Intelligent Agents

Reading: Russell's Chapters 2

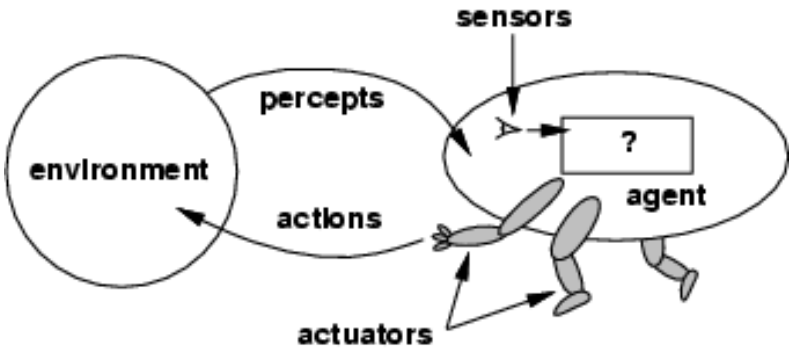
Agents

An agent is anything that can perceive its environment through sensors and act upon that environment through actuators

Human agent: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators

Robotic agent: camera and microphone for sensors; various motors for actuators

Agents and environments

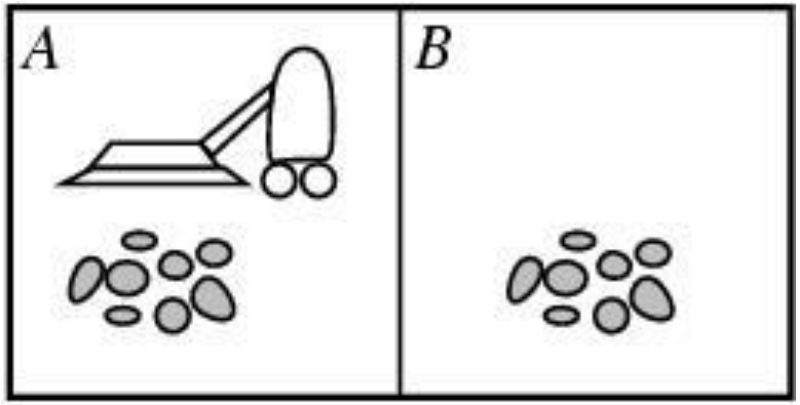


The agent function maps from percept histories to actions:

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

The agent program runs on the physical architecture to produce *f*

The vacuum-cleaner world

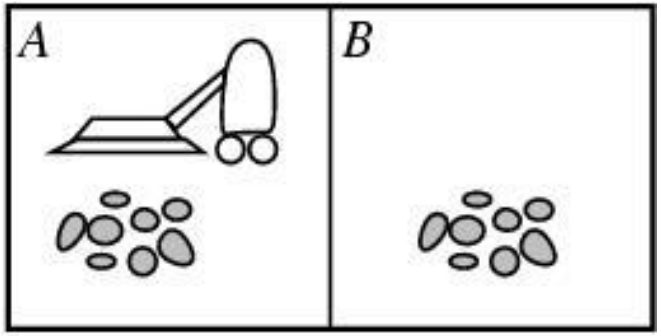


Environment: square A and B

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

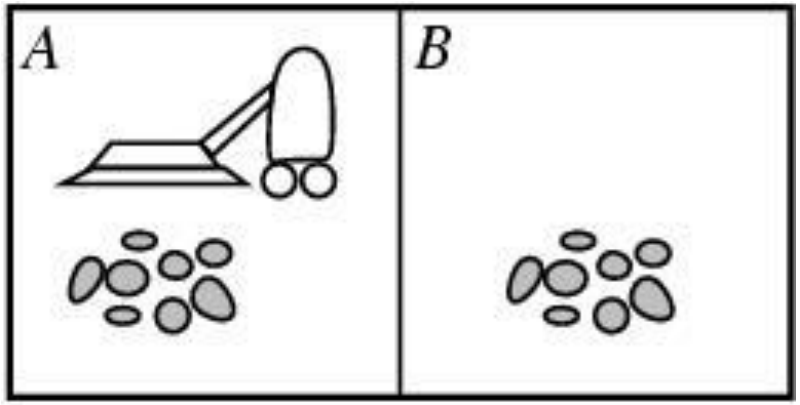
Actions: left, right, suck, and no-op

The vacuum-cleaner world



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*

Rational agents

For each possible percept sequence, a rational agent 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.

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

Environments

To design an agent we must specify its task environment.

PEAS description of the task environment:

- Performance
- Environment
- Actuators
- Sensors

PEAS

Consider, e.g., the task of designing an automated taxi driver:

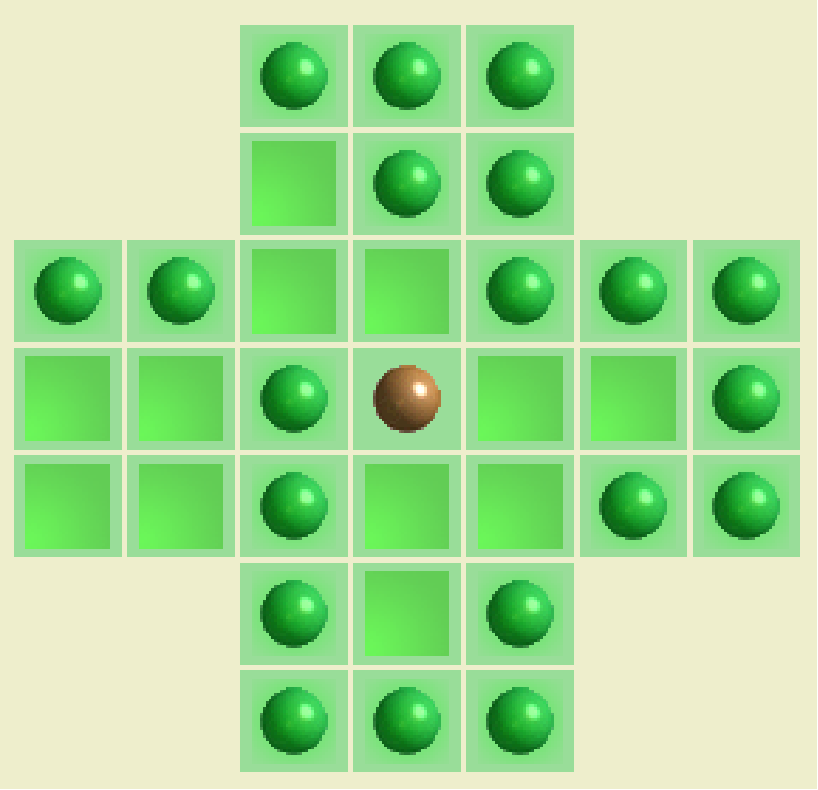
- Performance measure: Safe, fast, comfortable, maximize profits
- Environment: Roads, pedestrians, customers
- Actuators: Steering wheel, accelerator, brake, signal, horn
- Sensors: Cameras, sonar, speedometer, GPS, engine sensors

Environment types

	Solitaire	Backgammom	Chess	Taxi
Observable??				
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

Peg Solitaire



Environment types

Fully vs. partially observable: an environment is full observable when the sensors can detect all aspects that are *relevant* to the choice of action.

	Solitaire	Backgammon	Chess	Taxi
Observable??				
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

Fully vs. partially observable: an environment is full observable when the sensors can detect all aspects that are *relevant* to the choice of action.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

Deterministic vs. stochastic: if the next environment state is completely determined by the current state the executed action then the environment is deterministic.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

Deterministic vs. stochastic: if the next environment state is completely determined by the current state the executed action then the environment is deterministic.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

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

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

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

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??				
Discrete??				
Single-agent??				

Environment types

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.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??				
Discrete??				
Single-agent??				

Environment types

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.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??	YES	YES	YES(SEMI)	NO
Discrete??				
Single-agent??				

Environment types

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.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??	YES	YES	YES(SEMI)	NO
Discrete??				
Single-agent??				

Environment types

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.

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??	YES	YES	YES(SEMI)	NO
Discrete??	YES	YES	YES	NO
Single-agent??				

Environment types

Single vs. multi-agent: Does the environment contain other agents who are also maximizing some performance measure that depends on the current agent's actions?

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??	YES	YES	YES(SEMI)	NO
Discrete??	YES	YES	YES	NO
Single-agent??				

Environment types

Single vs. multi-agent: Does the environment contain other agents who are also maximizing some performance measure that depends on the current agent's actions?

	Solitaire	Backgammon	Chess	Taxi
Observable??	FULL	FULL	FULL	PARTIAL
Deterministic??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??	YES	YES	YES(SEMI)	NO
Discrete??	YES	YES	YES	NO
Single-agent??	YES	NO	NO	NO

Environment types

The simplest environment is

– Fully observable, deterministic, episodic, static, discrete and single-agent.

Most real situations are:

– Partially observable, stochastic, sequential, dynamic, continuous and multi-agent.

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

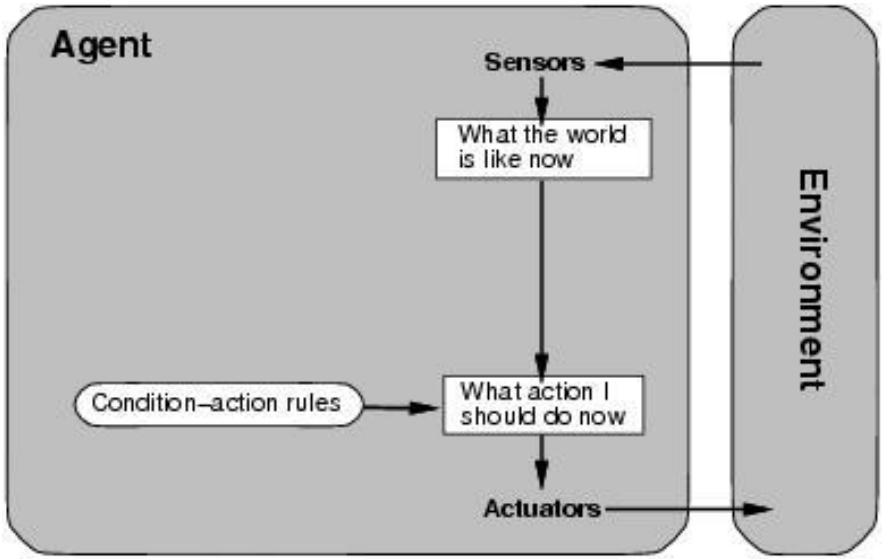
Agent types

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.

Agent types; simple reflex



Select action on the basis of *only the current* percept.

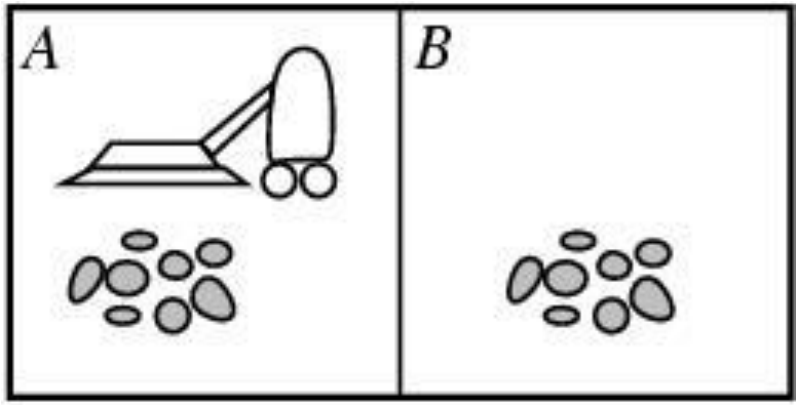
- E.g. the vacuum-agent

Large reduction in possible percept/action situations(next page).

Implemented through *condition-action rules*

- If dirty then suck

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*

Reduction from 4^T to 4 entries

Agent types; simple reflex

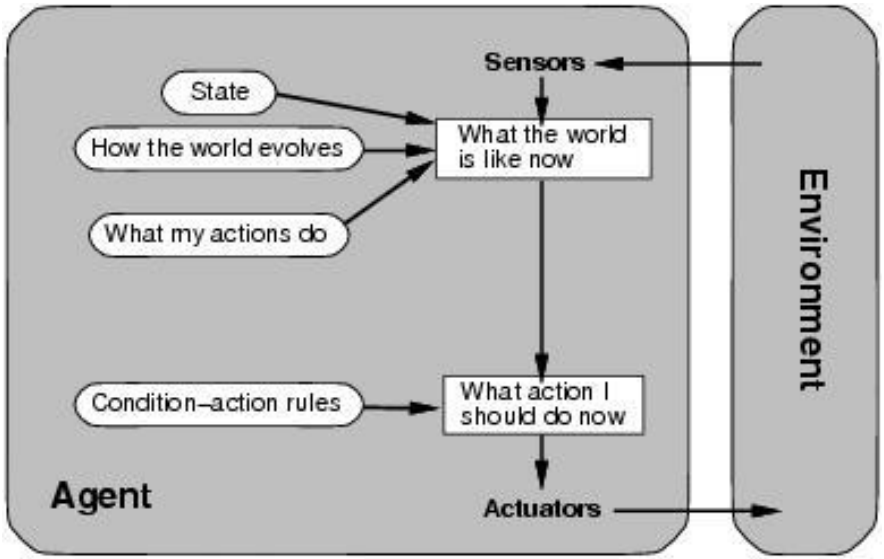
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

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.

Agent types; reflex and state



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.

$\Rightarrow$ *Model of World*

Agent types; reflex and state

function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action

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

state, a description of the current world state

action, the most recent action.

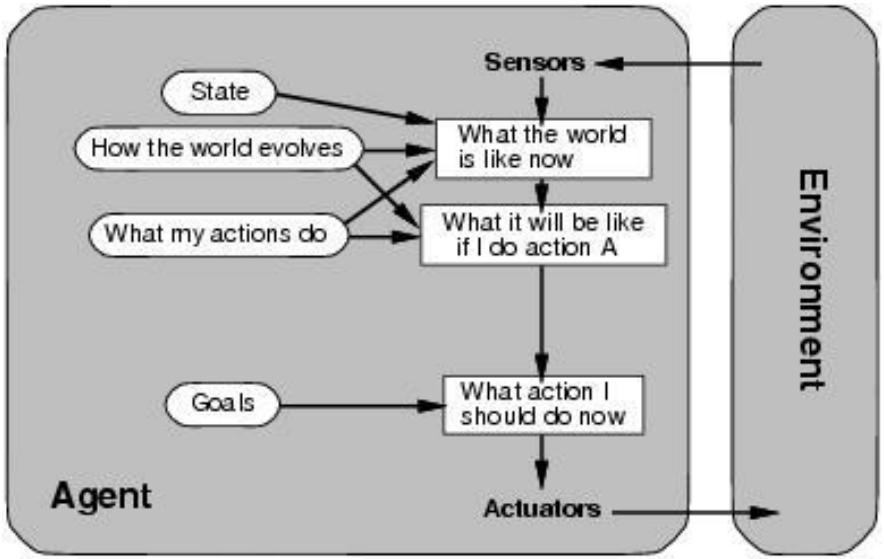
state ← UPDATE-STATE(*state*, *action*, *percept*)

rule ← RULE-MATCH(*state*, *rule*)

action ← RULE-ACTION[*rule*]

return *action*

Agent types; goal-based



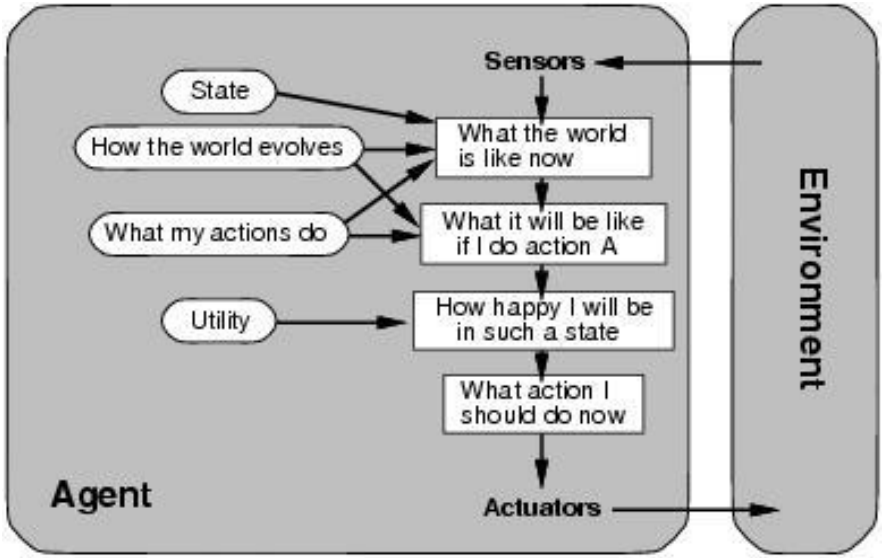
The agent needs a goal to know which situations are *desirable*.

- 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

Agent types; utility-based



Certain goals can be reached in different ways.

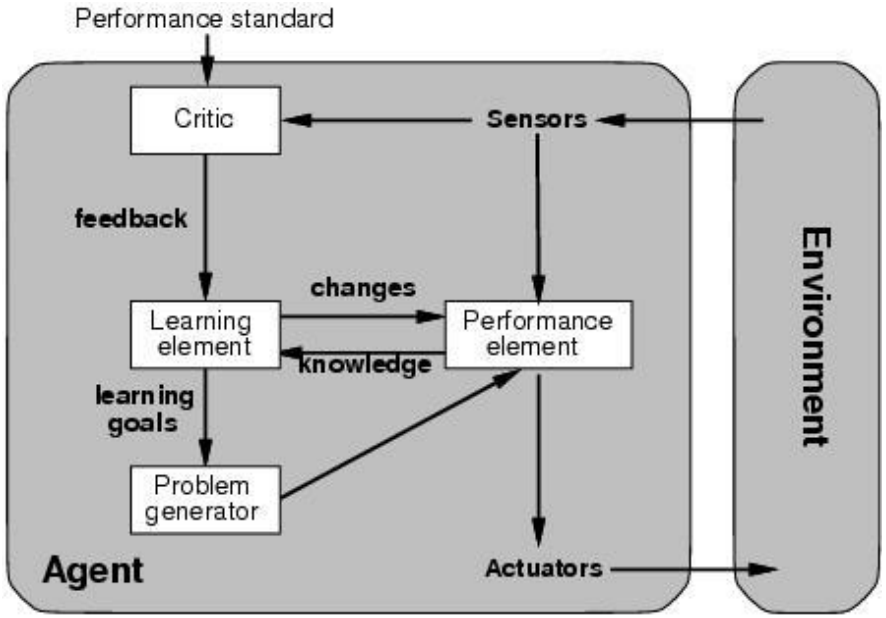
- Some are better, have a higher utility.

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.

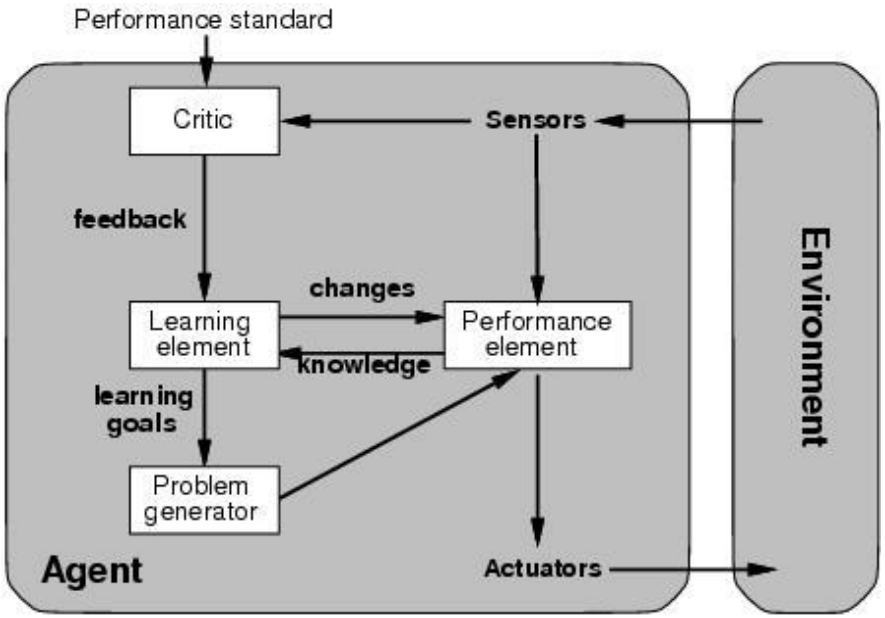
Agent types; learning



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.

Agent types; learning



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