# Intelligent Agents

# Outline

- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

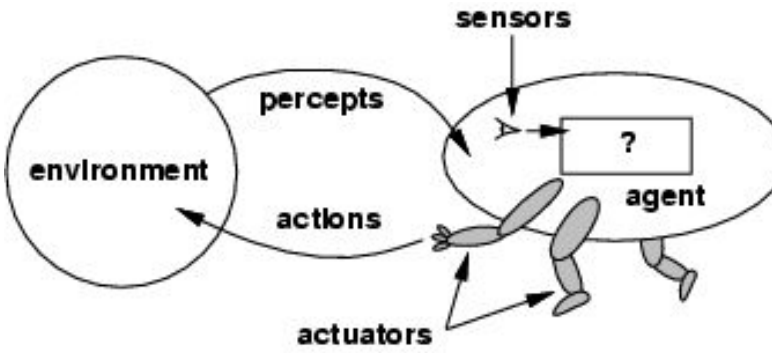
# Agents (rational agents)

- An agent is anything that can be viewed as perceiving its environment through sensors and acting 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: cameras and infrared range finders for sensors;various motors for actuators

# Agents and environments




- Percept: refer to the agents perceptual input at any given instant.
- The agent function maps any given percept sequence to an action [$f$: $\mathcal{P}^{\star}$ □ $\mathcal{A}$]
  Agent function will be implemented by agent program
- The agent function is an abstract mathematical description ; agent program is a concrete implementation, running on agent architecture

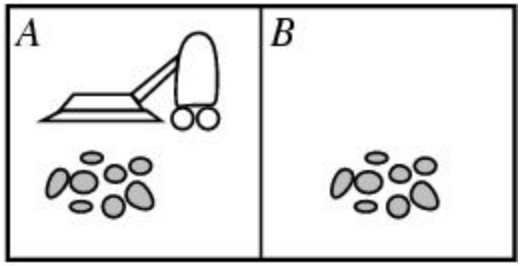
Percept : input at  given instance

Percept sequence : complete history

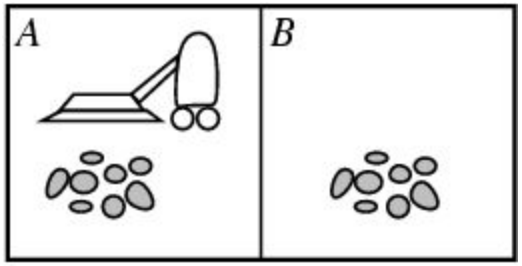
Tabulating the agent function that describes any given agent (for most agent it will be a long table)

Given an agent to experiment with, we can in principle construct this table by trying out all possible percept sequence and recording which actions the agent does in response

# Vacuum-cleaner world



# Vacuum-cleaner world

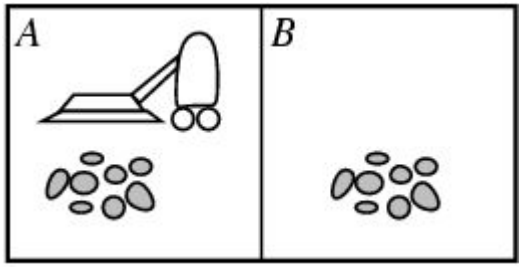



Input

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

Output

# Example: Vacuum Cleaner Agent




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

**history : complete sequence agent has perceived**

| Percept sequence | Action |
|---|---|
| $[A, Clean]$ | $Right$ |
| $[A, Dirty]$ | $Suck$ |
| $[B, Clean]$ | $Left$ |
| $[B, Dirty]$ | $Suck$ |
| $[A, Clean], [A, Clean]$ | $Right$ |
| $[A, Clean], [A, Dirty]$ | $Suck$ |
| ⋮ | ⋮ |

# Rational agents

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful

- Performance measure: An objective criterion for success of an agent's behavior

- 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.

# Rational agents

- Rational depend on :
  - ✔ The performance measure that defines degree of success.
  - ✔ What the agent knows about the environment.
  - ✔ The actions that the agent can perform.(Actuator)
  - ✔ The agent's percept sequence to date (Sensors)

- Def : 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 and whatever built-in knowledge the agent has.

# Rational agents

- Rationality is distinct from omniscience (all-knowing with infinite knowledge)
- Omniscience agent knows the actual outcome of its action and act accordingly
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration) learning
- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

# Specifying the task environment

- Task Environment : essentially the “problems” to which rational agent are the “solution”
- We have to specify the : Performance measure, Environment, Actuators, Sensors(PEAS)

# PEAS

- Automated taxi driver
- Medical diagnosis system
- Part-picking robot
- Interactive English tutor

# PEAS

- Must first specify the setting for intelligent agent design

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

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

# PEAS

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)

# PEAS

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

# PEAS

Agent: Interactive English tutor

- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

# Environment types

- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.

- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)

- Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

# Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)

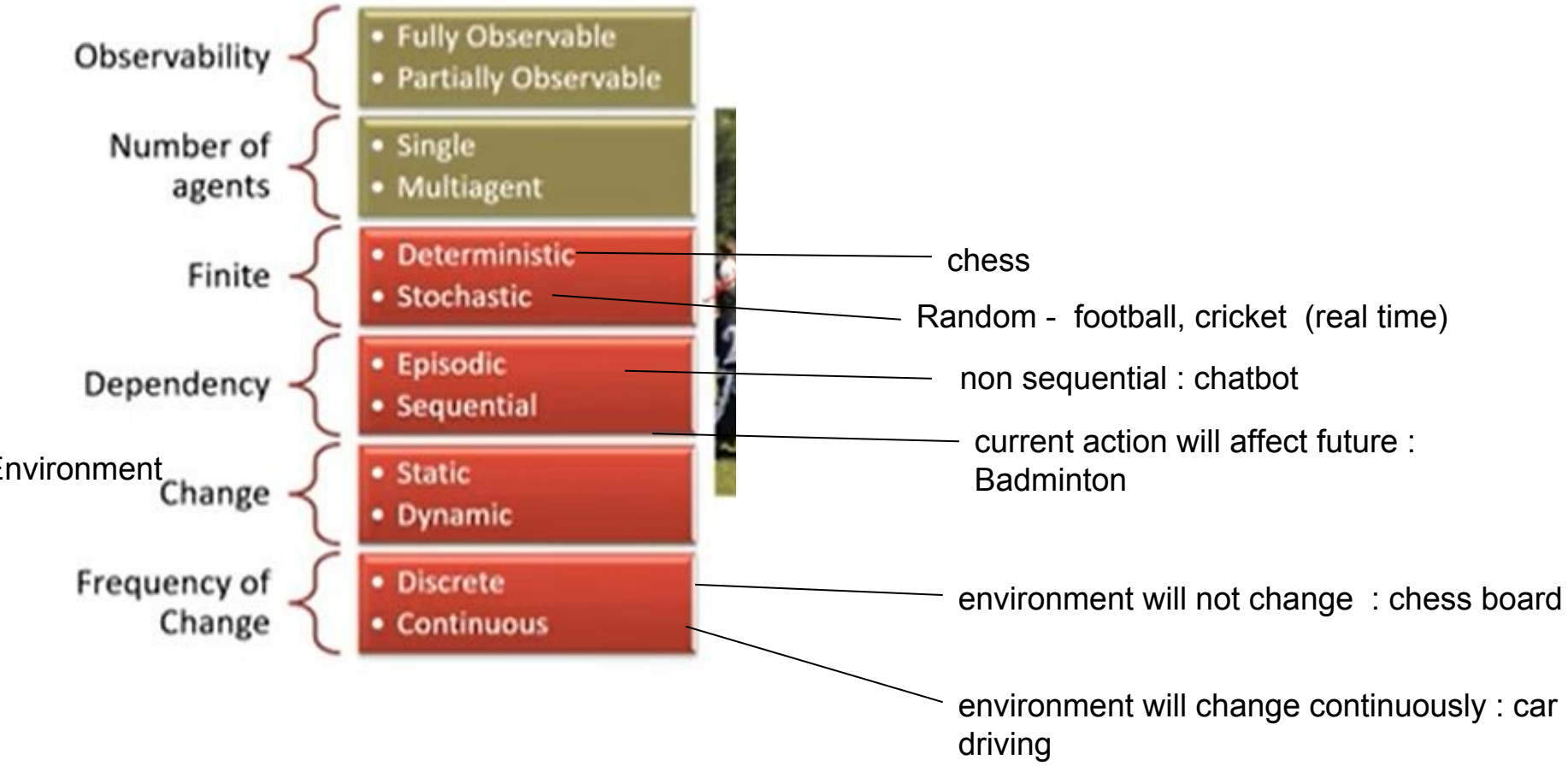
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.

- Single agent (vs. multiagent): An agent operating by itself in an environment.

# Environment types

| | Chess with a clock | Chess without a clock | Taxi driving |
|---|---|---|---|
| Fully observable | | | |
| Deterministic | | | |
| Episodic | | | |
| Static | | | |
| Discrete | | | |
| Single agent | | | |

# Types of environment



# 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 real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

# The structure of Agent

- The job of AI is to design the **agent program:** that implement the agent function mapping percepts to actions. We assume this program will run on some sort of computing device, which we will call the **architecture.**
- In general, the architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the effectors as they are generated.

***agent = architecture + program***

- **function** TABLE-DRIVEN-AGENT(percept) **returns** *action*

  **static:** *percepts,* a sequence, initially empty

  *table,* a table of action, indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*

  *action* <— *LOOKUP(percepts, table)* **return** *action*

***Agent Program :Keeps track of percept sequence and use it to index into a table of action.***

***The table represent explicitly the agent function that agent program embodies***

# 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.

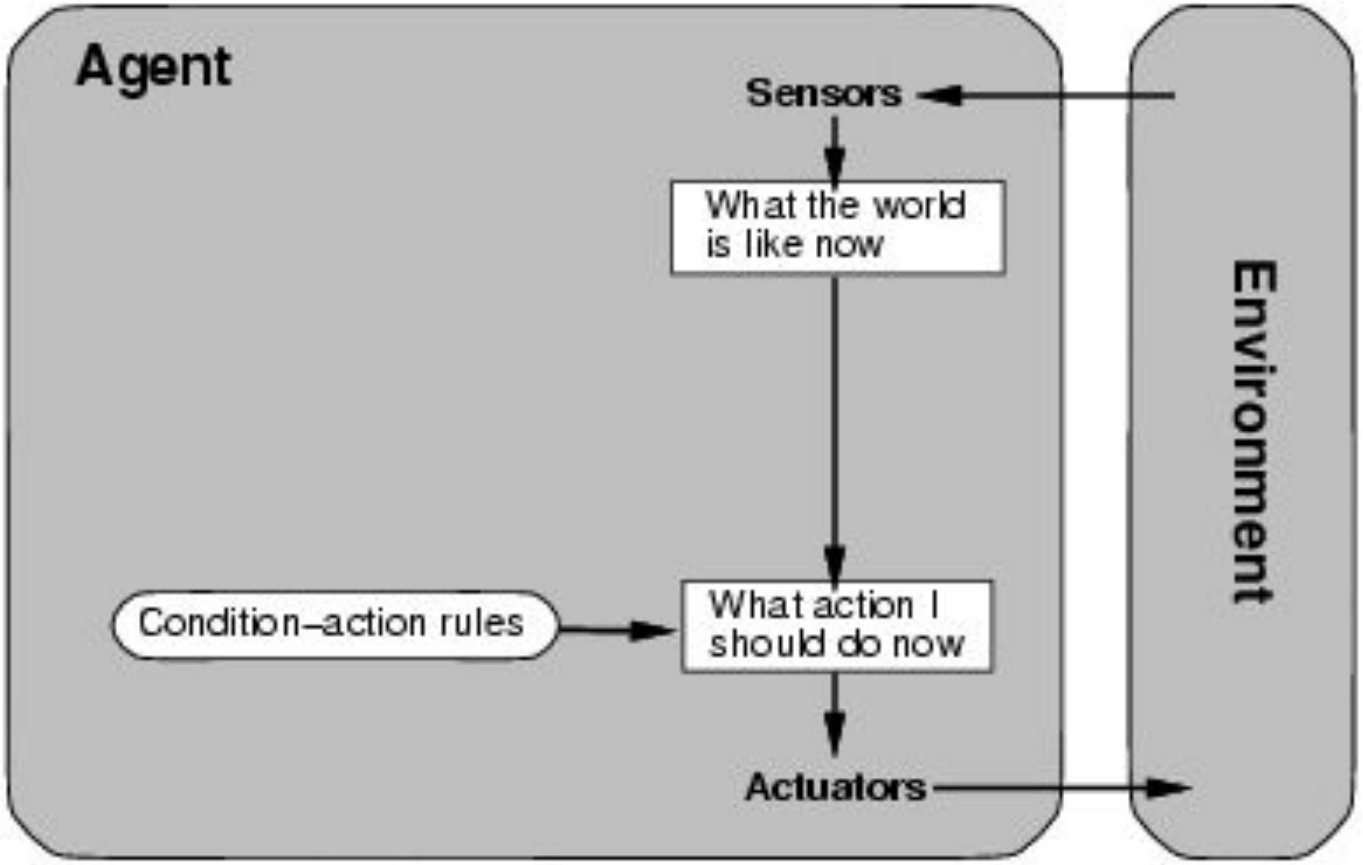
The key challenge for AI is to find out how to write programs that to the extent possible produce rational behavior from small amount of code rather than large number of table entries.

# Agent types/Program

Four basic types in order of increasing generality:

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

# Simple reflex agents




We use rectangle to denote the current internal state of the agents decision process and ovals to represent the background information used in the process

# Simple Reflex Agent

- **function** SiMPLE-REFLEX-AGENT(Percept)**returns** *action*
- **static:** *rules,* a set of condition-action rules
- *state* <— INTERPRET-INPUT(percept)
- rule <- RULE-MATCH(state, *rules)*
- *action* <- RULE-ACTiON[rule]
- **return** *action*

*It acts according to a rule whose condition matches the current state as defined by percept.*

# Simple Reflex Agent

The simplest Kind of agent, agent select actions on the basis of the current percept, ignoring the rest of the percept history.

- Vacuum Cleaner – based on current location and on whether that contains dirt
- Car driving: if the car in front brakes, and its brake lights come on, then the driver should notice this and initiate braking. **condition-action *rule*** ***if*** *car-in-front-is-braking* ***then*** *initiate-braking.*

## *Issues*

- *If decision can be made on the basis of current percept i.e only If the environment is fully observable.*
- sensors do not provide access to the complete state of the world. In such cases, the agent may need to maintain some internal state information
- *Infinite Loop: due to partially observable environment.*

function
REFLEX_VACUUM_AGENT([LOCATION,STSTUS])
returns an action

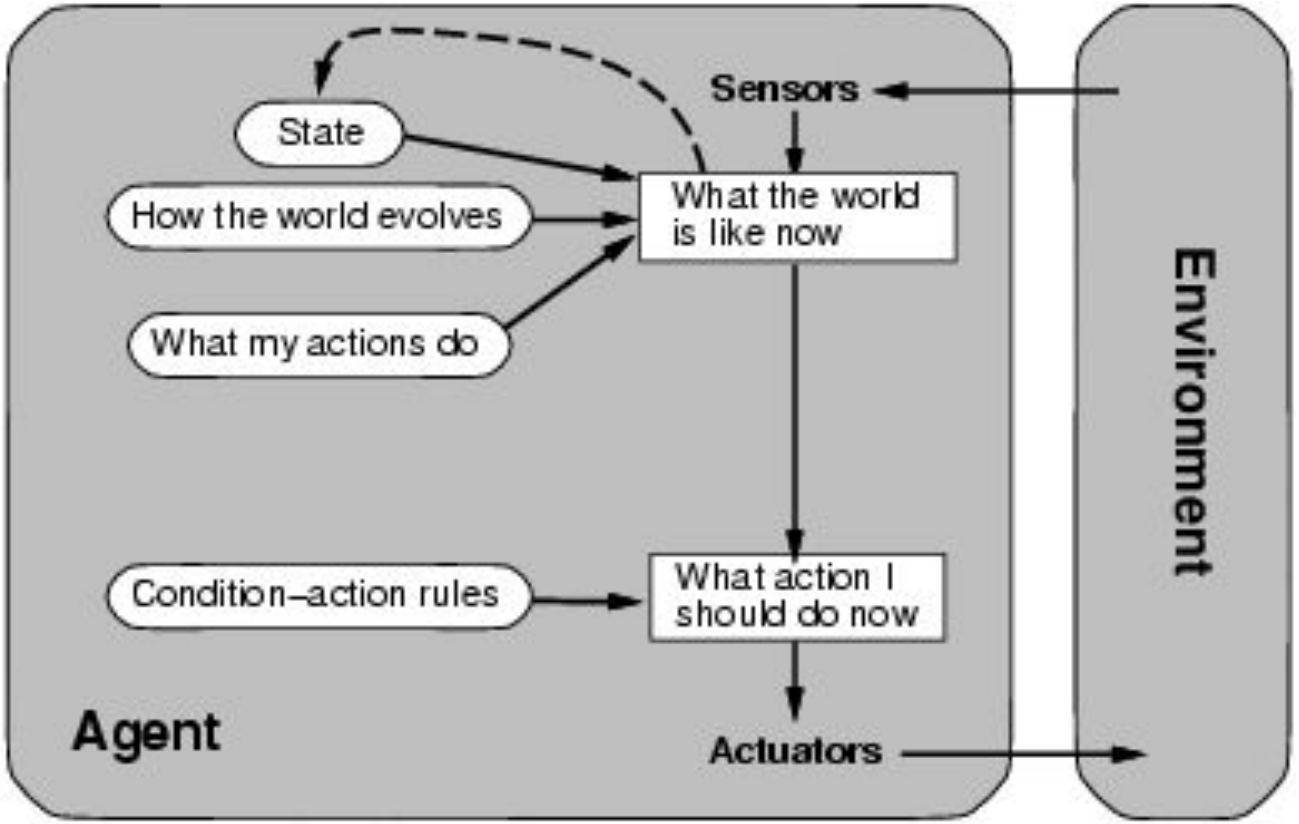
if STATUS = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left

# Model based reflex agents

- Agent to keep track of the part of the word it cant see now.
- Maintain some sort of internal state that depend on the percept history and thereby reflect at least some of unobservable aspect of current state.
- Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.

1. we need some information about how the world evolves independently of the agent—for example, that an overtaking car generally will be closer behind than it was a moment ago.
2. we need some information about'how the agent's own actions affect the world—for example, that when the agent changes lanes to the right, there is a gap (at least temporarily) in the lane it was in before, or that after driving for five minutes northbound ; on the freeway one is usually about five miles north of where one was five minutes ago.

# Model-based reflex agents




Current percept is combined with old internal state to generate the updated description of the current state.

# Model-based reflex agents

- **Function Reflex agent with state returns an** *action*
- **static:**
- *state* <— Update-state(state,action,percept)
- rule <- RULE-MATCH(state, *rules)*
- *action* <- RULE-ACTiON[rule]
- **return** *action*
- Update state  is responsible for creating the new internal state description as well as interpreting the new percept in the light of existing knowledge about the state.

Action may depend on history or unperceived aspects of the world.

- Need to maintain internal world model.

Example:

Agent: robot vacuum cleaner

Environment: dirty room, furniture.

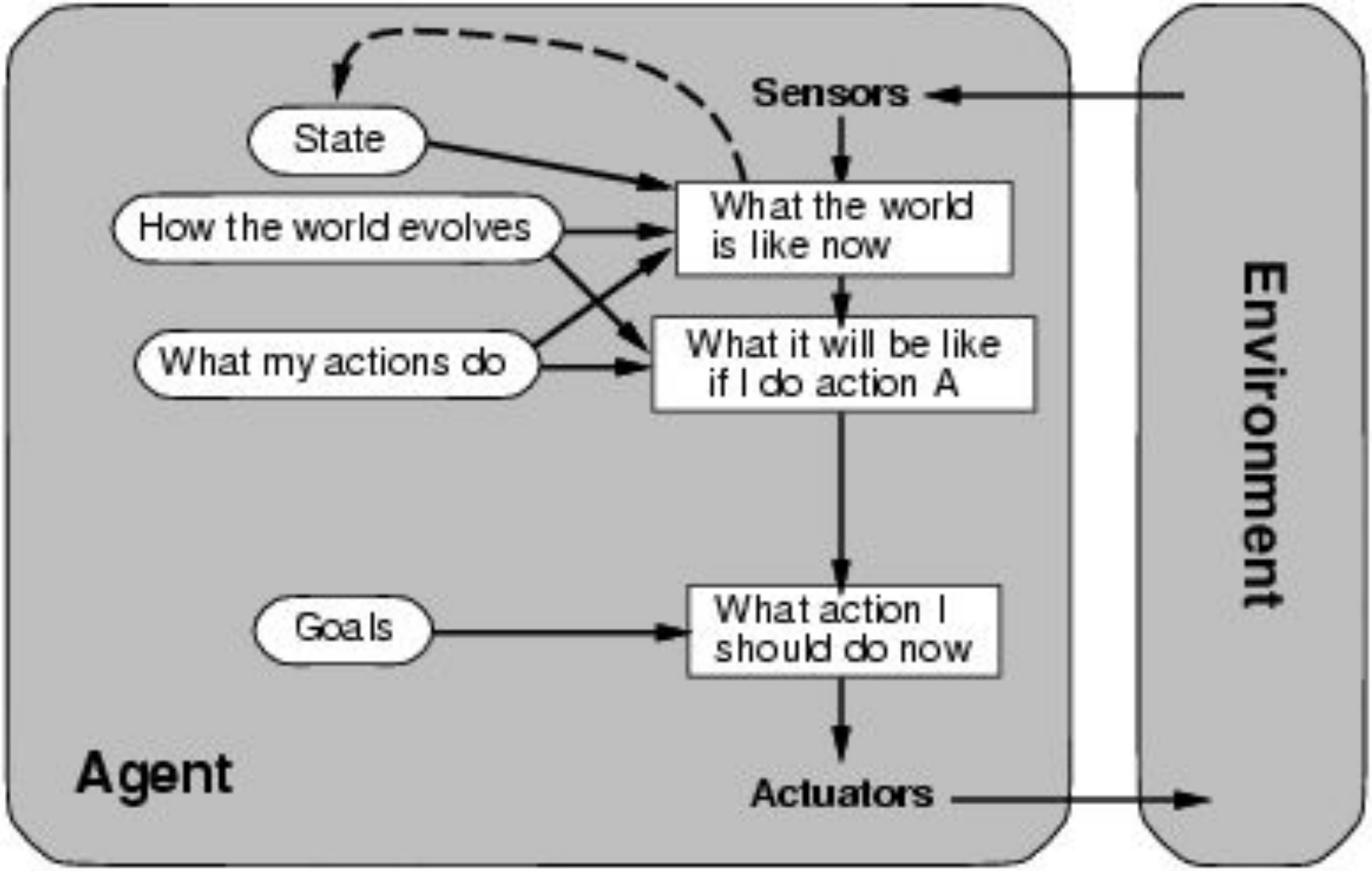
Model: map of room, which areas already cleaned.

Sensor/model trade-off.

# Goal-based agents

- Knowing about the current state of the environment is not always enough to decide what to do. For example, at a road junction, the taxi can turn left, right, or go straight on. The right decision depends on where the taxi is trying to get to.
- Agent needs some sort of **goal** information, which describes situations that are desirable for example, being at the passenger's destination.
- Agent Program choose the action that achieve the goal
- Searching and Planning are the subfields of AI devoted to finding action sequence that achieve the agents' goal.
- Decision making is different from condition-action rule
- Reflex agent use built in rule to map percept to action , Goal based could reason
- Although the goal based agent appeared less efficient it is more flexible because knowledge that support its decision is represented explicitly and can be modified.

# Goal-based agents



## goal-based agent

```
def action (agent, environment):
let state = agent. s e n s e (environment)
for action in agent.available Actions :
let resultingState = agent.do (action, state)
if resultingState == agent. goal:
return action
```

# Utility-based agents

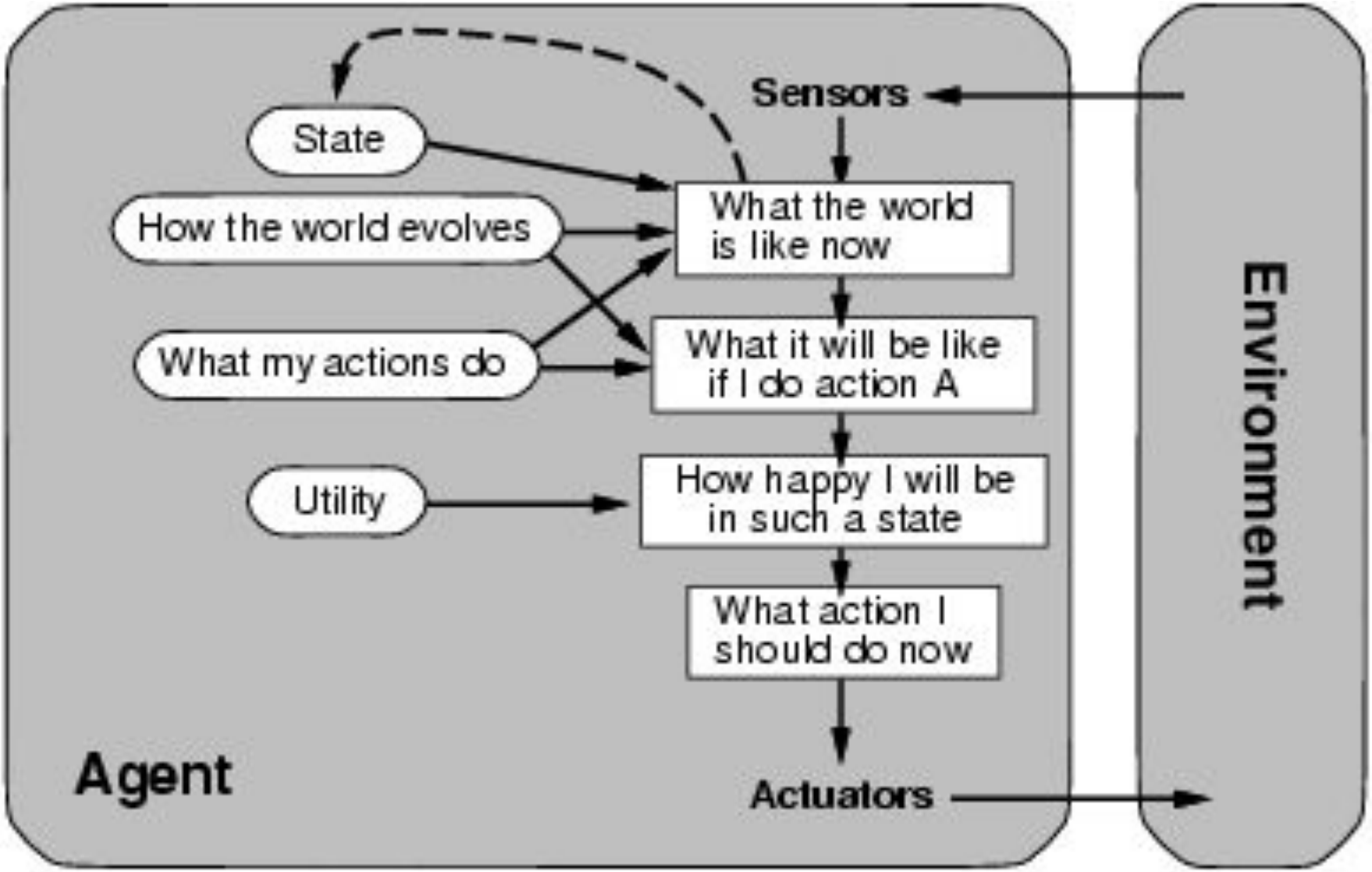
- Goals alone are not really enough to generate high-quality behavior.

***action sequences that will get the taxi to its destination, thereby achieving the goal, but some are quicker, safer, more reliable, or cheaper than others.***

- Utility is therefore a function that maps a state onto a real number.
- A complete specification of the utility function allows rational decisions in two kinds of cases where goals have trouble:

1. when there are conflicting goals only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate trade-off.
2. when there are several goals that the agent can aim for, none of which can *be* achieved with certainty, utility provides a way in which the likelihood of success can be weighed up against the importance of the goals.

# Utility-based agents



```
def action ( agent , environment ) :
let state = agent . sense ( environment )
let maxUtility = −Infinity
let maxUtility State = None
for action in agent.available Actions :
let resulting State = agent. do (action, state)
let resulting StateUtility = agent. getUtility ( r e s u l t i n g State )
if resulting StateUtility > maxUtility :
maxUtility = resulting StateUtility
maxUtility State = resulting State
return maxUtility State
```

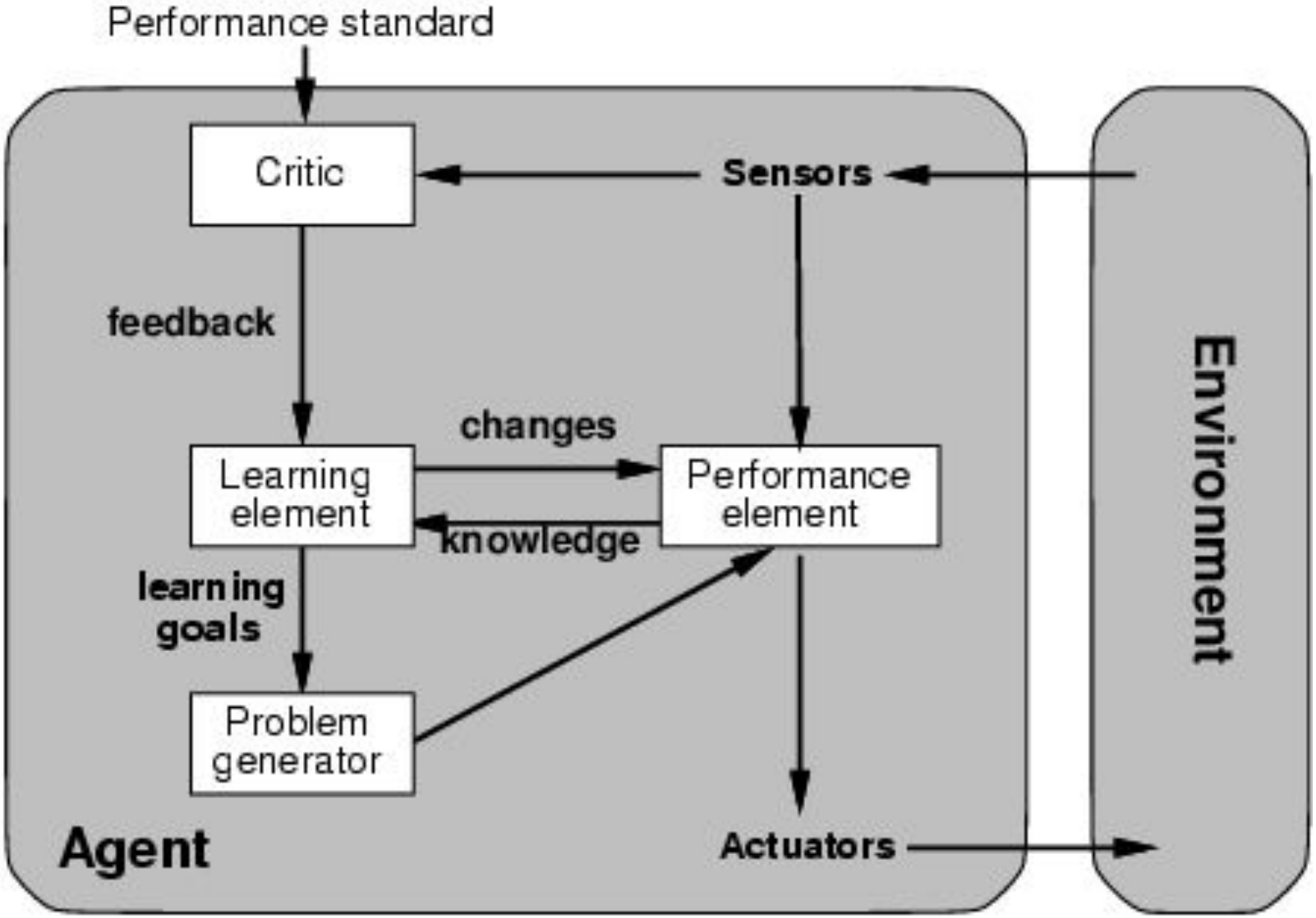
# Learning agents

- All agents can improve their performance through learning

1. Learning element-which is responsible for making improvement
2. Performance element-selecting external actions: it takes percepts and decide on actions
3. Critic how agent is doing and how the performance should be modified to do better.
4. Problem generator suggest exploratory actions that lead to new and informative experiences.

- Critic tells the learning element how well the agent is doing w.r.t performance standard

# Learning agents



# Compare

- A simple reflex agent. It works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.
- A reflex agent with internal state. It works by finding a rule whose condition matches the current situation (as defined by the percept and the stored internal state) and then doing the action associated with that rule.
- Goal based keep track of world state as well as set goal it is trying to achieve and choose an action that will lead to achievement of its goal
- Utility based uses utility function that measures its preferences among states of the world and chooses the action that leads to the best expected utility. Expected utility is calculated by averaging all possible outcome state, weighted by probability of outcome .

For each of the following agents develop a
1. PEAS description of task environment ,
2. Environment type and
3. Write agent

- Robot Soccer Player
- Internet book shopping agents
- Autonomous Mars rover
- Mathematician's theorem proving assistant