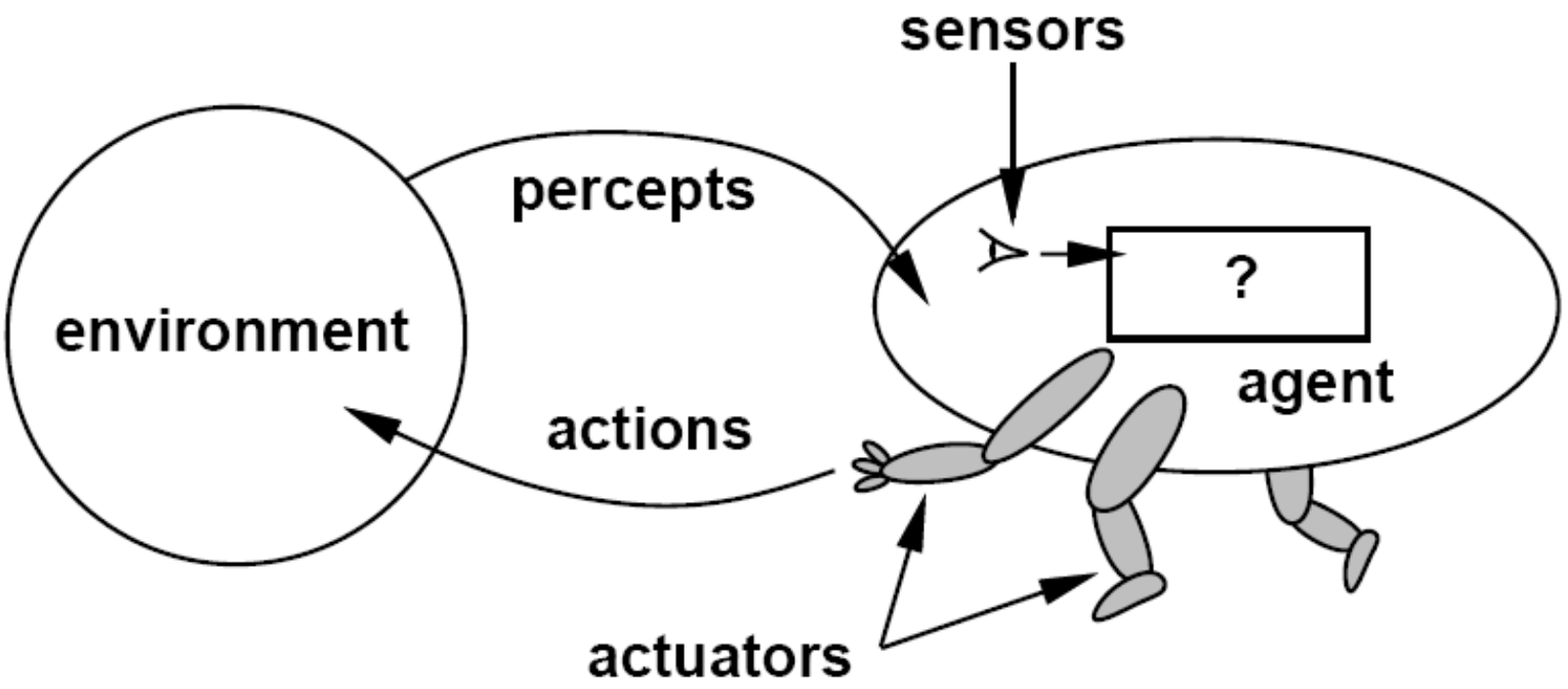
Intelligent Agents

Outline

- Agent function and agent program
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

Agents

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

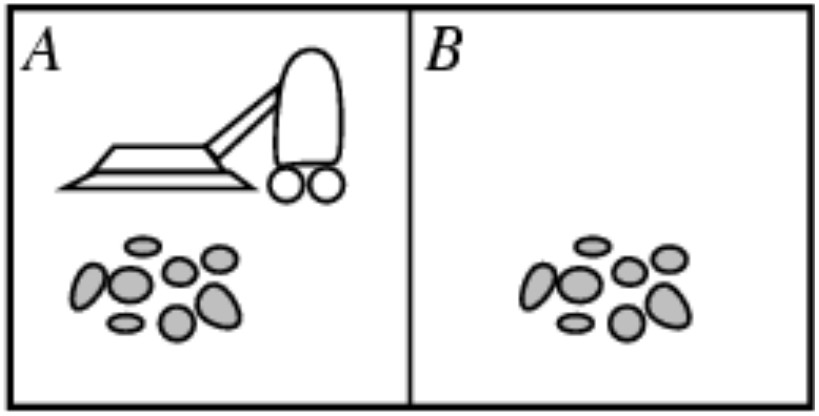


sensors/percepts and actuators/actions?

- Human agent:
  - eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- Robotic agent:
  - cameras and infrared range finders for sensors; various motors for actuators
- Software Agent:
  - keystrokes, file contents & network packets as sensory inputs; acts by displaying on screen, writing to files & sending network packets

Example: Vacuum-cleaner world

- **Percepts:**
  Location and status,
  e.g., [A, Dirty]
- **Actions:**
  Left, Right, Suck, NoOp



Example vacuum agent program:

function Vacuum-Agent([location,status]) returns an action

if status = Dirty *then* return Suck

else if location = A *then* return Right

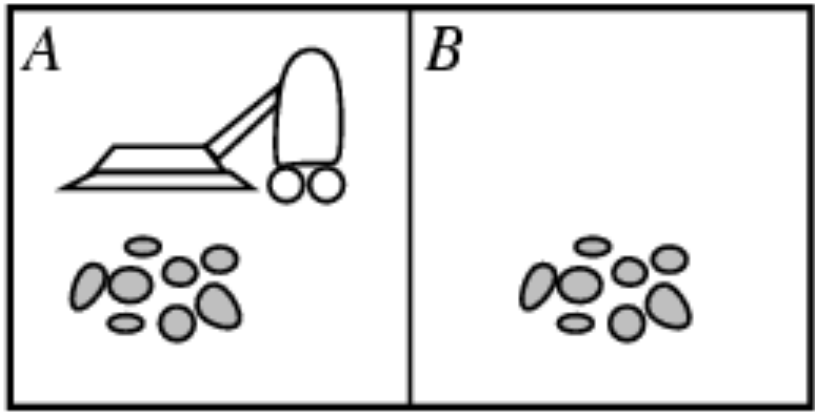
else if location = B *then* return Left

Rational agents

- Select actions expected to maximize its **performance measure**
- Based on evidence provided by the percept sequence and the agent's built-in knowledge
- "do the right thing"
- **Performance measure (utility function):** An *objective* criterion for success of an agent's behavior

Example: Vacuum-Agent

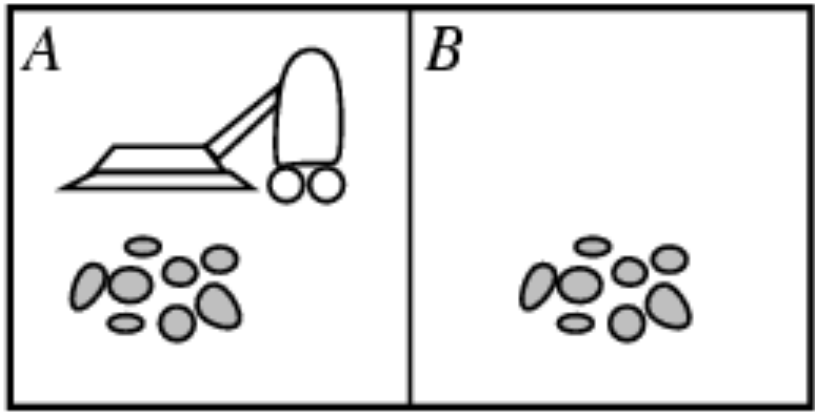
- **Percepts:**
  Location and status,
  e.g., [A,Dirty]
- **Actions:**
  Left, Right, Suck, Dump, NoOp



- Potential performance measures for our vacuum agent?
  - amount of dirt cleaned up,
  - amount of time taken,
  - amount of electricity consumed,
  - amount of noise generated, etc.

Example: Vacuum-Agent

- **Percepts:**
  Location and status,
  e.g., [A,Dirty]
- **Actions:**
  Left, Right, Suck, Dump, NoOp



function 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

Is this agent rational?

Autonomy

- Autonomous agent – behavior determined by its own experience
  - ability to learn and adapt
  - can always say “no”
  - needs enough built-in knowledge to survive
- Not autonomous agent
  - guided by its designer

Task Environment Specification

- Problem specification: **Performance measure, Environment, Actuators, Sensors (PEAS)**

- **Example: autonomous taxi**
  - **Performance measure**
    - Safe, fast, legal, comfortable trip, maximize profits
  - **Environment**
    - Roads, other traffic, pedestrians, customers
  - **Actuators**
    - Steering wheel, accelerator, brake, signal, horn
  - **Sensors**
    - Cameras, speedometer, GPS, odometer, engine sensors, keyboard

Agent: Spam filter

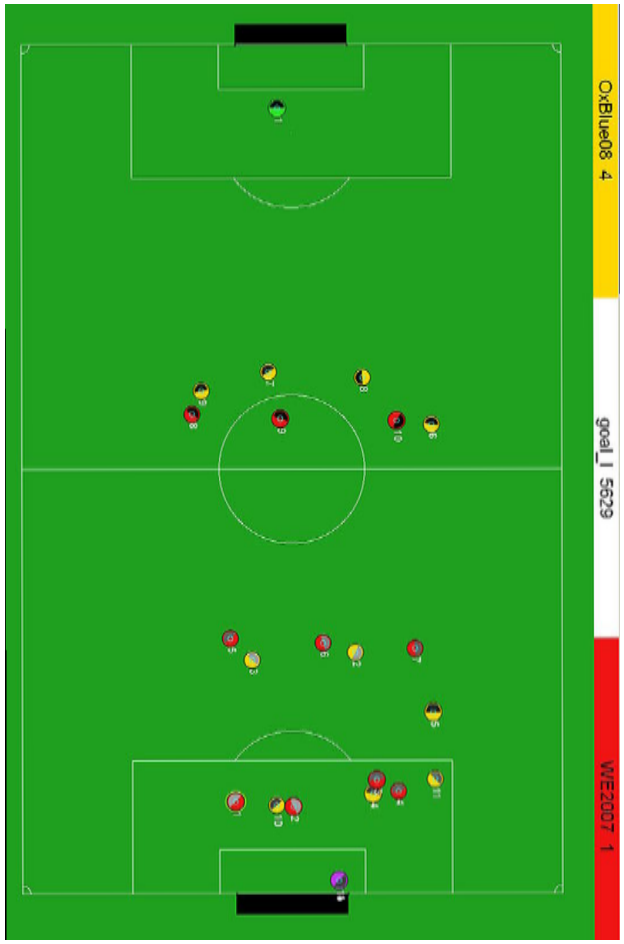
- Performance measure
  - Minimizing false positives, false negatives
- Environment
  - A user's email account, email server
- Actuators
  - Mark as spam, delete, etc.
- Sensors
  - Incoming messages, other information about user's account

Environment types

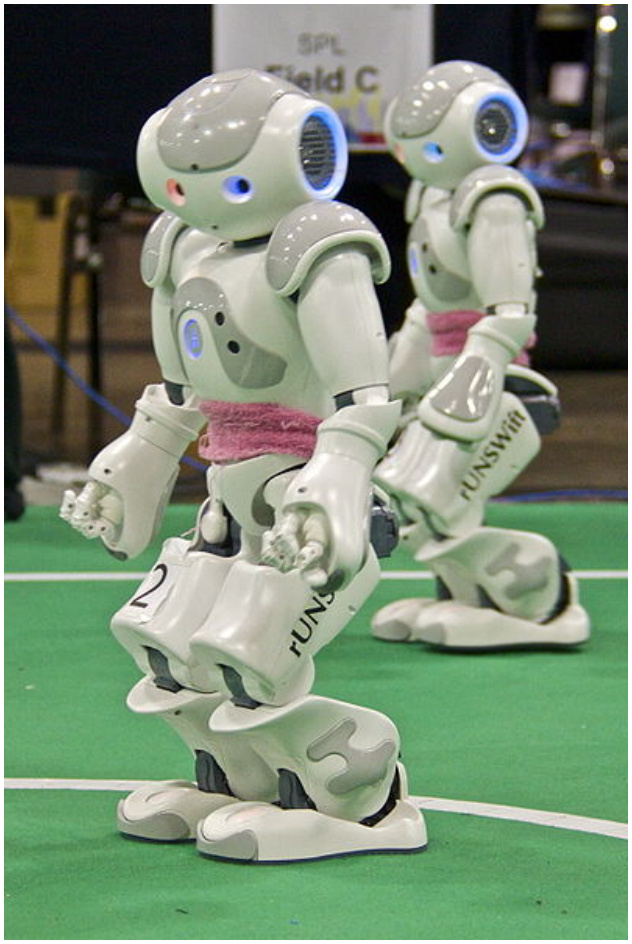
- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multi-agent

Fully observable vs. partially observable

- Do the agent's sensors give it access to the complete state of the environment?



vs.



Deterministic vs. stochastic

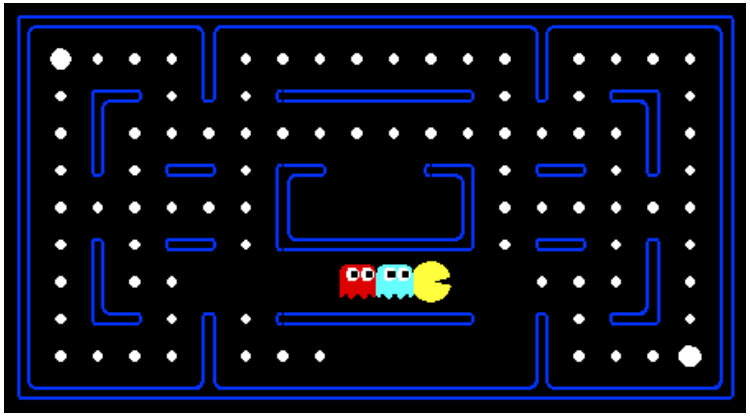
- Is the next state of the environment completely determined by the current state and the agent's action?

vs.

Episodic vs. sequential

- Is the agent's experience divided into unconnected episodes, or is it a coherent sequence of observations and actions?

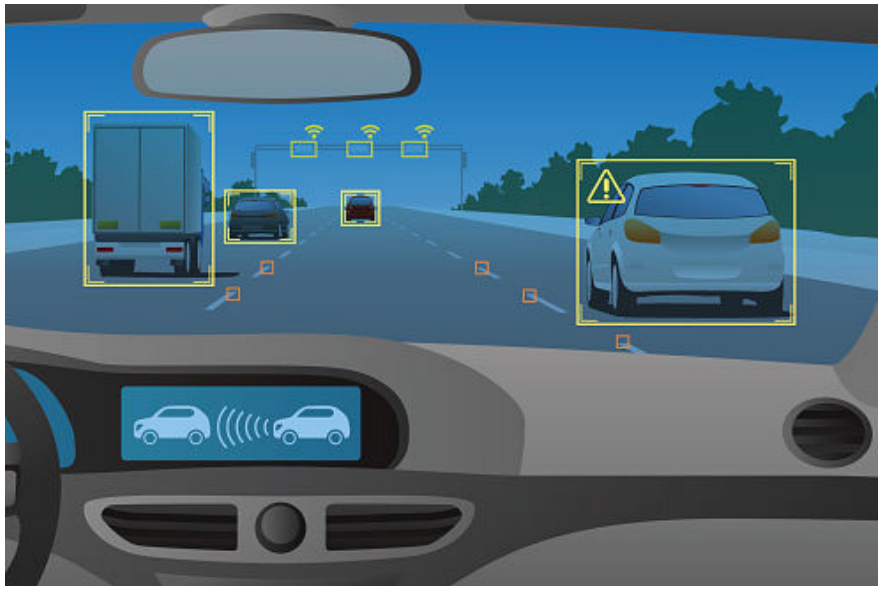
vs.



Static vs. dynamic

- Is the world changing while the agent is thinking?

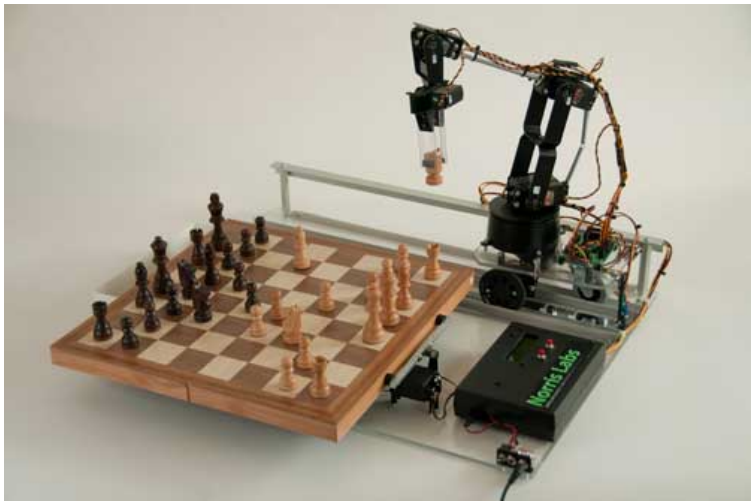
vs.



Discrete vs. continuous

- Does the environment provide a fixed number of distinct percepts, actions, and environment states?
  - Time can also evolve in a discrete or continuous fashion

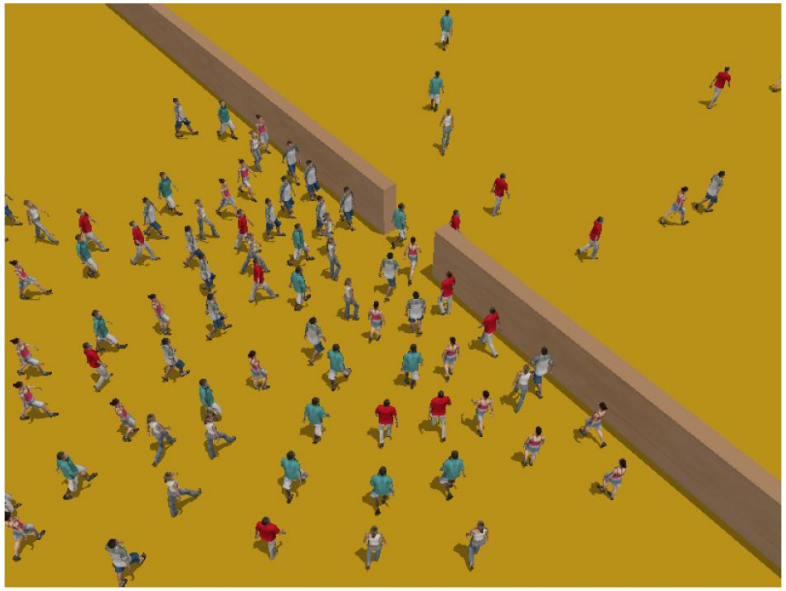
vs.



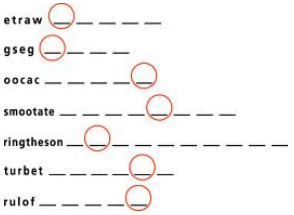
Single-agent vs. multiagent

- Is an agent operating by itself in the environment?

vs.



Examples of different environments



Word jumble solver

Chess with a clock

Scrabble

Autonomous driving

	Word jumble solver	Chess with a clock	Scrabble	Autonomous driving
Observable	Fully	Fully	Partially	Partially
Deterministic	Deterministic	Deterministic	Stochastic	Stochastic
Episodic	Episodic	Sequential	Sequential	Sequential
Static	Static	Dynamic	Static	Dynamic
Discrete	Discrete	Continuous	Discrete	Continuous
Single agent	Single	Multi	Multi	Multi

Hierarchy of agent types

(0) Table-driven agents

- percept sequence/action table to find the next action
- (large) **lookup table**

(1) Simple reflex agents

- **condition-action rules**,
- production system; no memory of past world states

(2) Agents with memory

- **internal state** to keep track of past states of the world

(3) Agents with goals

- state and **goal information** about desirable situations
- take future events into consideration

(4) Utility-based agents

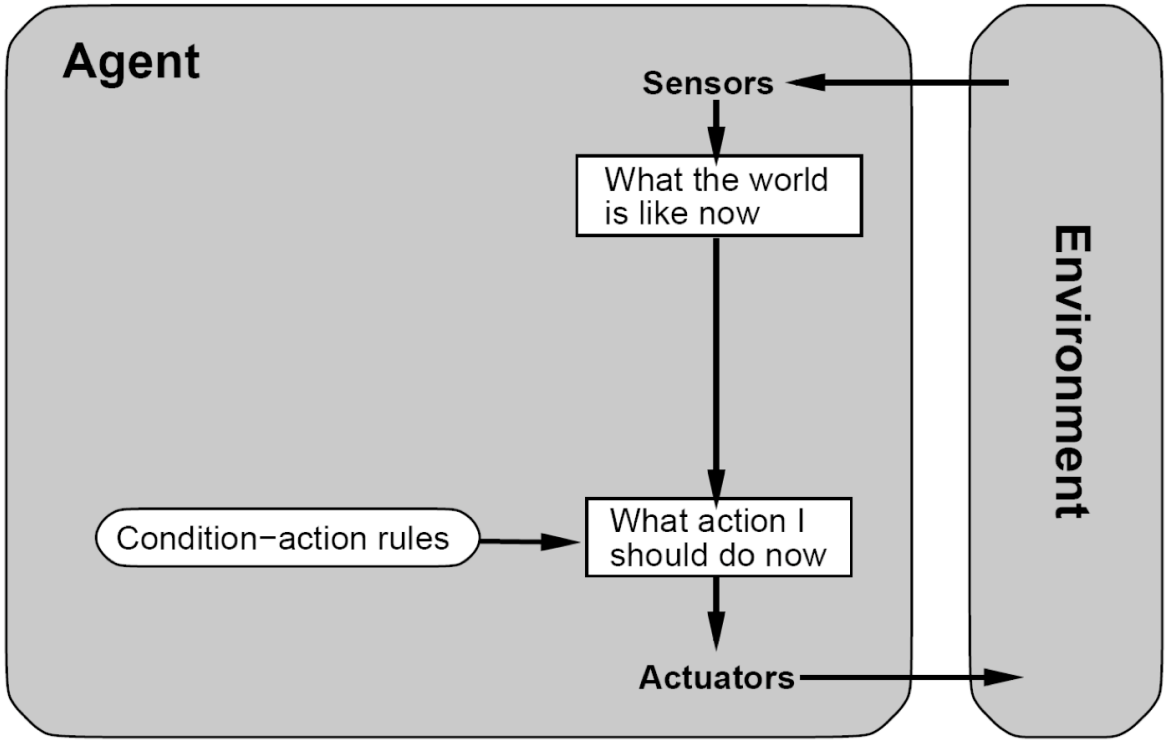
- base decisions on **utility theory** in order to act rationally

Table-Driven Agents - Problems

- Daunting sizes of tables
  - physical agents cannot store
  - designer – no time and knowledge to create
  - Not very practical!

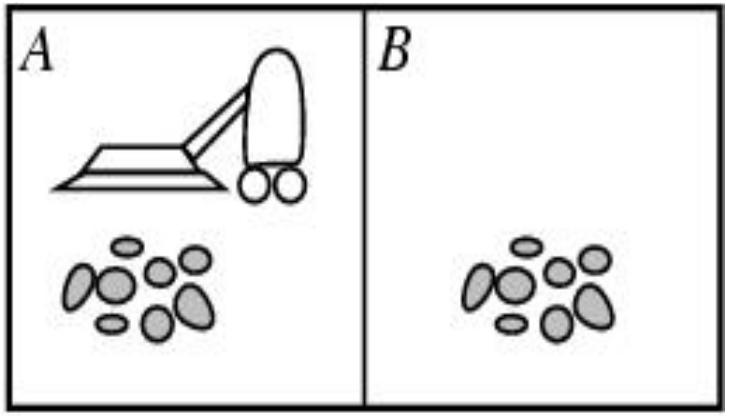
Simple reflex agent

- Select action based on current percept only
- Implemented through condition-action rules: if dirty then suck



Examples of classic **if-then** algorithms:
if (I sense a certain input**) then (**I apply a specific rule**)**
Issue: environment needs to be fully-observable. (Consider a self-driving car?)

The vacuum-cleaner world



function REFLEX-VACUUM-AGENT (percept) returns an action

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

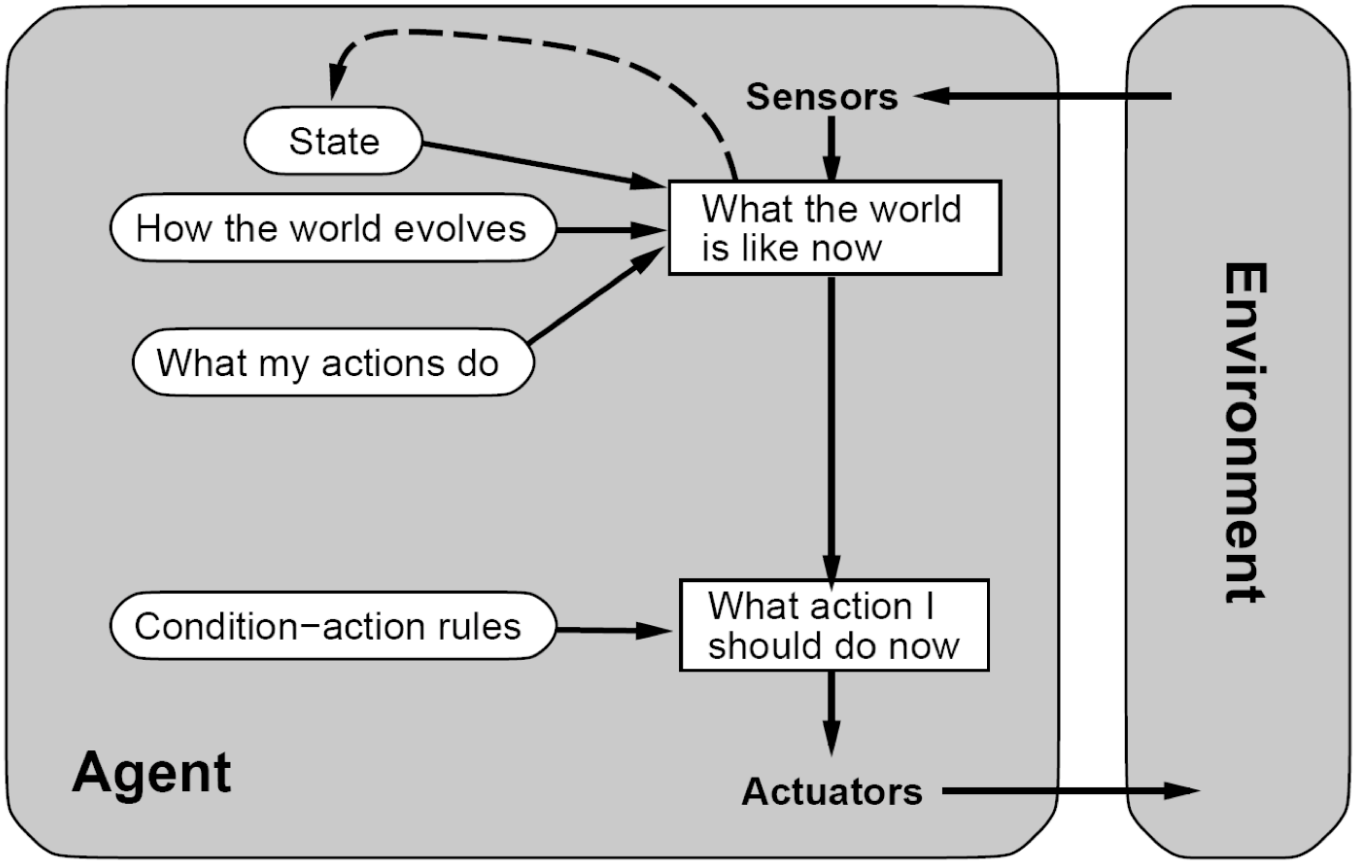
rule ← RULE-MATCH(*state*, *rules*)

action ← RULE-ACTION[*rule*]

return *action*

Model-based reflex agent

- Internal state: aspects of the environment that cannot be currently observed
- For partially observable environments



Model-based reflex agents

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

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

model, a description of how the next state depends on current state and action

state, a description of the current world state

action, the most recent action

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

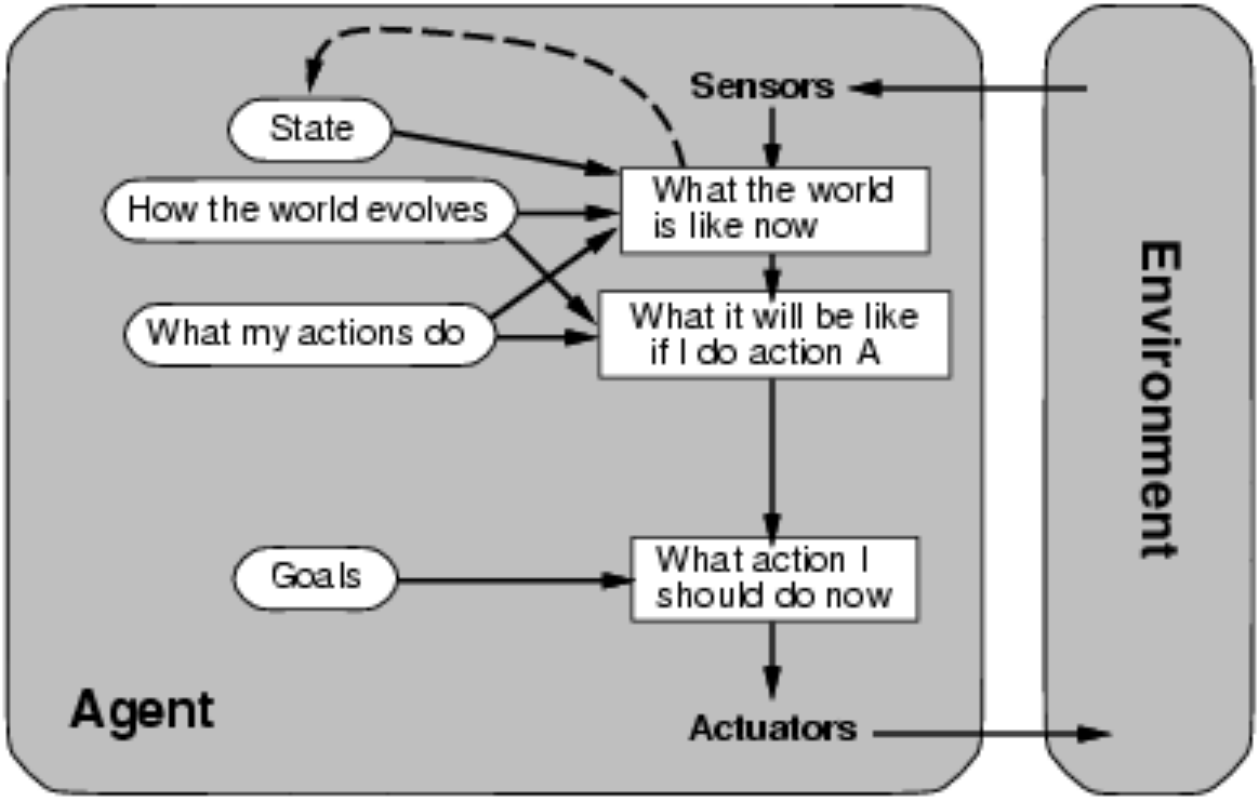
rule ← RULE-MATCH(*state*, *rules*)

action ← RULE-ACTION[*rule*]

return *action*

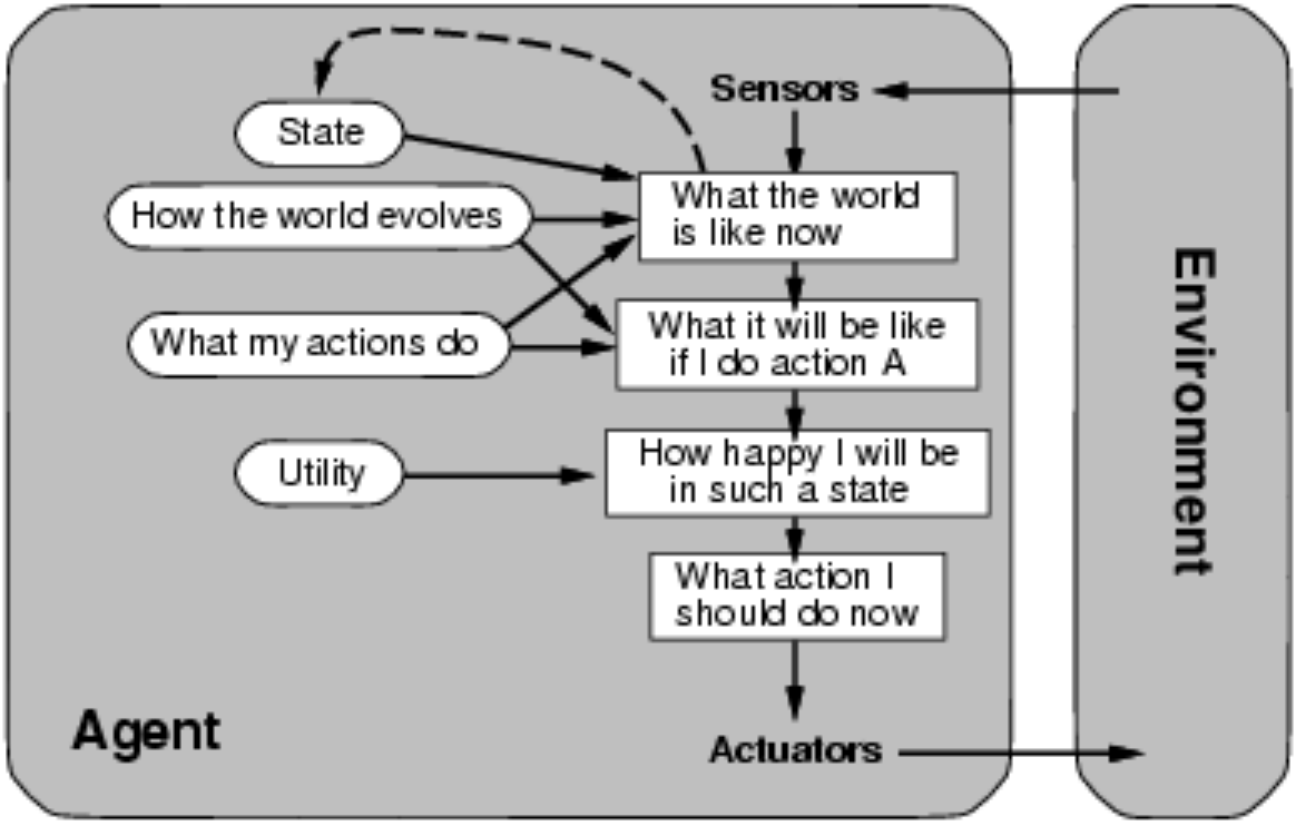
Goal-based agent

- Goal information to select between possible actions in the current state
- Goal: desirable situations



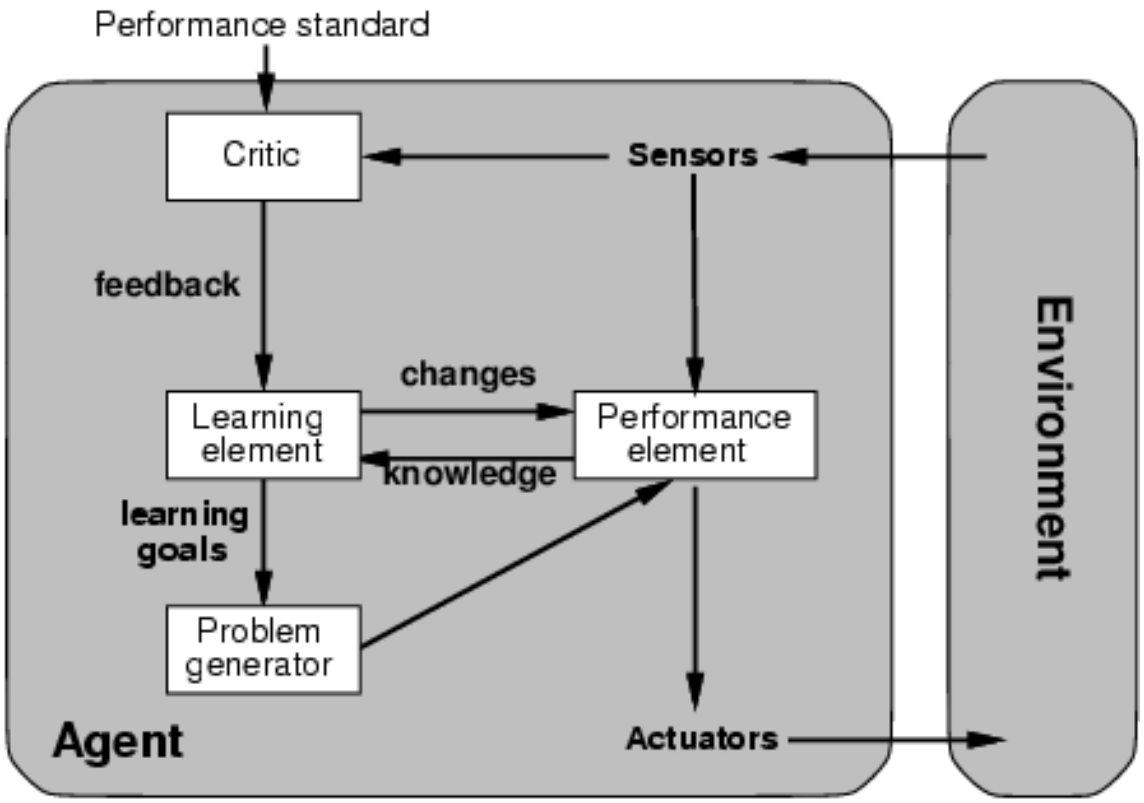
Utility-based agent

- Utility function: evaluate desirability of states resulting from each possible action



- Certain goals can be reached in different ways:
  - "better" ways have a higher utilities
- Utility function: maps (sequence of) state(s) onto a real number.

Learning/Autonomous agent



- All previous agent-programs: methods for selecting *actions*
  - Learning mechanisms can perform this task
  - Teach instead of instructing
  - Advantage: robustness in initially unknown environments.

Summary

- An **agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program
- A **rational agent** always chooses the action that maximizes its expected performance, given its percept sequence so far
- An **autonomous agent** uses its own experience rather than built-in knowledge of the environment by the designer
- An **agent program** maps percepts to actions and updates its internal state
  - **Reflex agents** respond immediately to percepts
  - **Goal-based agents** act in order to achieve their goal(s)
  - **Utility-based agents** maximize their own utility function
- The most challenging environments are partially observable, stochastic, sequential, dynamic, and continuous, and contain multiple intelligent agents.