Intelligent Agents

Chapter 2

Agents

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

- percept: perceptual input (eg. text, image, sound, ...)

- rational agent: for each possible percept sequence, a rational agent selects an action that maximizes performance measure, given evidence provided by the sequence, and built-in knowledge in the agent
  - does the right thing
- performance measure: criterion for success
  - good vs bad
  - better vs worse
  - clear criterion vs less well defined
- Rationality: reasonably correct
  - not perfection!

Environment types

- <u>Fully observable</u> (vs. partially observable)
  - Fully - everything seen shows all relevant information
  - partially - noise, inaccurate sensors, hidden info,...
- <u>Deterministic</u> (vs. stochastic)
  - next state depends on current state and next action
  - stochastic - probabilistic; other factors involved (complex?)
- <u>Episodic</u> (vs. sequential):
  - episodic one self-contained, independent situation
  - sequential - current decision affects future ones

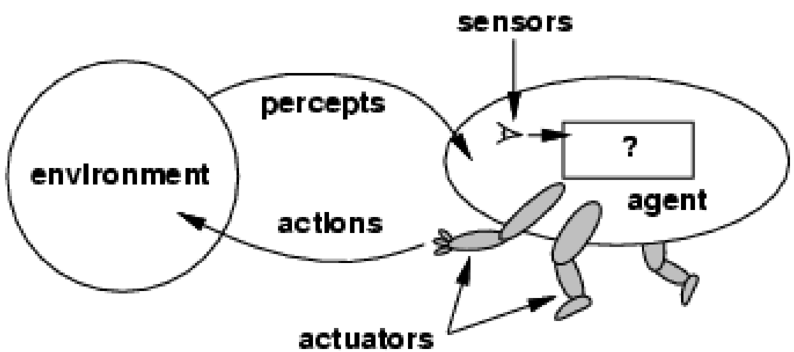
Environment types

- Static (vs. dynamic)
  - static - environment is fixed during decision making
  - dynamic - environment changes
- Discrete (vs. continuous)
  - discrete - finite # states (measurements, values,...)
  - Continuous - smooth, infinite scale
- Single agent (vs. multi-agent)
  - single - one agent involved
  - multi - more than one (adversary or cooperative)

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.

Agents and environments

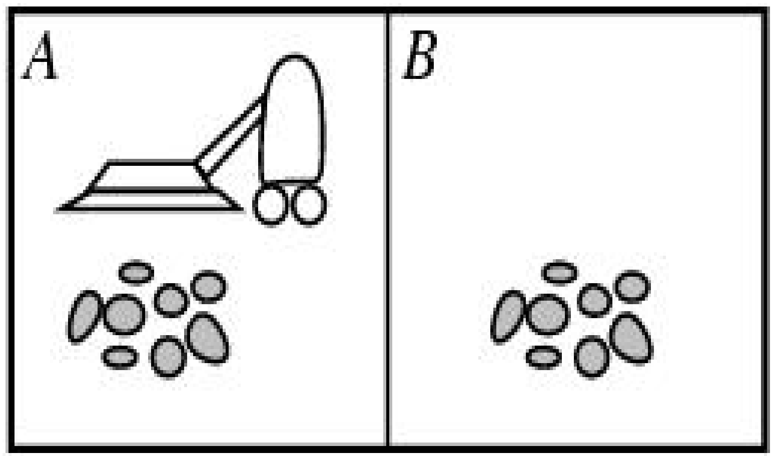


- The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^\star \rightarrow \mathcal{A}]$$

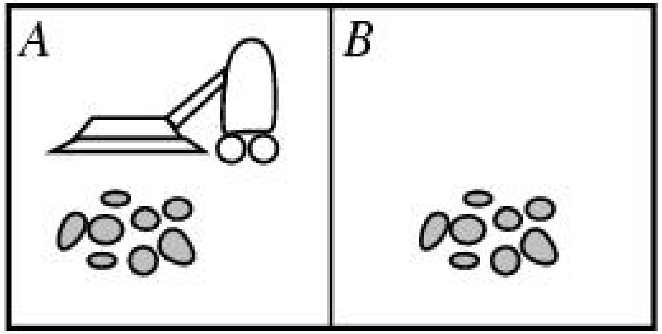
- The agent program runs on the physical architecture to produce *f*
- agent = architecture + program

The vacuum-cleaner world



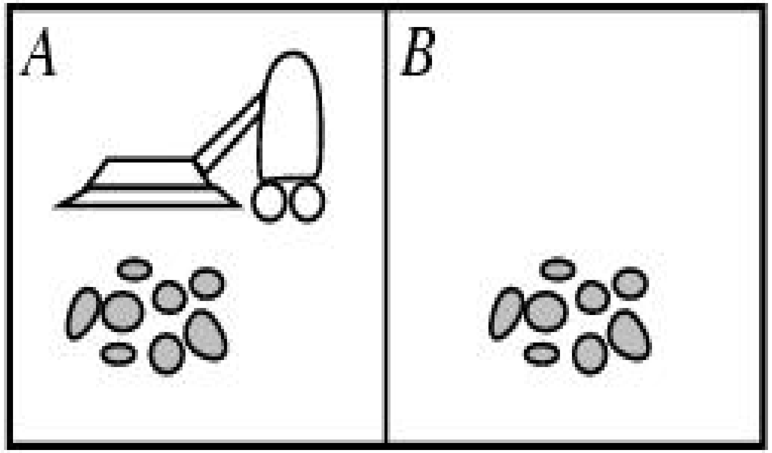
- Environment: square A and B
- Percepts: [location and content] e.g. *[A, Dirty]*
- Actions: left, right, suck, and no-op

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*

What is the right function? Can it be implemented in a small agent program?

The concept of rationality

- A **rational agent** is one that does the right thing.
  - Every entry in the table is filled out correctly.
- What is the right thing?
  - Approximation: the most *succesfull* agent.
  - *Measure of success?*
- Performance measure should be objective
  - E.g. the amount of dirt cleaned within a certain time.
  - E.g. how clean the floor is.
  - …
- *Performance measure according to what is wanted in the environment instead of how the agents should behave.*

Rationality

- What is rational at a given time depends on four things:
  - Performance measure,
  - Prior environment knowledge,
  - Actions,
  - Percept sequence to date (sensors).

Rational agents

- Rational Agent: 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 to date, and whatever built-in knowledge the agent has.

Rationality

- **Rationality $\neq$ omniscience**
  - An omniscient agent knows the actual outcome of its actions.
- **Rationality $\neq$ perfection**
  - Rationality maximizes *expected* performance, while perfection maximizes *actual* performance.

Rationality

- The proposed definition requires:
  - Information gathering/exploration
    - To maximize future rewards
  - Learn from percepts
    - Extending prior knowledge
  - Agent autonomy
    - Compensate for incorrect prior knowledge

PEAS

- To design a rational agent we must specify its task environment.
- PEAS description of the environment:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

Example

- **Agent**: Internet shopping agent
- Performance measure: price,quality,appropriateness,efficiency
- Environment: current and future WWW sites,vendors,shippers
- Actuators: display to user,follow URL,fill in form
- Sensors: HTML pages (text, graphics, scripts)

Example

- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure: ?
  - Environment: ?
  - Actuators: ?
  - Sensors:?

Example

- E.g. Fully automated taxi:
  - PEAS description of the environment:
    - – Performance?
      - » Safety, destination, profits, legality, comfort
    - – Environment?
      - » Streets/freeways, other traffic, pedestrians, weather,, …
    - – Actuators?
      - » Steering, accelerating, brake, horn, speaker/display,…
    - – Sensors?
      - » Video, sonar, speedometer, engine sensors, keyboard, GPS, …

Example

- **Agent**: Medical diagnosis system
- Performance measure:
- Environment:
- Actuators:
- Sensors:

Example

- **Agent**: Medical diagnosis system
- Performance measure: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

Example

- Agent: Interactive English tutor
- Performance measure: ?
- Environment:?
- Actuators:?
- Sensors:?

Example

- **Agent**: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students, testing agency
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

Example

- Agent: Part-picking robot
- Performance measure:
- Environment:
- Actuators:
- Sensors:

Example

- Agent: Part-picking robot
- Performance measure: Percentage of parts in correct bins
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

Agents: examples

- <u>Human agent</u>: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for <u>actuators</u>
- <u>Robotic agent</u>: cameras and infrared range finders for sensors; various motors for actuators
- <u>Software agent</u>: sensors: keystrokes, file contents, network packets; actuators: display on screen, write files, send network packets etc

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	Backgammom	Intenet shopping	Taxi
Observable??	FULL	FULL	PARTIAL	PARTIAL
Deterministic ??	YES	NO	YES	NO
Episodic??	NO	NO	NO	NO
Static??	YES	YES	SEMI	NO
Discrete??	YES	YES	YES	NO
Single-agent??	YES	NO	NO	NO

Environment types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- The environment type largely determines the agent design
- The simplest environment is
  - Fully observable, deterministic, episodic, static, discrete and single-agent.
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- **Aim:** find a way to implement the rational agent function concisely

Agent Structure

- How does the inside of the agent work?
  – Agent = architecture + program
- All agents have the same skeleton:
  – Input = current percepts
  – Output = action
  – Program= manipulates input to produce output
- Note difference with agent function.

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

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

static: *percepts*, a sequence initially empty
table, a table of actions, indexed by percept sequence

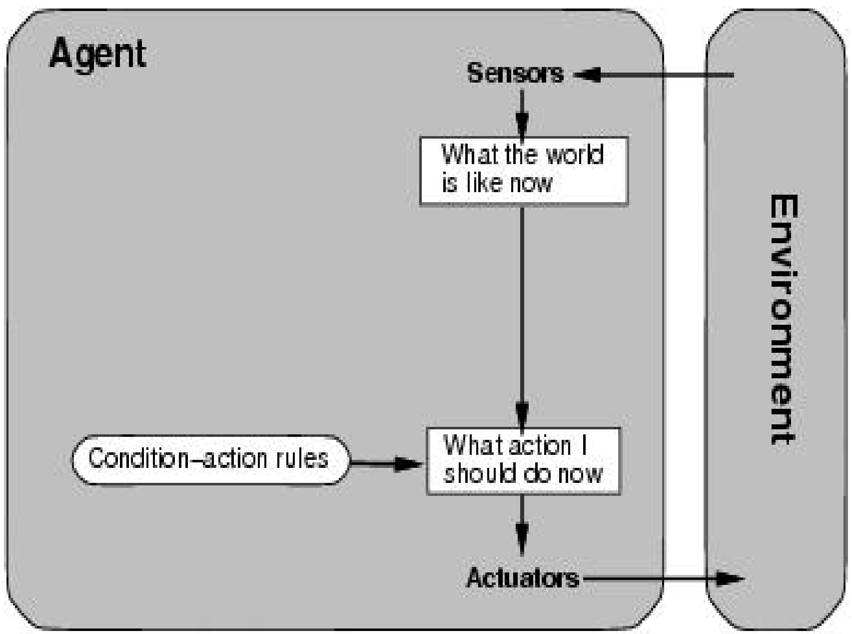
append *percept* to the end of *percepts*
action $\leftarrow$ LOOKUP(*percepts*, *table*)
return *action*

This approach is doomed to failure

Table-lookup agent

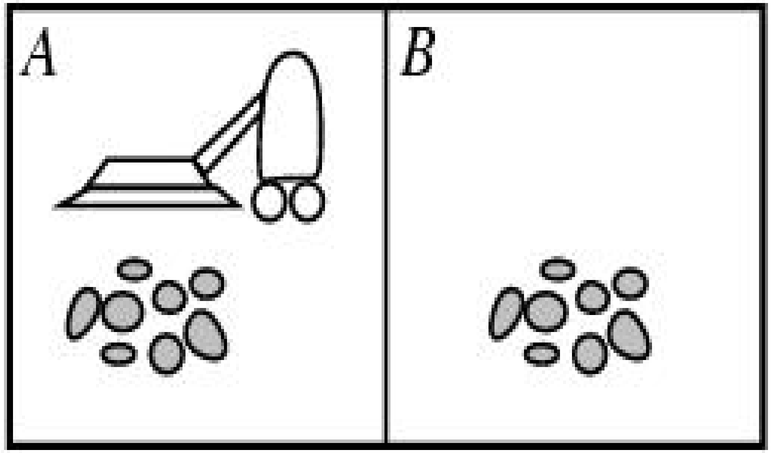
- \input{algorithms/table-agent-algorithm}
- Drawbacks:
  - Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries

Agent types; simple reflex



- Select action on the basis of *only the current* percept.
  - E.g. the vacuum-agent
- Large reduction in possible percept/action situations.
- 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*

Agent types; simple reflex

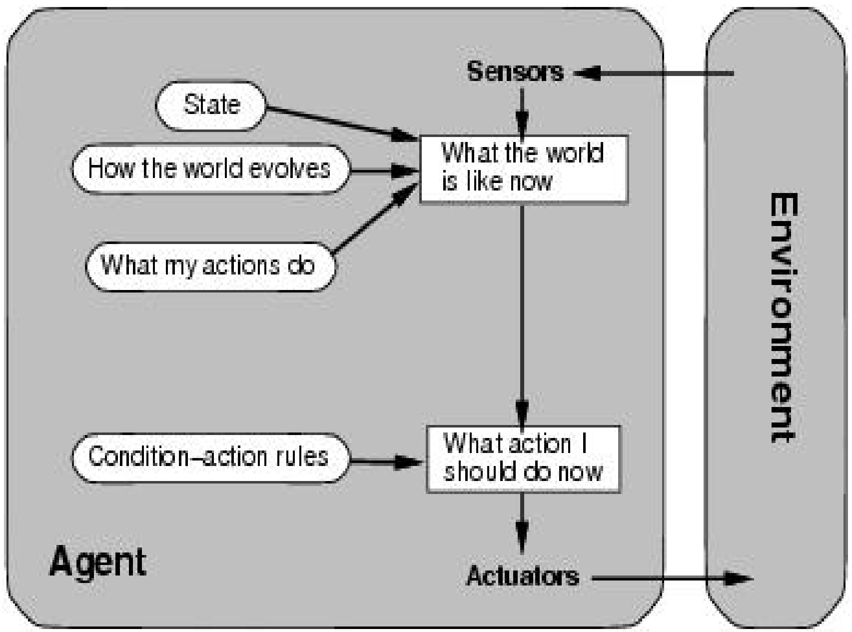
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

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.

Model-based reflex agent



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

  $\Rightarrow$ *Model of World*

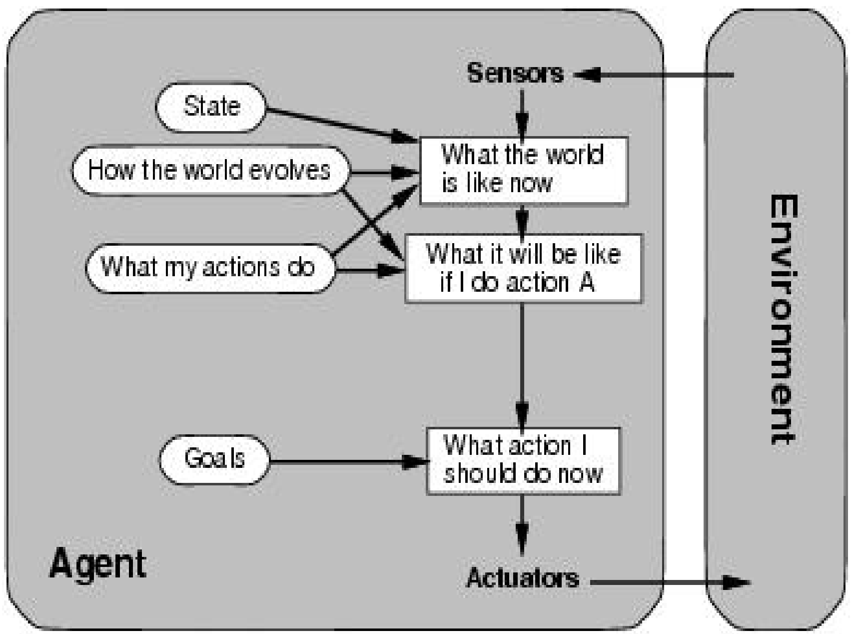
Model-based reflex agent

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.

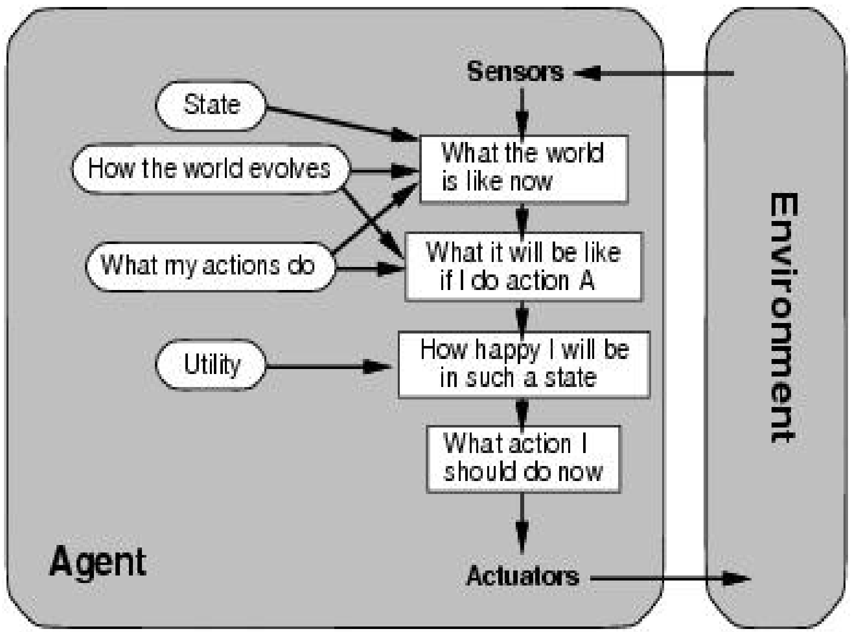
state ← UPDATE-STATE(*state*, *action*, *percept*)
rule ← RULE-MATCH(*state*, *rule*)
action ← RULE-ACTION[*rule*]
return *action*

Agent types; goal-based



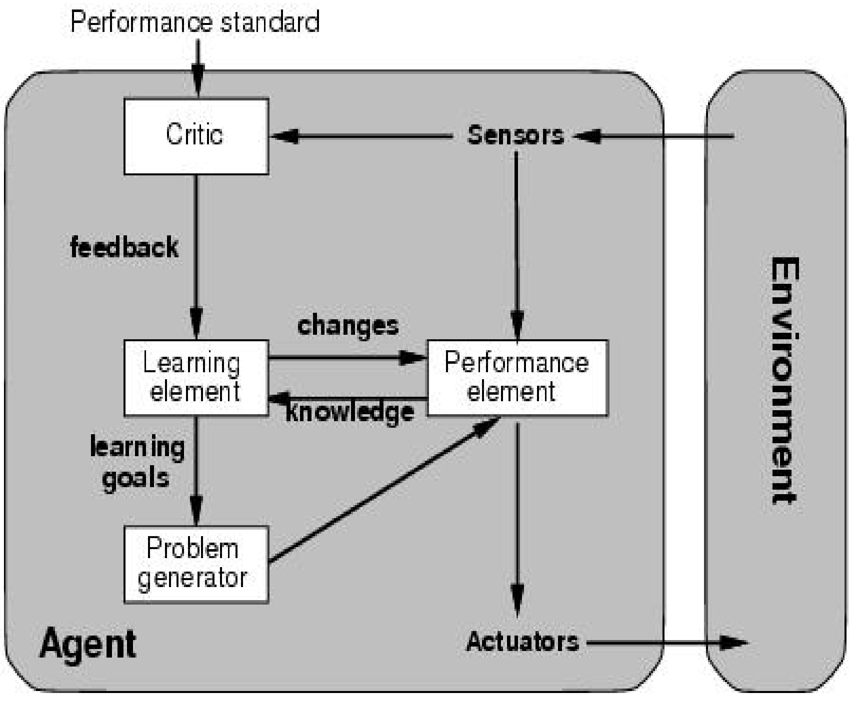
- The agent needs a goal to know which situations are *desirable*.
  - Things become difficult when long sequences of actions are required to find the goal.
- Typically investigated in **search** and **planning** research.
- Major difference: future is taken into account
- Is more flexible since knowledge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

Agent types; utility-based



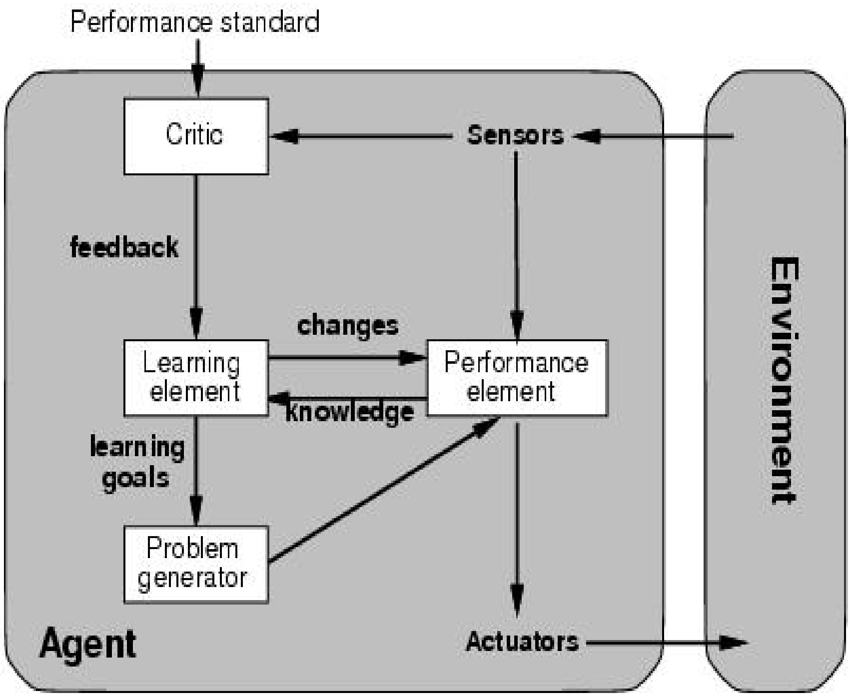
- Certain goals can be reached in different ways.
  - Some are better, have a higher utility.
- Utility function maps a (sequence of) state(s) onto a real number.
- Improves on goals:
  - Selecting between conflicting goals
  - Select appropriately between several goals based on likelihood of success.

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