Intelligent Agents

Outline

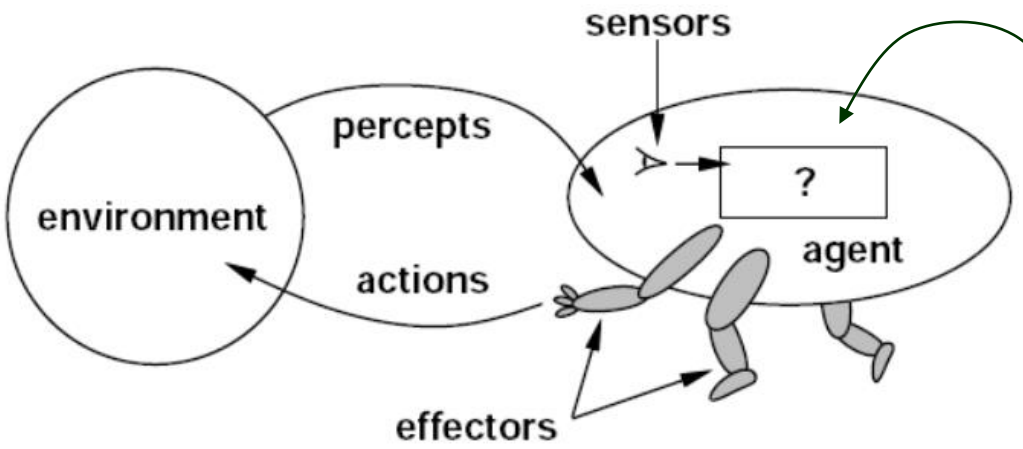
I. Agents and rational behavior

II. The nature of environments

III. The structure of agents

* In part based on notes by Dr. Jin Tian.

I. Agents



- Perception (sensors)
- Reasoning / cognition
- Action (actuators)

Percept: perceptual inputs at any given instant.

Percept sequence: complete history of everything the agent has ever perceived.

Agent function (behavior):

$$\text{a percept sequence} \xrightarrow{\text{reasoning}} \text{an action}$$

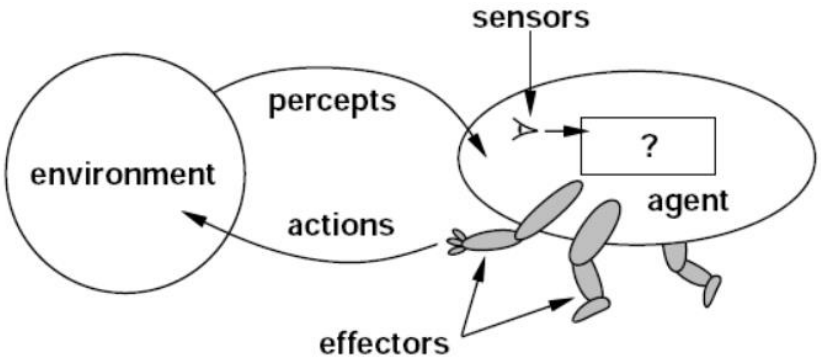
* Illustration from https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/index.html.

Construction of the Agent Function

Tabulation?

- ♠ Very large, if not infinite table!

- ♦ Instead, implement the function internally by an agent program.
- ♦ The program runs on the agent's architecture to produce the function.

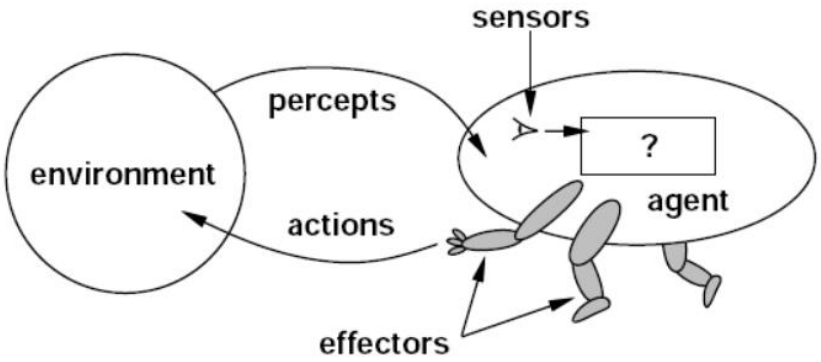


Construction of the Agent Function

Tabulation?

- ♠ Very large, if not infinite table!

- ♦ Instead, implement the function internally by an agent program.
- ♦ The program runs on the agent's architecture to produce the function.



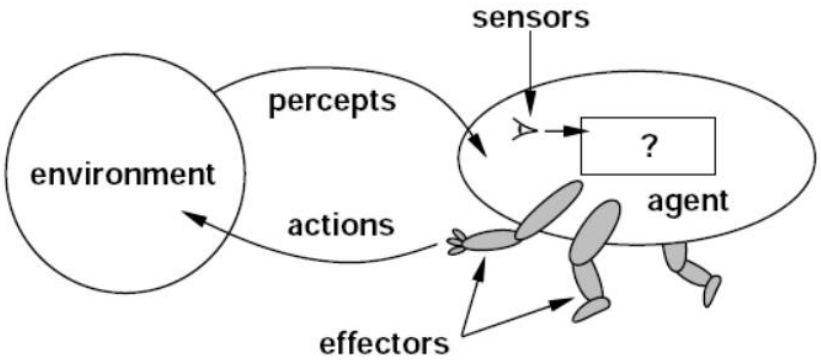
Agent = architecture + program

Construction of the Agent Function

Tabulation?

- ♠ Very large, if not infinite table!

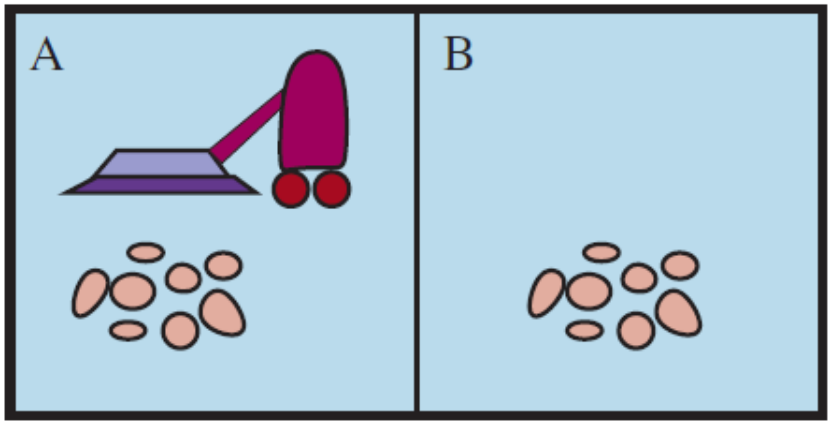
- ♦ Instead, implement the function internally by an agent program.
- ♦ The program runs on the agent's architecture to produce the function.



Agent = architecture + program

- ♦ Abstract description vs concrete implementation!

The Vacuum-Cleaner World



- *Environment*: squares A & B
- *Percepts*: [A, *Dirty*]

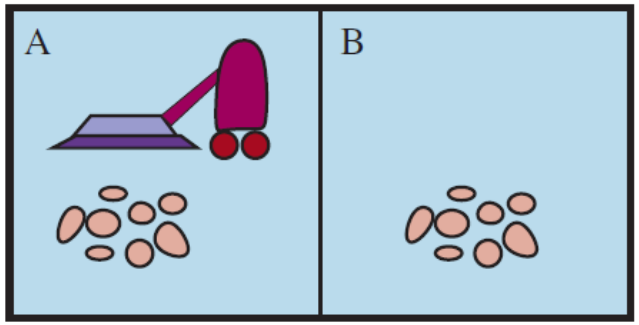
square the vacuum cleaner is in

state of the square

- *Actions*: *left*, *right*, *suck*, *nothing*

Partial Tabulation

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*
⋮	⋮

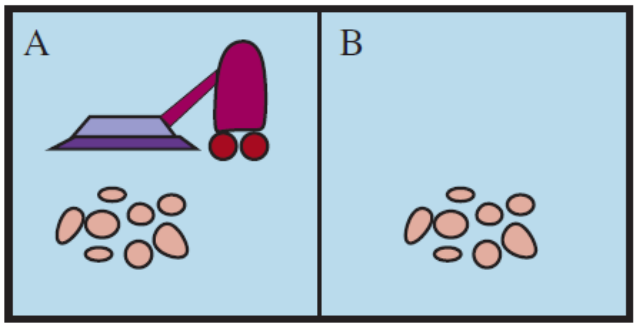


- Many way to fill in the right column
- What is the right way?

⇩

Good/bad, intelligent/stupid?

Rational Behavior?



if *status* == *Dirty* then return *Suck*
else if *location* == *A* then return *Right*
else if *location* == *B* then return *Left*

Is this agent rational?

No, needless oscillation once all the dirt is cleaned up!

⇩ improve

Do nothing when all the squares are clean.

Rationality

What is rational depends on four things:

- performance measure defining the criterion of success
- prior knowledge of the environment
- performable actions by the agent
- perceptual sequence to date

A rational agent should select an action expected to maximize its performance measure.

Performance Measure

- Awards one point for each clean square at each time step.

Meanwhile, assume

- known environment
- unknown dirt distribution and agent's initial location
- only available actions: *Left*, *Right*, and *Suck*
- *Left* and *Right* having no effect if they would take the agent outside
- perfect sensing of location and dirt existence

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*
⋮	⋮

This agent is rational.

Omniscience vs Rationality

- An omniscient agent knows the actual outcome of its actions.

  Impossible in reality!

- Rationality maximizes the expected performance.

  - Learn as much as it perceives.
  - Does not require omniscience.

- Perfection maximizes actual performance.

Rationality ≠ omniscience ≠ perfection

Omniscience vs Rationality

- An omniscient agent knows the actual outcome of its actions.

  Impossible in reality!

- Rationality maximizes the expected performance.
  - Learn as much as it perceives.
  - Does not require omniscience.
- Perfection maximizes actual performance.

Rationality ≠ omniscience ≠ perfection

II. Task Environment

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

- performance measure
- environment of the agent
- agent's actuators and sensors

} PEAS

Automated Taxi Driver

Its task environment in the PEAS description:

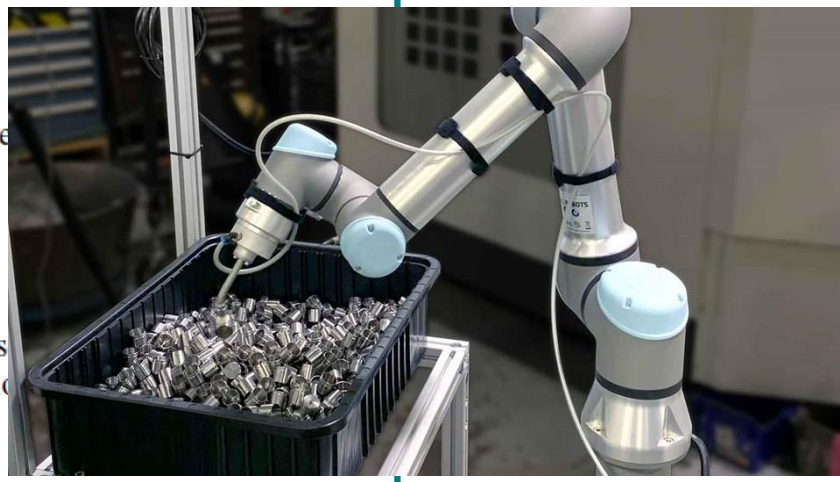
Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

PEAS for Other Agents

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

PEAS for Other Agents

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, te diagnoses, treatments	
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of s categorizati	
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



Universal Robots ActiNav autonomous bin picking kit

Environment Properties

- Categorize task environments according to properties.

⇩

appropriate families of techniques
for agent implementation

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle Chess with a clock						
Poker Backgammon						
Taxi driving Medical diagnosis						
Image analysis Part-picking robot						
Refinery controller English tutor						

Environment Property 1

- Fully observable if the sensors can detect all aspects that are *relevant* to the choice of action.
  vs. partially observable

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

Environment Property 2

- Single-agent
  vs. multiagent

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single				
Chess with a clock	Fully	Multi ← competitive				
Poker	Partially	Multi				
Backgammon	Fully	Multi				
Taxi driving	Partially	Multi ← cooperative				
Medical diagnosis	Partially	Single				
Image analysis	Fully	Single				
Part-picking robot	Partially	Single				
Refinery controller	Partially	Single				
English tutor	Partially	Multi				

Environment Property 3

- Deterministic vs. stochastic

if the next state of the environment is completely determined by the current state and the action executed by the agent.
otherwise.

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

unable to keep track of all the cards in opponents' hands; must be treated as nondeterministic

Environment Property 4

- Episodic
  vs. sequential

if the agent's experience is divided into atomic episodes, among which one does not depend on the actions taken in previous ones.

if the current decision could affect all future decisions.

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

instantaneous actions can have long-term consequences.

Environment Property 5

- Dynamic if the environment can change while the agent is choosing an action.
  vs. semidynamic if the environment changes under the agent's action only.
  vs. static otherwise.

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

Environment Property 6

- Discrete vs. continuous

The distinction applies to

- the environment's state
- the way time is handled
- the agent's percepts and actions

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

Environment Property 6

- Discrete vs. continuous

The distinction applies to

- the environment's state
- the way time is handled
- the agent's percepts and actions

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

III. The Structure of Agents

- The job of AI is to design an agent program that implements

$$\text{percepts} \longmapsto \text{action}$$

$$\textit{agent} = \textit{architecture} + \textit{program}$$

Computing device, sensors & actuators

- All agent programs have the same skeleton:
  - input: current percepts
  - output: action
  - program: manipulates inputs to produce output

Table Lookup Agent

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*

It retains complete percept sequence in memory.

Doomed to **failure** due to

- daunting table size (e.g., easily over 10^{150} entries for chess)
- no storage space
- no time for construction
- no way for the agent to learn all the entries
- no guidance on how to fill the table entries

Basic Agent Types

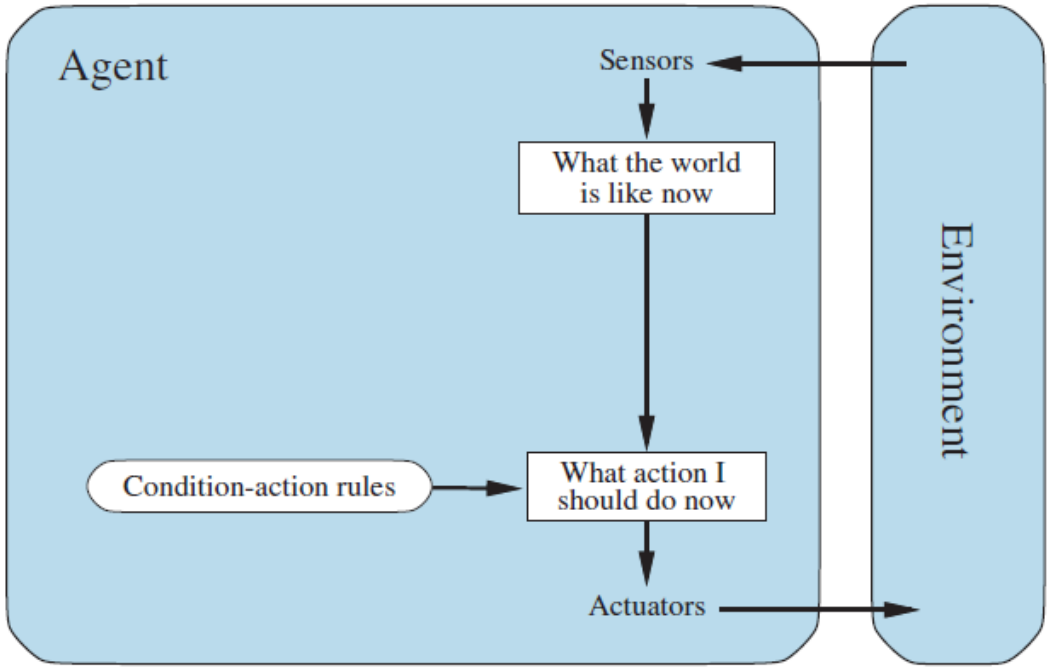
Four basic types embody the principles underlying almost all intelligent systems:

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

All of them can be converted into

- Learning-based agents

Simple Reflex Agent



Rectangles: agent's current internal state
Ovals: background information used in the process.

- Select actions based on the current percepts, and ignore the percept history.

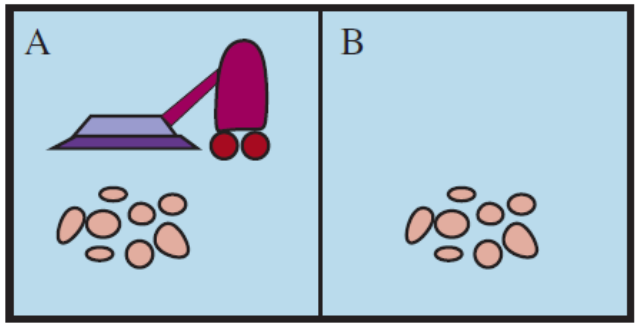
  E.g., the vacuum agent

- Implemented through *condition-action* rule.

  `if` dirty `then` suck

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

Vacuum-Cleaner World (Revisited)



if *status* == *Dirty* then return *Suck*
else if *location* == *A* then return *Right*
else if *location* == *B* then return *Left*

Simple Reflex Agent

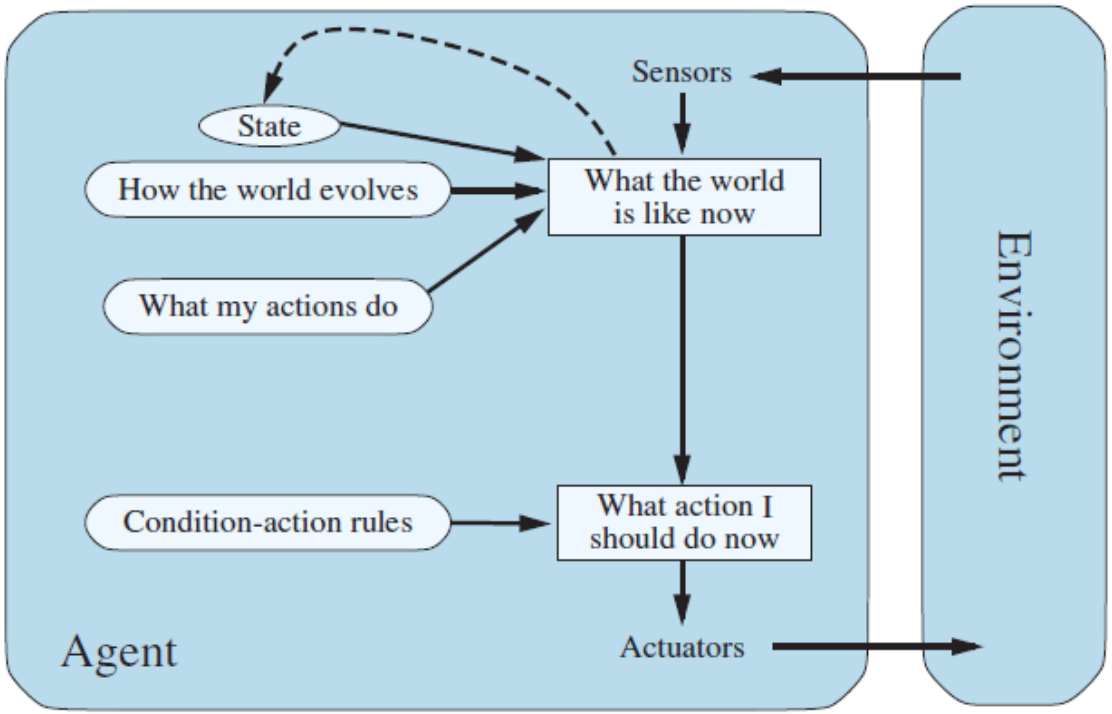
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

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

- Limited intelligence

  It will work **only if** the correct decision can be made based on only the current percept, i.e., **only if** the environment is fully observable.

Model-based Reflex Agent



- Partially observable environment.
- Need to maintain some internal state.
- Update it using *knowledge*.

⇩

Model of the world

  - How does the world change?
  - How do actions affect the world?

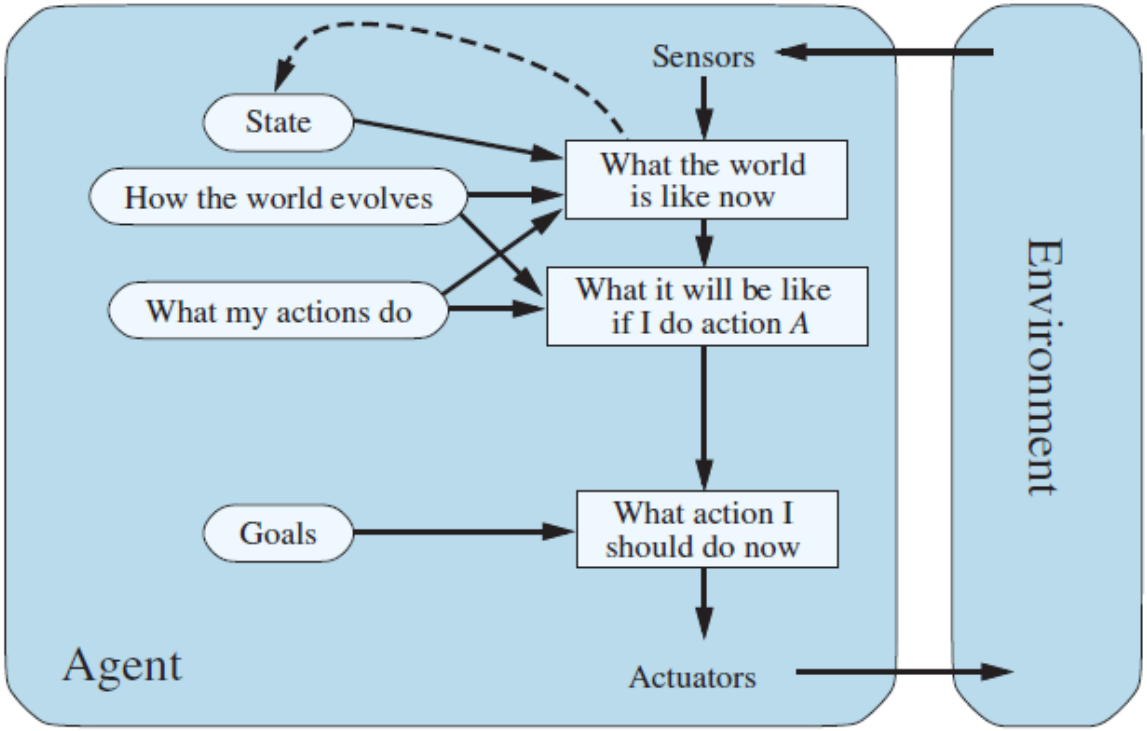
How This Agent Works

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*

- It is rarely possible to describe the exact current state of the environment.
- The maintained "state" does not have to describe the world.

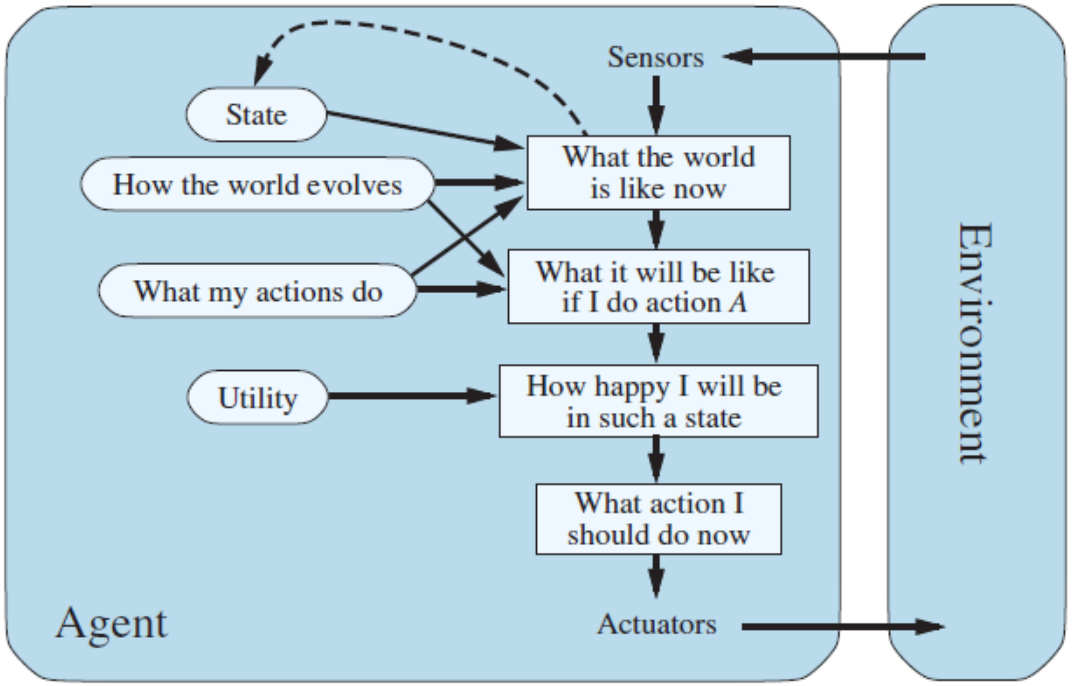
Goal-Based Agent



- Needs also some goal information describing desirable situations.
- *Search* and *planning*

  when a long sequence of actions is required to find the goal.
- Difference in *taking the future into account*.

Utility-Based Agent



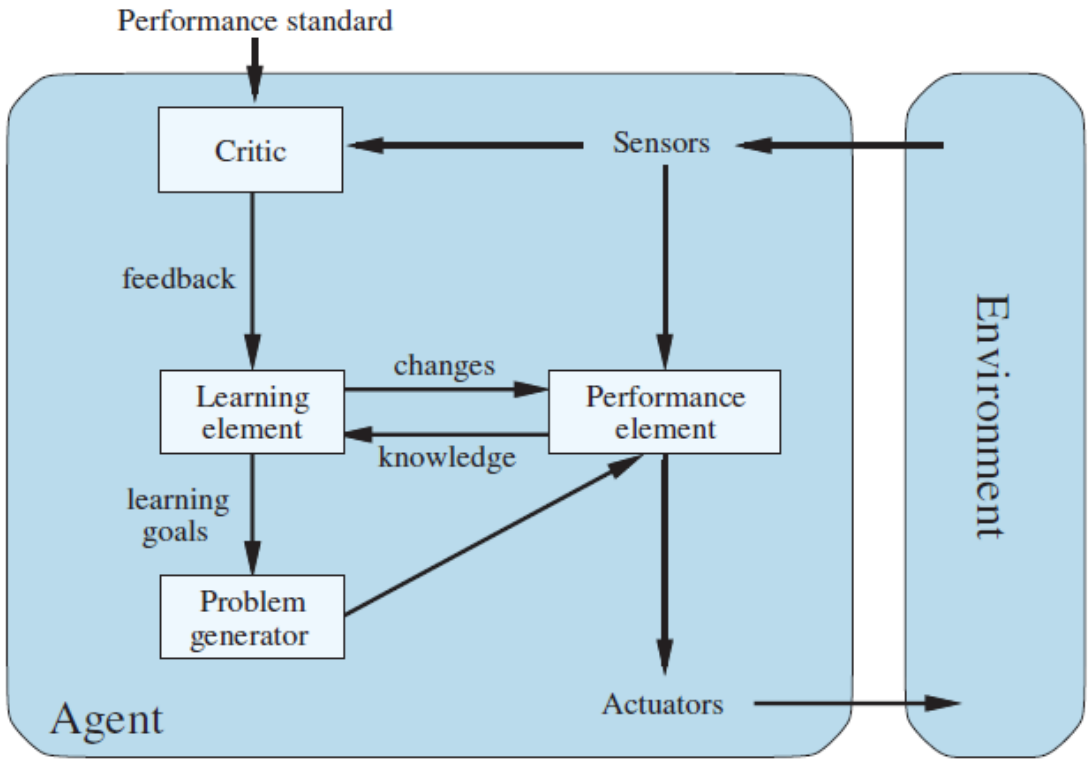
- Different ways to achieve a goal sometimes.
- Use a *utility function* that maps a (sequence of states) to a real number (*utility*)

  ↑

  internal performance measure
- Maximize expected utility.

- Goal improvements:
  - selection among conflicting goals
  - selection based on likelihood of success and goal importance

Learning-Based Agent

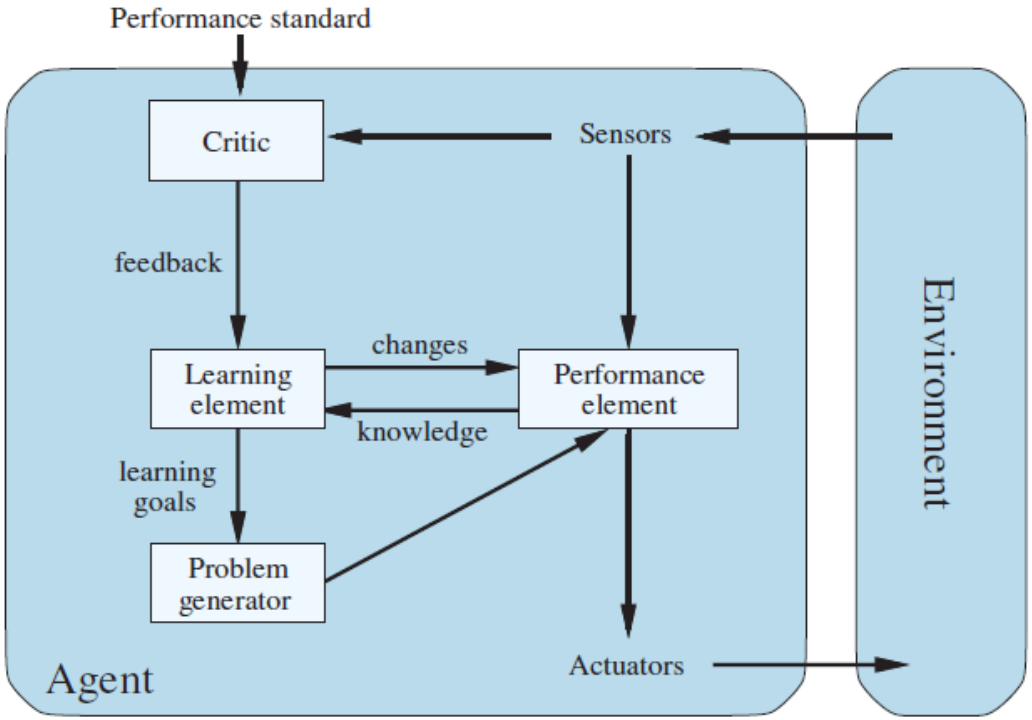


- Preferred method for creating state-of-the-art AI systems:
  - ♦ Allow operation in initially unknown environments.
  - ♦ Adapt to changes in the environment --- robustness.
- Modifications of the four components to bring them in closer agreement with the available feedback

⇩

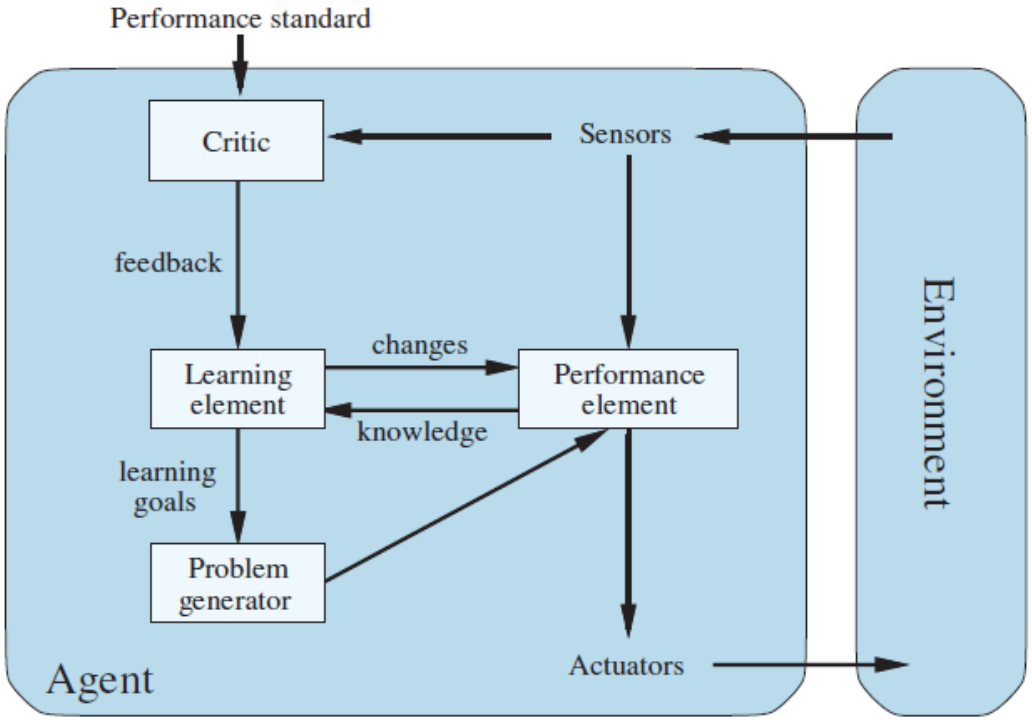
Better overall performance

Learning-Based Agent



- *Learning element* introduces improvements in performance element.
- *Critic* provides feedback on the agent's performance based on fixed performance standard.
- *Performance element* selects actions based on the precepts.
- *Problem generator* suggests actions that will lead to new and informative experiences.

Learning-Based Agent



- *Learning element* introduces improvements in performance element.
- *Critic* provides feedback on the agent's performance based on fixed performance standard.
- *Performance element* selects actions based on the precepts.
- *Problem generator* suggests actions that will lead to new and informative experiences.