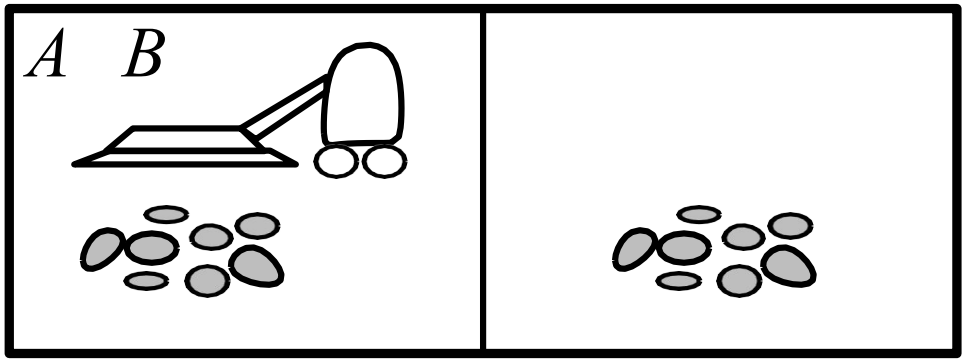
An agent can be anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

The agent function maps from percept histories to actions:

$$f : P* \rightarrow A$$

The agent program runs on the physical architecture to produce f

Vacuum-cleaner world



Percepts: location and contents, e.g., [A, Dirty]

Actions: Left, Right, Suck, NoOp

A vacuum-cleaner agent

Percept sequence	Action
[*A, Clean*]	*Right*
[*A, Dirty*]	*Suck*
[*B, Clean*]	*Left*
[*B, Dirty*]	*Suck*
[*A, Clean*], [*A, Clean*]	*Right*
[*A, Clean*], [*A, Dirty*]	*Suck*
…	…

Rationality

We need: Performance measure to evaluates the environment sequence

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

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date

Rationality

Rational ≠ omniscient

- percepts may not supply all relevant information (rational ≠ clairvoyant)
- action outcomes may not be as expected (rational ≠ successful)

Rational ⇒ exploration, learning, autonomy

PEAS

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

- **Performance measure**
- **Environment**
- **Actuators**
- **Sensors**

PEAS

Agent Type	Performance Measure	Environment	Actuators	Sensors
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
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

Key Concepts in AI Environments

- **Fully/Partially Observable**: agent has access to all/part of relevant information.
- **Deterministic/Non deterministic or Stochastic**: next state is entirely predictable/non predictable from the current state and action.
- **Episodic/Sequential**: actions depends on previous/sequence of actions.
- **Static/Dynamic**: environment does not change/change while the agent is deciding on an action.
- **Discrete/Continuous**: The state and action spaces consist of distinct, countable options/non finite options
- **Single/Multi-agent**: Only one/More than one agent is acting in the environment (competition or collaboration)

Example: Chess?

- **Fully/Partially Observable**?
- **Deterministic/Nondeterministic**?
- **Episodic/Sequential**?
- **Static/Dynamic**?
- **Discrete/Continuous**?
- **Single/Multi-agent**?

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
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

The environment type largely determines the agent design

The real world is (of course) partially observable, non deterministic, sequential, dynamic, continuous, multi-agent

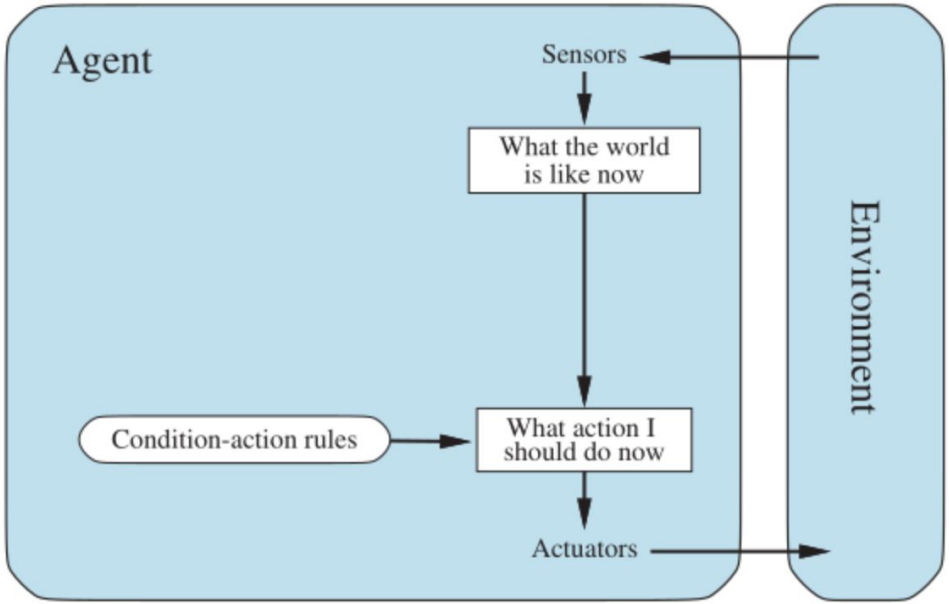
Agent types

agent = architecture + program

Various basic types in order of increasing generality:

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

Simple reflex agents



Selects actions based solely on the current percept, ignoring the history or future consequences

Example

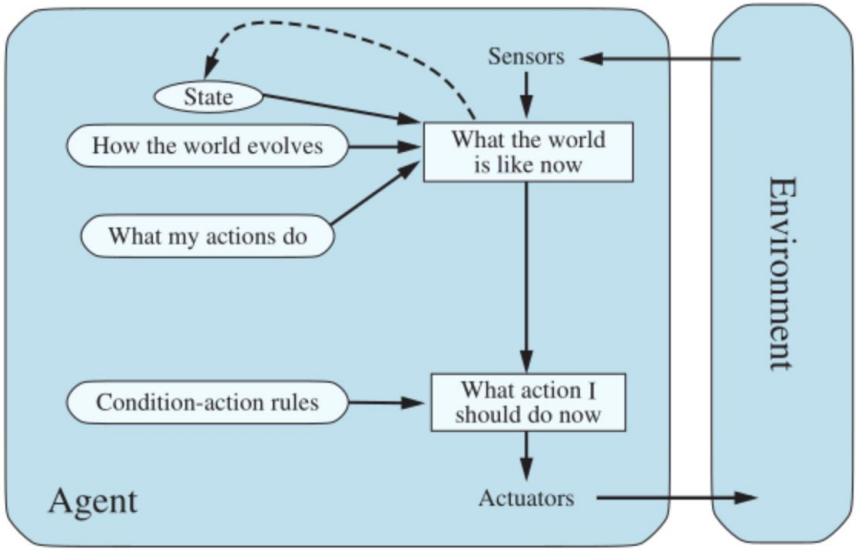
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

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

```
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
```

Problems:

- Use only the current percept to make decisions and this often is not enough
- Requires full observability
- Possible infinite loops (e.g., vacuum cleaner move left and right even if clean)

Model-based agents



Use an internal model of the environment to keep track of unseen aspects and predict the effects of actions. Model represent the best-guess for the of the current environment

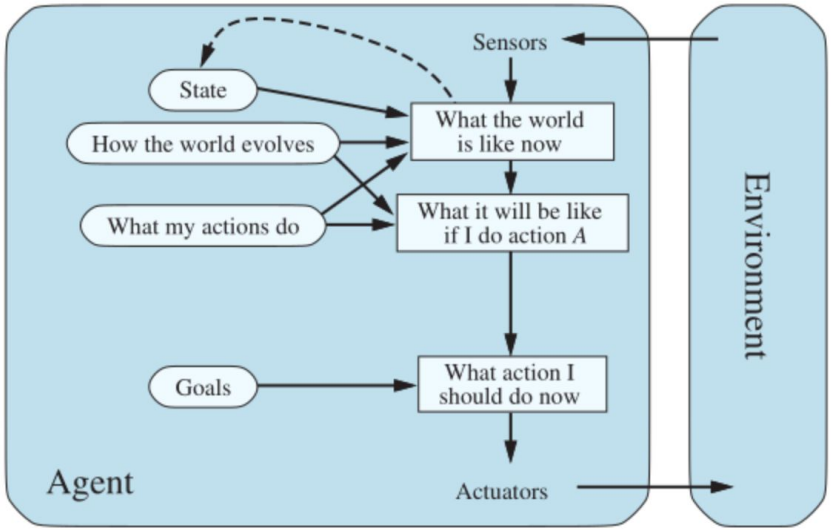
Example

```
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
```

Problems:

- Knowing the state ≠ knowing the goal
- E.g., robot that can explore a maze but does not know its goal

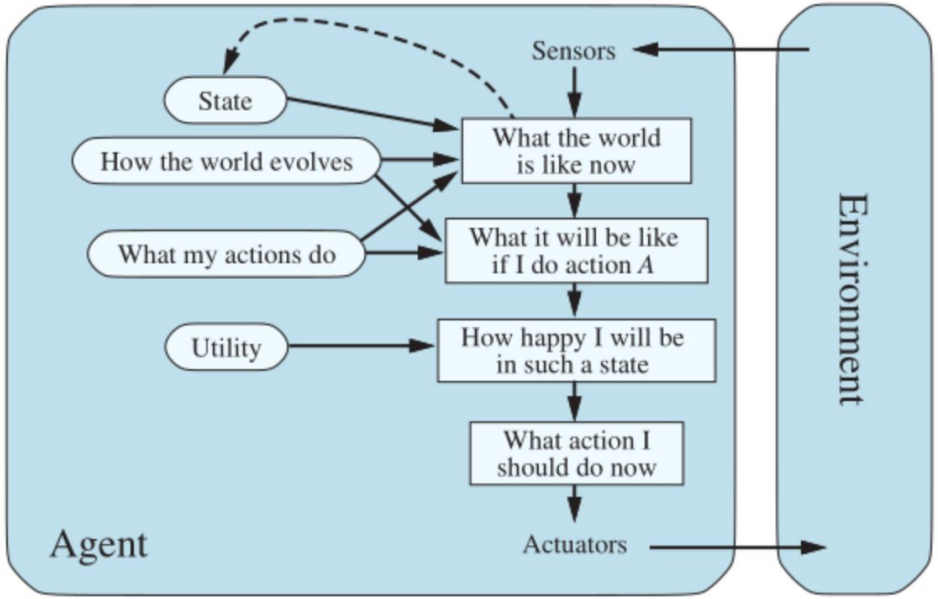
Goal-based agents



Problems:

- Goals may not be enough → cost of the actions matter

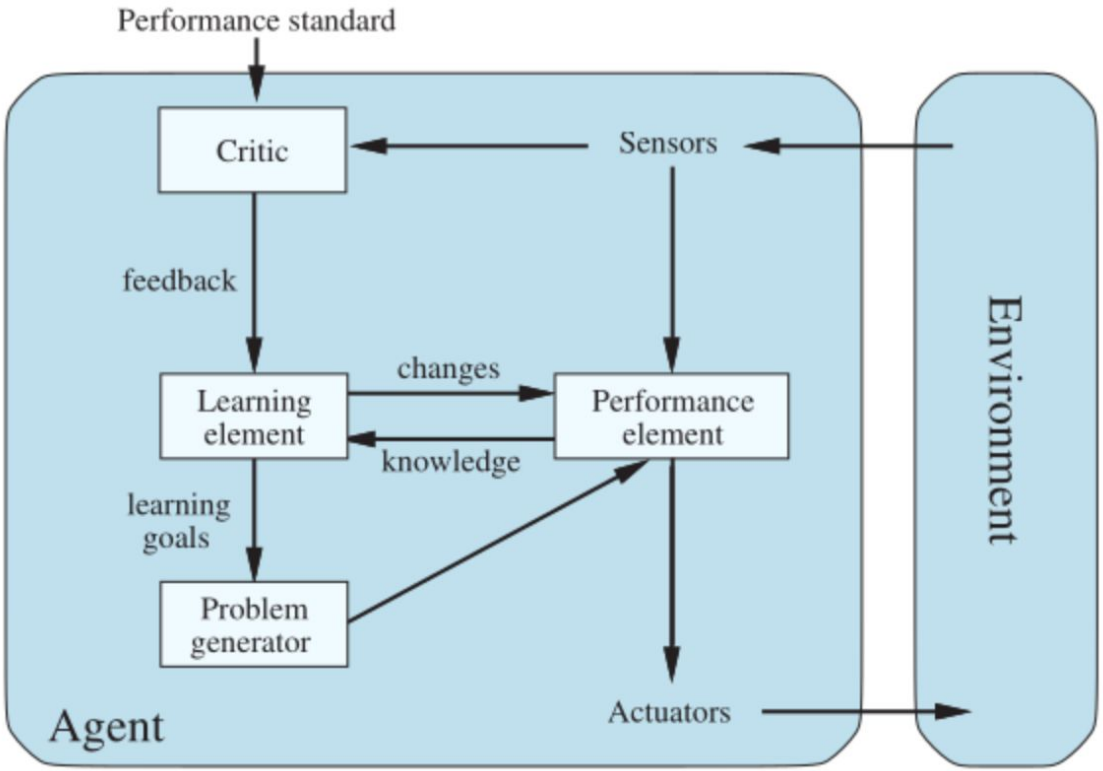
Utility-based agents



Problems:

- defining a precise and comprehensive utility function can be complex and computationally expensive

Learning agents



Learning agents

Improve their performance over time by acquiring knowledge and adapting to the environment.

Key Components:

- Performance Element: Select actions using the current knowledge
- Learning Element: Responsible for making improvements based on feedback.
- Critic: Provides feedback on the agent's actions to guide learning. Suggest how performance element needs to be modified to do better in the future
- Problem Generator: Suggests exploratory actions to discover new knowledge.

Pros

- Adapt to changing environments.
- Operate effectively with incomplete or evolving knowledge.

Cons

- Balancing exploration (learning) and exploitation (acting optimally).
- Managing computational complexity in large environments.

Homework

Read Chapter 2