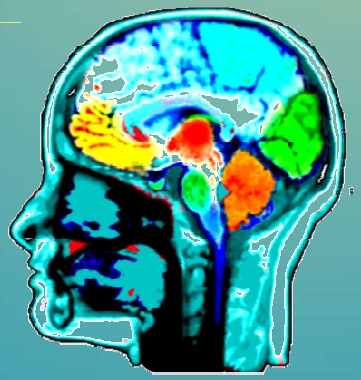
Introduction
To
Artificial Intelligence

Isfahan University of Technology (IUT)
1402



Rational Agents

Dr. Hamidreza Hakim
hamid.hakim.u@gmail.com

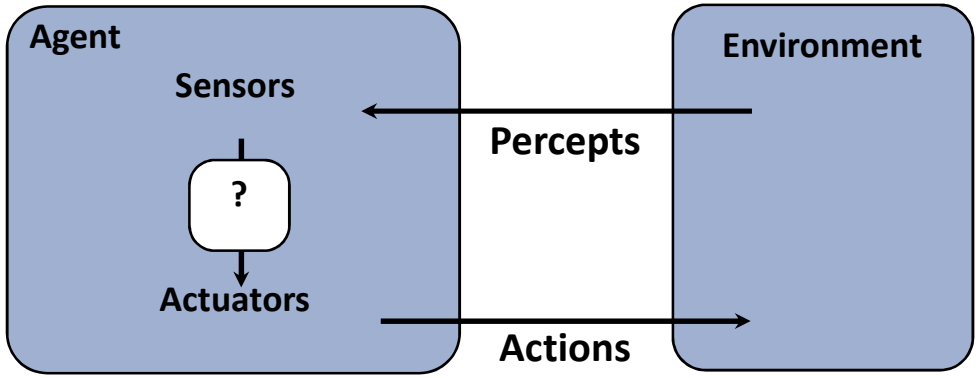
[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.]


بسم الله الرحمن الرحيم

Outline

- Agent
- Rationality
- PEAS
- Environment Type
- Agent Type

Agents and environments



- An agent ***perceives*** its environment through ***sensors*** and ***acts*** upon it through ***actuators***
- The ***agent function*** maps percept sequences to actions
- It is generated by an ***agent program*** running on a ***machine***

- ***Sample : Human as Agent***

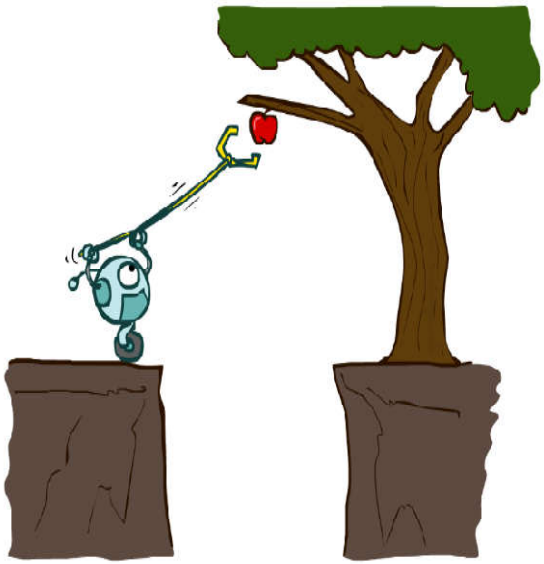
عامل توسط حسگرها درک رو از محیط می گیره و اینجا ینی ؟ باید تصمیم بگیره که در مقابل این درک چه کاری باید انجام بشه و به اقدامگرها فرمان داده میشه که یک عملی رو روی محیط انجام بده --> این میشه عامل

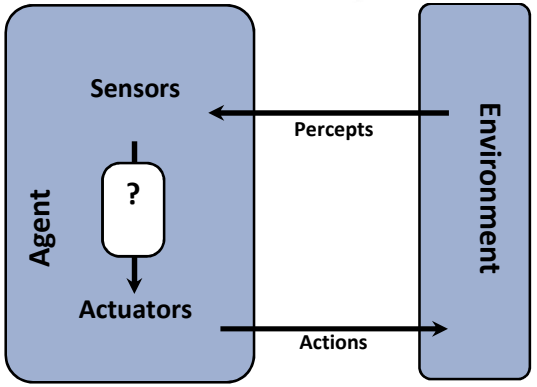
Rational Agent

- Abstractly, an agent is a function from percept histories to actions:

$$f: \mathcal{P}^* \rightarrow \mathcal{A}$$

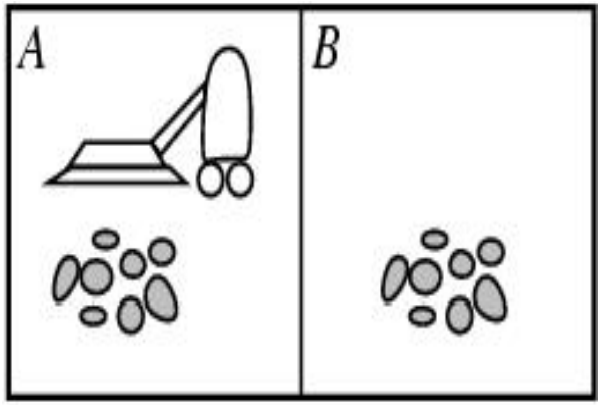
- For any given class of environments and tasks, we seek the
  - agent (or class of agents) with the best (expected) performance (or utility)
- Caveat: computational limitations make perfect rationality unachievable
  - design best program for given machine resources





تابع: f
درک: p
مجموعه دنباله درک: *p
به مجموعه اعمال : A

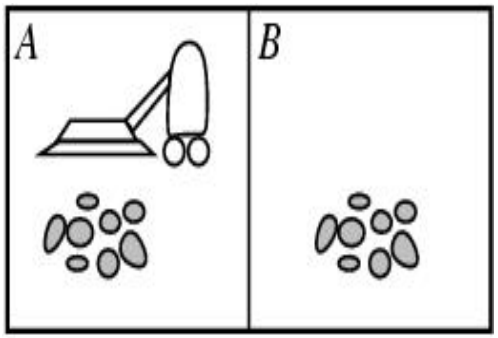
The vacuum-cleaner world



- Environment: square A and B
- Percepts: [location and content] e.g. *[A, Dirty]*
- Actions: left, right, suck, and no-op

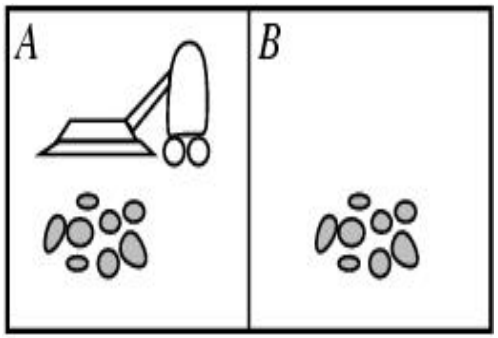
The vacuum-cleaner world

Q: what is agent function?



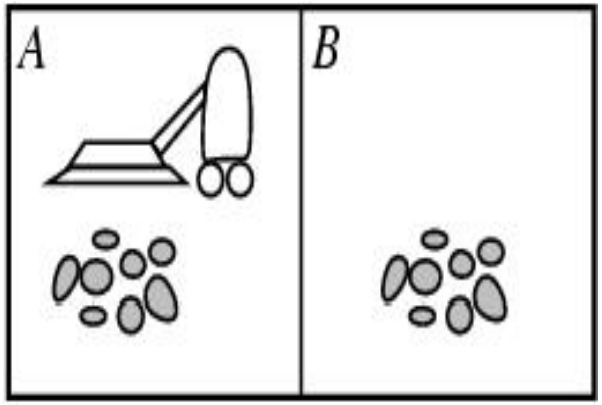
Percept sequence	Action
[A,Clean]	
[A, Dirty]	
[B, Clean]	
[B, Dirty]	
[A, Clean],[A, Clean]	
[A, Clean],[A, Dirty]	
…	

The vacuum-cleaner world



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
…	…

The vacuum-cleaner world



function REFLEX-VACUUM-AGENT ([*location, status*]) return an action
    if *status* == *Dirty* then return *Suck*
    else if *location* == *A* then return *Right*
    else if *location* == *B* then return *Left*

Q: Is this agent rational?
Loop:no Op

دنیای جارو
یک دنیایی که فقط دوتا سلول داره
عامل یک ربات جارو کننده است که می تونه مشکل بکنه و زباله ها رو برداره
توی این دوتا سلول می تونه زباله باشه یا نباشه
ادراکات این عامل : مکانی هست که توش قرار گرفته و محتویات اون سلول مثلا مکان ربات توی سلول A است و اون سلول تمیز است
اعمالی که این ربات می تونه انجام بده: از پی دی اف بخون

جدول توی اون پی دی اف: از دنباله درک می ریم به عمل اینو حواست باشه

Rational agents

- A rational agent should

select an action

that is expected

to maximize its performance measure,

given the evidence provided by:

- its percept sequence,
- the built-in knowledge it has.

- E.g., the performance measure of a vacuum-cleaner agent could include the amount of:
  - dirt cleaned up,
  - time taken,
  - electricity consumed,
  - noise generated, etc.

What is the world?

Modeling the world

- To design a rational agent, we must

  specify the **task environment**

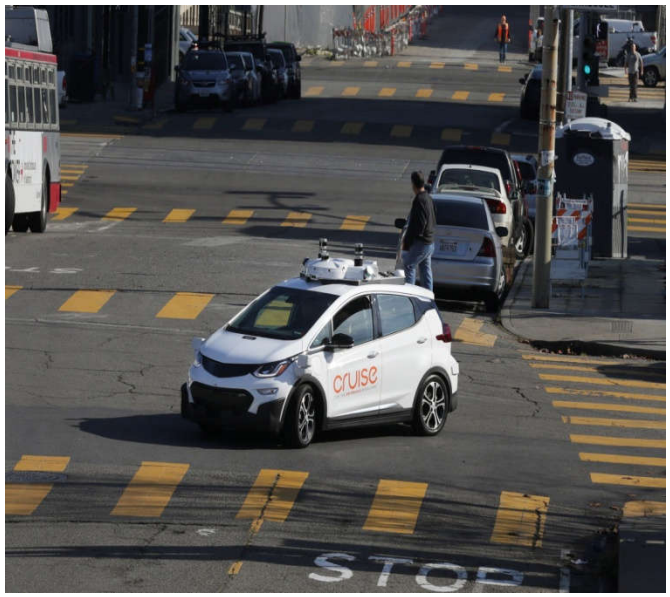
  – Also known as PEAS (e.g. in automated taxi agent):

  Performance measure (sometimes with constraints)

  Environment

  Actuators

  Sensors



PEAS: Automated taxi

- Performance measure
  - Income, happy customer, vehicle costs, fines, insurance premiums
- Environment
  - streets, other drivers, customers, weather, police…
- Actuators
  - Steering, brake, gas, display/speaker
- Sensors
  - Camera, radar, accelerometer, engine sensors, microphone, GPS

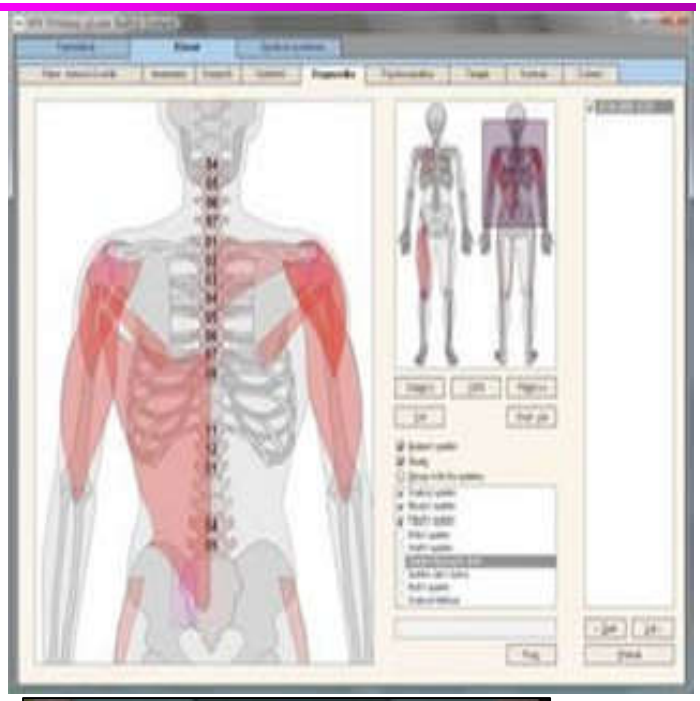
Image: http://nypost.com/2014/06/21/how-google-might-put-taxi-drivers-out-of-business/

PEAS: Pacman

- Performance measure
  - -1 per step; + 10 food; +500 win; -500 die; +200 hit scared ghost
- Environment
  - Pacman dynamics (include ghost behavior)
- Actuators
  - Left Right Up Down or NSEW
- Sensors
  - Entire state is visible (except power pellet duration)

PEAS: Medical diagnosis system

- Performance measure
  - Patient health, cost, reputation
- Environment
  - Patients, medical staff, insurers, courts
- Actuators
  - Keyboard/mouse
- Sensors
  - Screen display, email

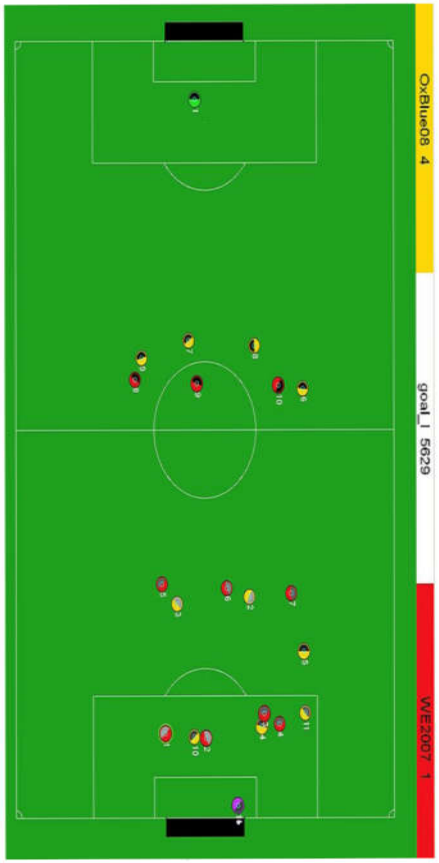


Types of environments

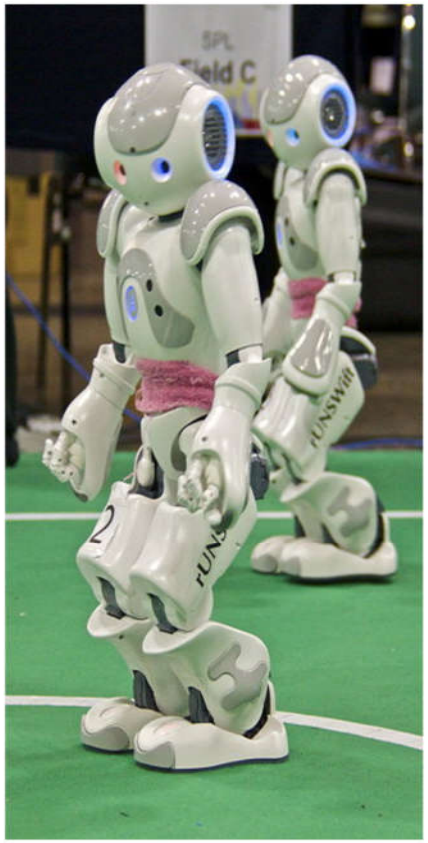
design rational agents

Fully observable vs. Partially observable

An environment is *fully observable* when the sensors can detect all aspects that are *relevant* to the choice of action.



VS.



Single agent vs. Multi agent

Does the environment contain other agents who are also maximizing some performance measure that depends on the current agent's actions?

VS.

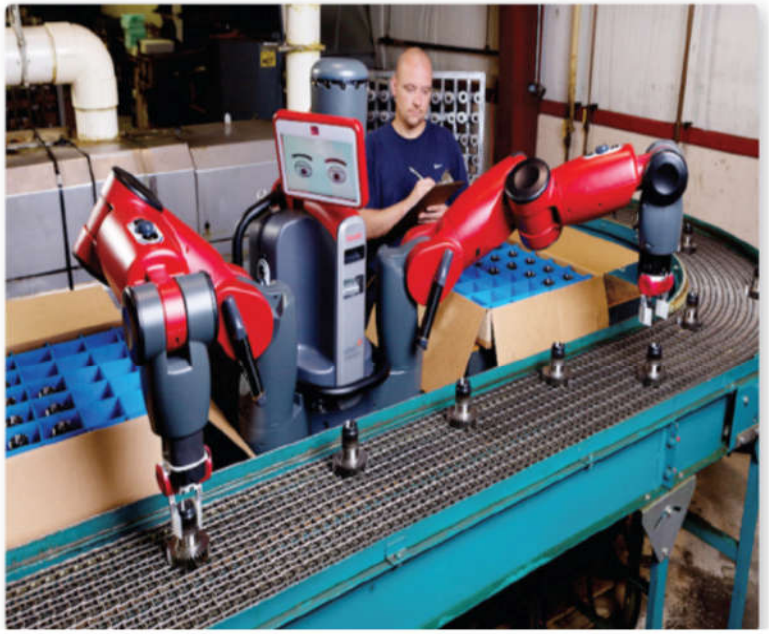
Deterministic vs. Stochastic

if the next environment state is completely determined by the current state the executed action then the environment is deterministic.

VS.

Episodic vs. Sequential

In an episodic environment, the agent's experience is divided into atomic episodes. The choice of action depends only on the episode itself

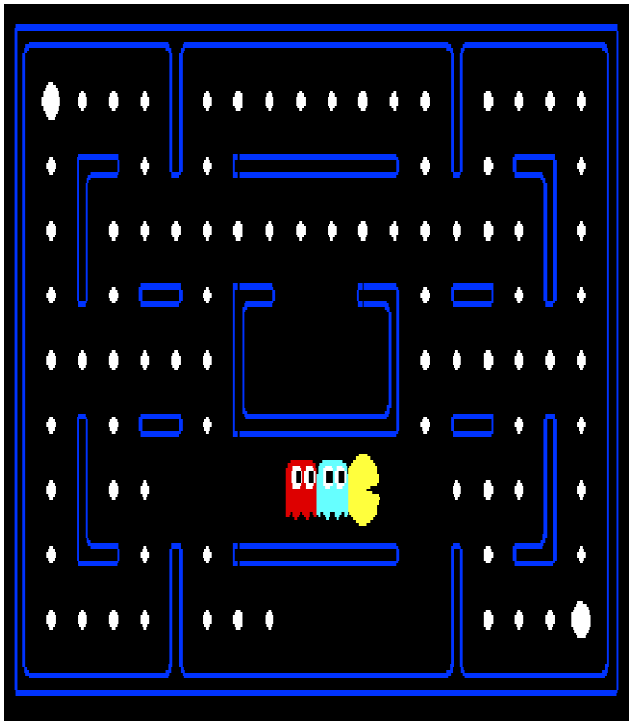


VS.

Static vs. Dynamic

If the environment can change while the agent is choosing an action, the environment is dynamic. Semi-dynamic if the agent's performance changes even when the environment remains the same.

vs.



Discrete vs. Continuous

This distinction can be applied to
the state of the environment,
the way time is handled and to the percepts/actions of the agent.

vs.

Known vs. Unknown

This distinction refers to the agent's knowledge about the "*laws of physics*" of the environment, rather than referring to the environment. If the environment is unknown, the agent will have to learn how it works in order to make good decisions

VS.

Mars Exploration Rover (MER)

Environment types

Task Environment	Chess	Backgammon	Med. Diagnosis	Taxi Driving
Fully Observable?	Yes	Yes	No	No
Single Agent?	No	No	Yes	No
Deterministic?	Yes	No	No	No
Episodic	No	No	No	No
Static?	Yes/Semi	Yes	No	No
Discrete?	Yes	Yes	No	No
Known?	Yes	Yes	No	No

Environment types

- The simplest environment is
  - Fully observable, single-agent, deterministic, episodic, static, discrete and known.
- But, most real situations are:
  - Partially observable, multi-agent, stochastic, sequential, dynamic, continuous and unknown.

Agent design

- **The environment type largely determines the agent design**
  - ***Partially observable*** => agent requires ***memory*** (internal state)
  - ***Multi-agent*** => agent may need to behave ***randomly***
  - ***Nondeterministic*** => agent may have to prepare for ***contingencies***
  - ***Static*** => agent has enough time to compute a rational decision
  - ***Continuous time*** => continuously operating ***controller(preprocessing)***
  - ***Unknown physics*** => need for ***exploration***
  - ***Unknown perf. measure*** => observe/interact with ***human principal***

The Structure of Agents

design rational agents

Agent types

Function TABLE-DRIVEN_AGENT(*percept*) **returns** an action

static: *percepts*, a sequence initially empty

table, a table of actions, indexed by percept sequence

append *percept* to the end of *percepts*

action ← LOOKUP(*percepts*, *table*)

return *action*

This approach is doomed to failure

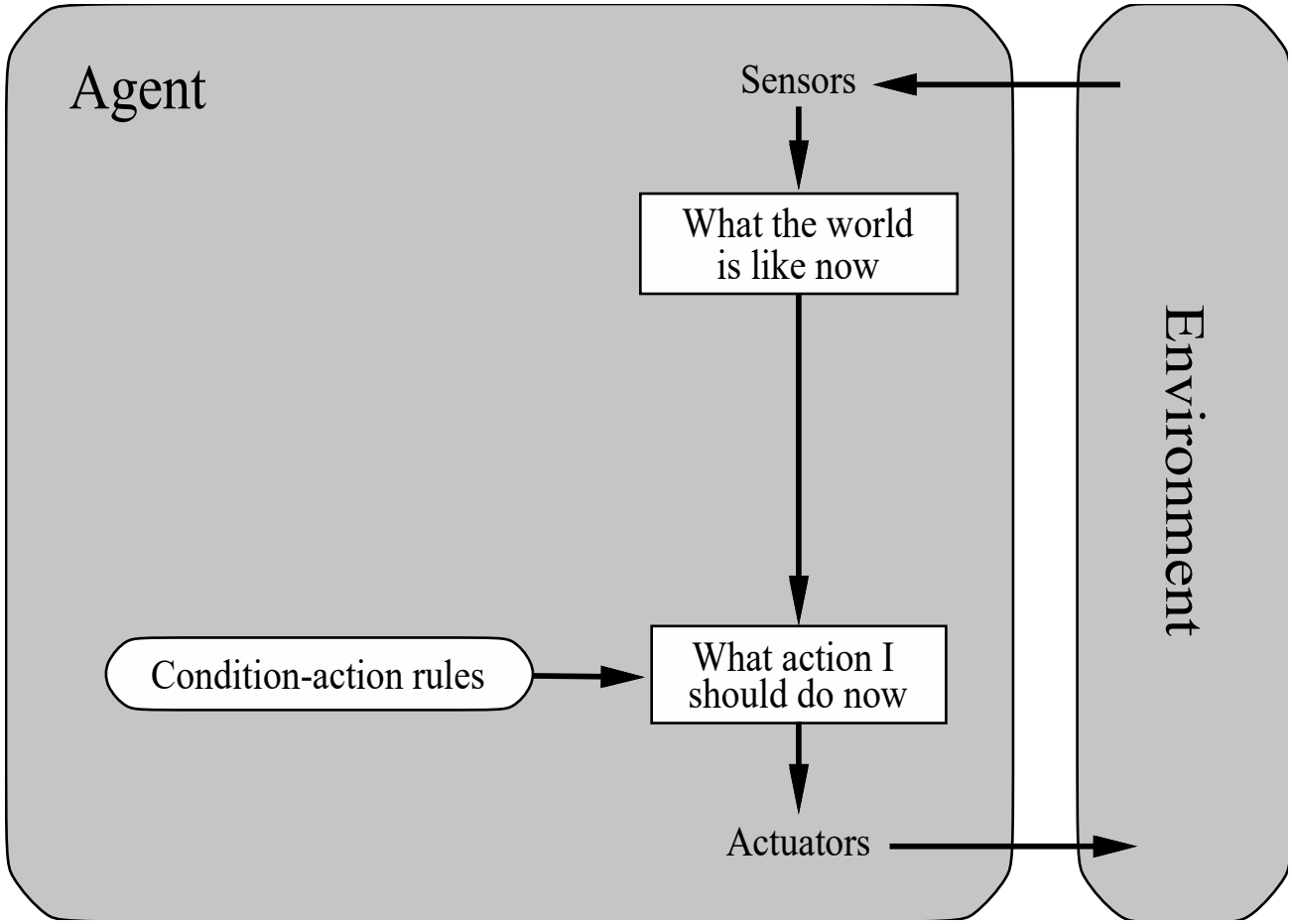
Table-lookup agent

- Drawbacks:
  - Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries

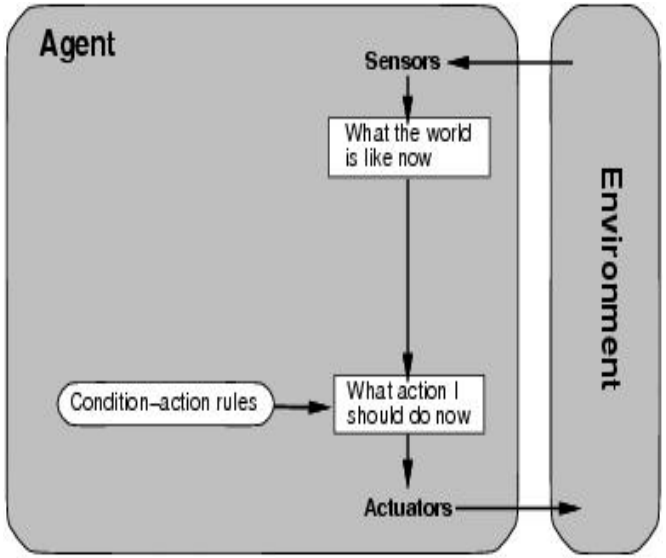
Agent types

- Four basic kind of agent programs will be discussed:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
- All these can be turned into learning agents.

Simple reflex agents

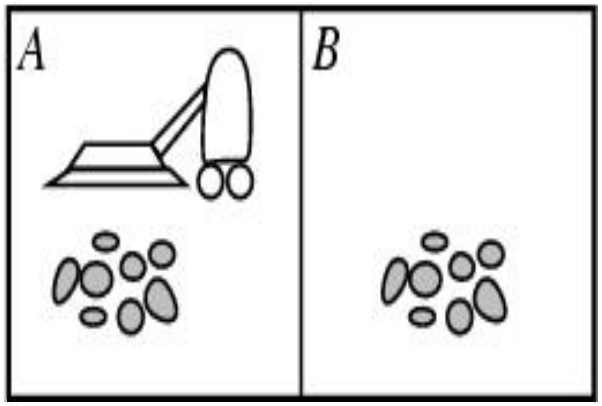


Simple reflex agents (Cont.)



- Select action on the basis of *only the current* percept.
  - E.g. the vacuum-agent
- Large reduction in possible percept/action situations (next page).
- Implemented through *condition-action rules*
  - If dirty then suck

Simple reflex agents (Cont.)



function REFLEX-VACUUM-AGENT ([*location, status*]) return an action

if *status* == *Dirty* then return *Suck*

else if *location* == *A* then return *Right*

else if *location* == *B* then return *Left*

Reduction from 4^T to 4 entries

Simple reflex agents (Cont.)

- Simple reflex behaviors occur even in more complex environments.
- E.g., 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.
- Based on the visual input, the condition “The car in front is braking” is recognized.
- This triggers some connected reflex in the agent program to “initiate braking.”
- Such a connection is called a condition–action rule

Simple reflex agents (Cont.)

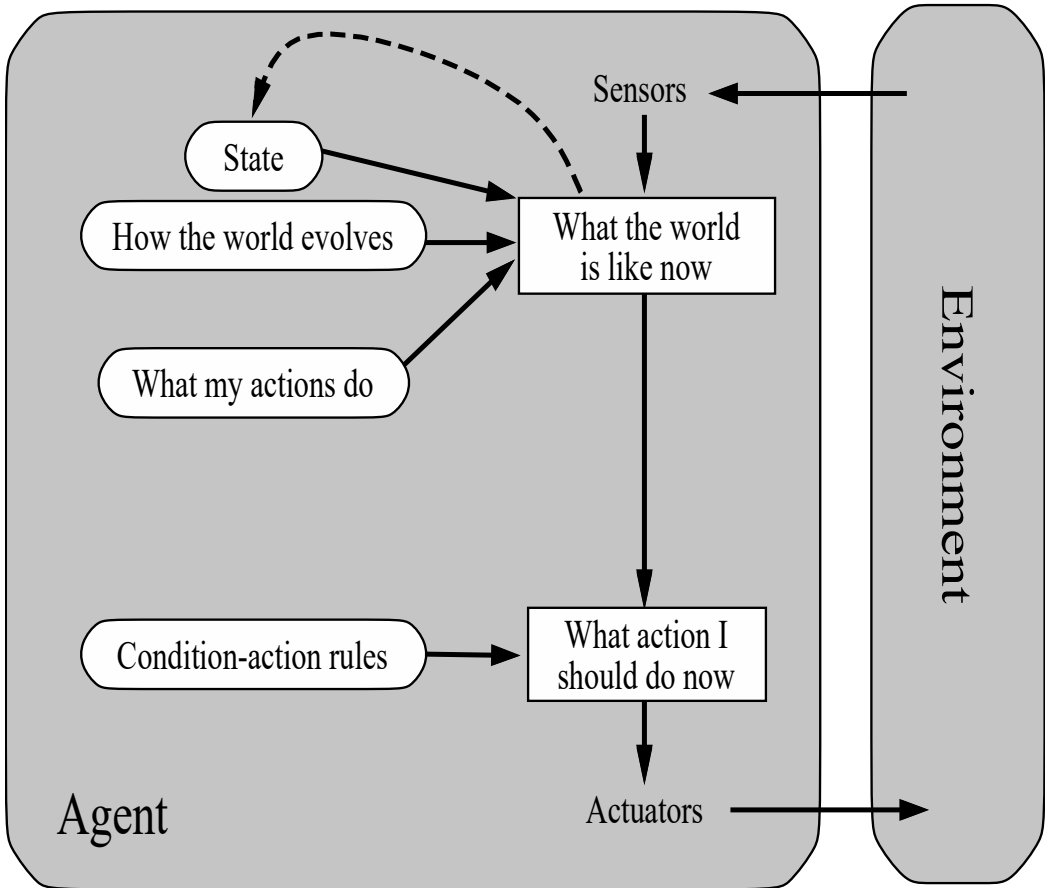
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

static: *rules*, a set of condition-action rules

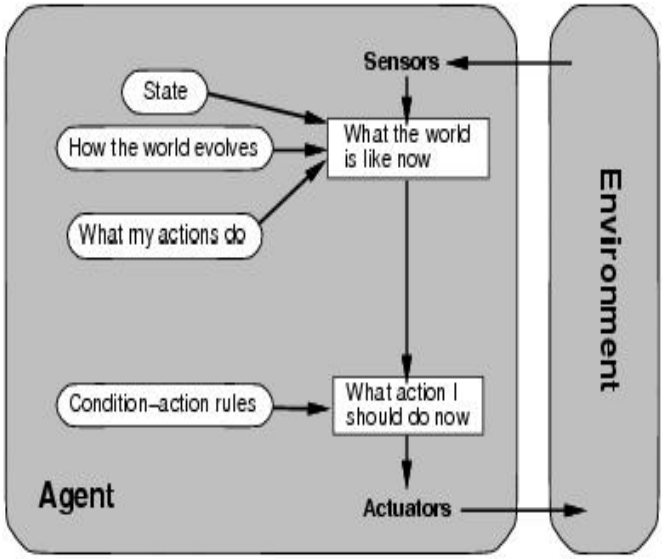
state ← INTERPRET-INPUT(*percept*)
rule ← RULE-MATCH(*state*, *rule*)
action ← RULE-ACTION[*rule*]
return *action*

Will only work if the environment is *fully observable* otherwise infinite loops may occur.

Model-Based Reflex agents



Model-Based Reflex agents (Cont.)



- To tackle *partially observable* environments.
  - Maintain internal state
- Over time update state using world knowledge
  - How does the world change.
  - How do actions affect world.
  - how the agent perceives the world

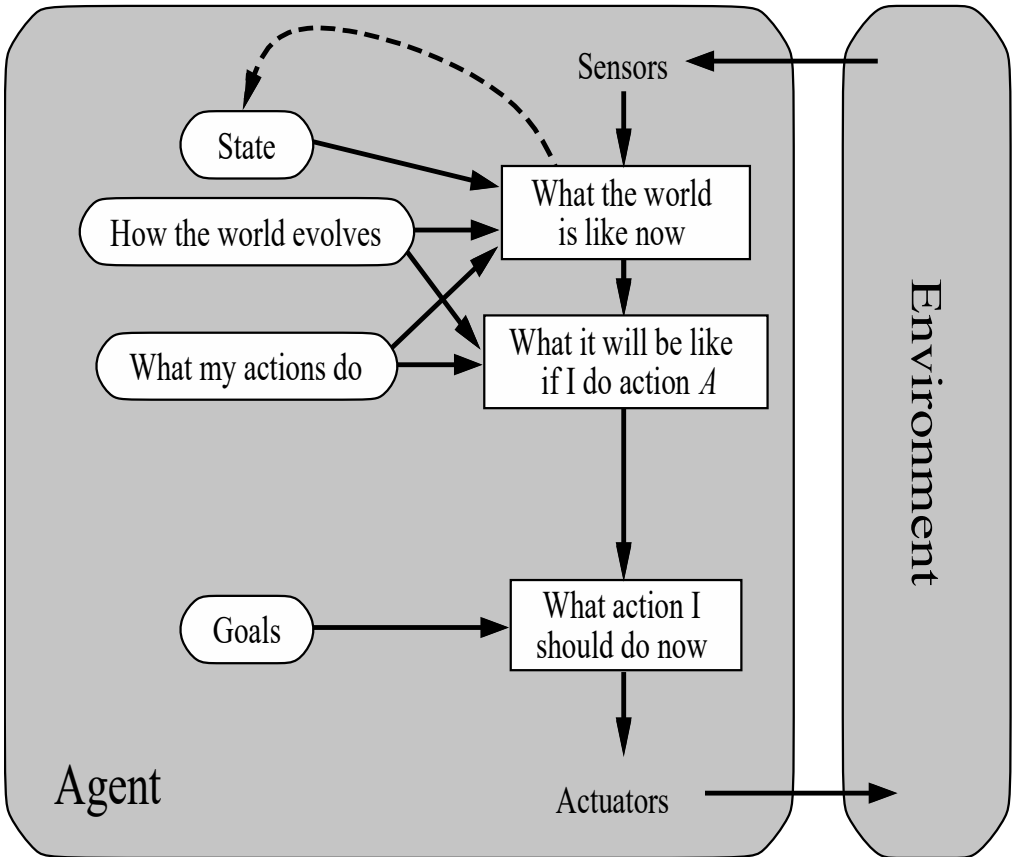
  $\Rightarrow$ *Model of World*

Model-Based Reflex agents (Cont.)

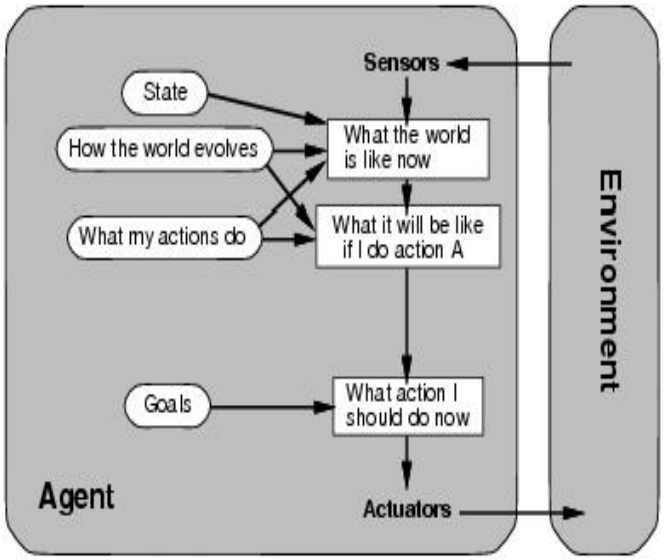
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              transition_model, a description of how the next state depends on
                    the current state and action
              sensor_model, a description 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
```

Goal-based agents

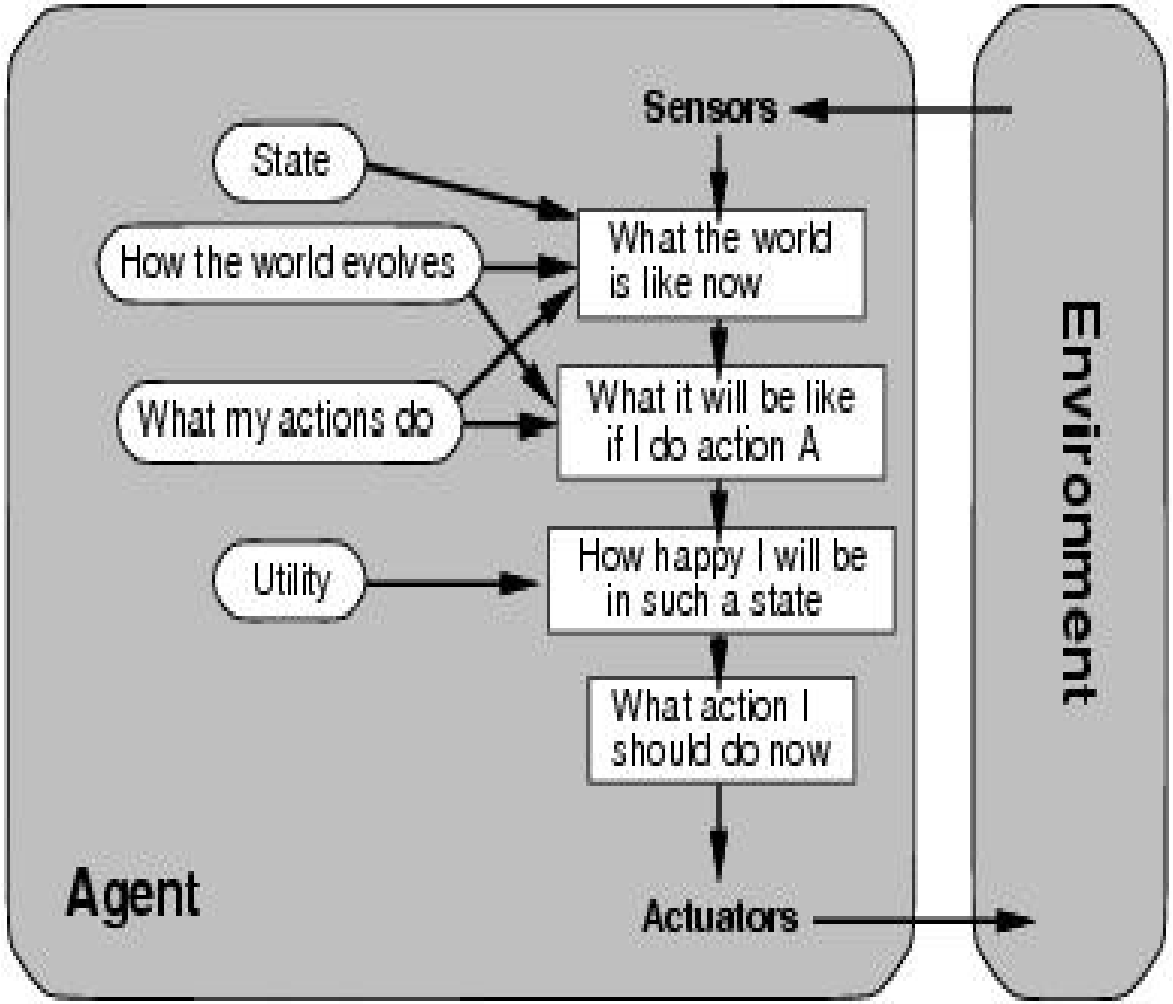


Goal-based agents (Cont.)

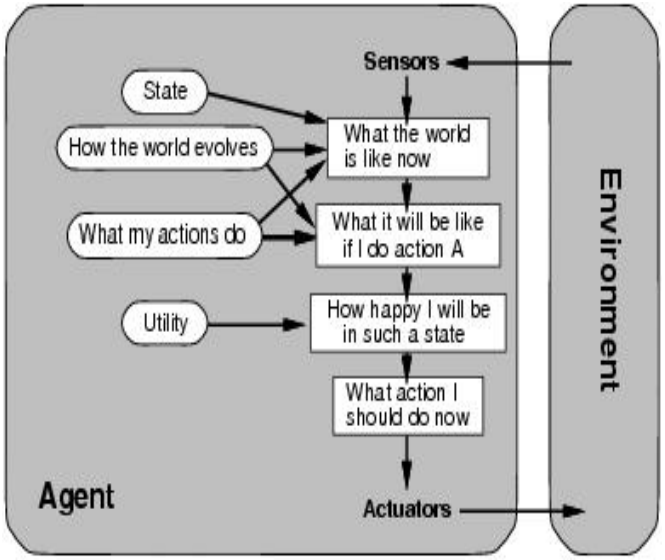


- The agent needs a goal to know which situations are *desirable*.
  - Things become difficult when long sequences of actions are required to find the goal.
- Typically investigated in **search** and **planning** research.
- Major difference: future is taken into account

Utility-based agents

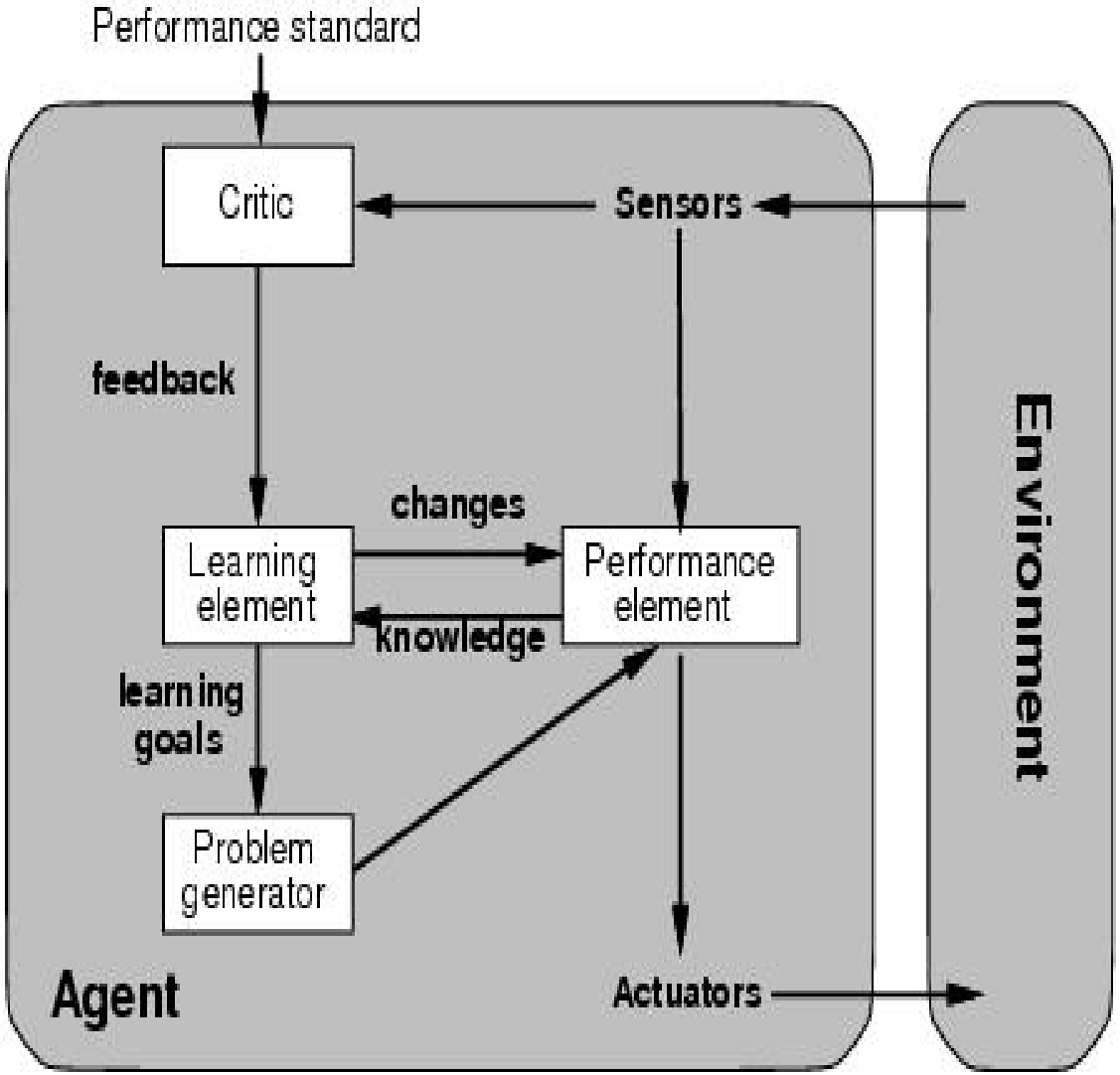


Utility-based agents (Cont.)

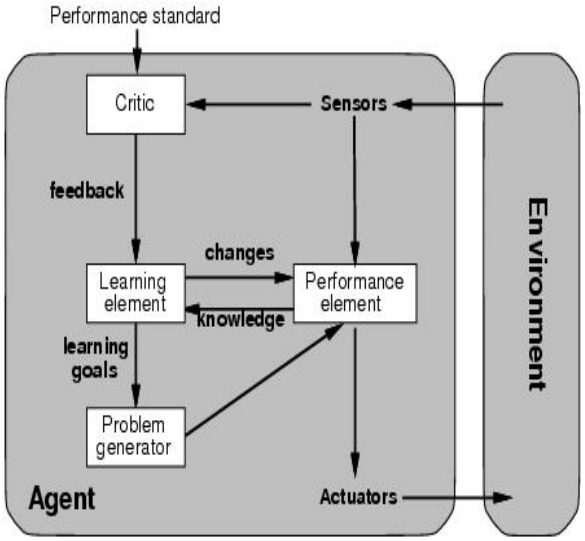


- Certain goals can be reached in different ways.
  - Some are better, have a higher utility.
- Utility function maps a (sequence of) state(s) onto a real number.
- Improves on goals:
  - Selecting between conflicting goals
  - Select appropriately between several goals based on likelihood of success.

Learning agents

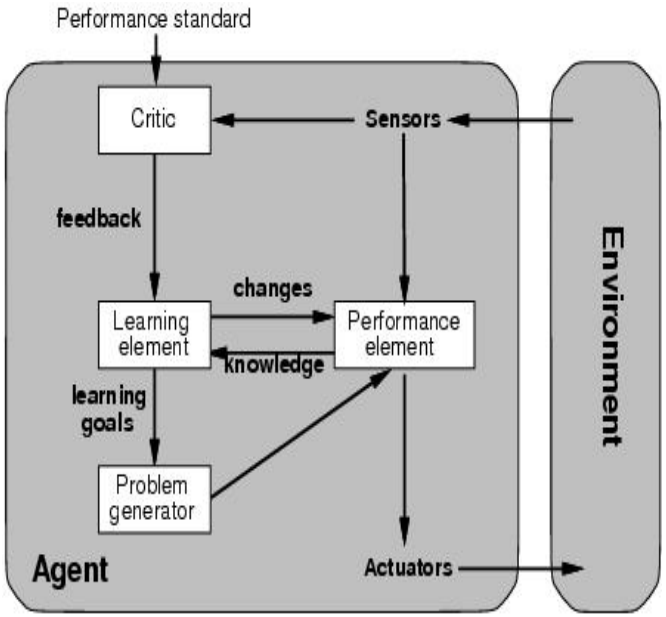


Learning agents (Cont.)



- All previous agent-programs describe methods for selecting *actions*.
  - Yet it does not explain the origin of these programs.
  - Learning mechanisms can be used to perform this task.
  - Teach them instead of instructing them.
  - Advantage is the robustness of the program toward initially **unknown** environments.

Learning agents (Cont.)



- *Learning element*: introduce improvements in performance element.
  - Critic provides feedback on agents performance based on fixed performance standard.
- *Performance element*: selecting actions based on percepts.
  - Corresponds to the previous agent programs
- *Problem generator*: suggests actions that will lead to new and informative experiences.
  - Exploration vs. exploitation

Spectrum of representations