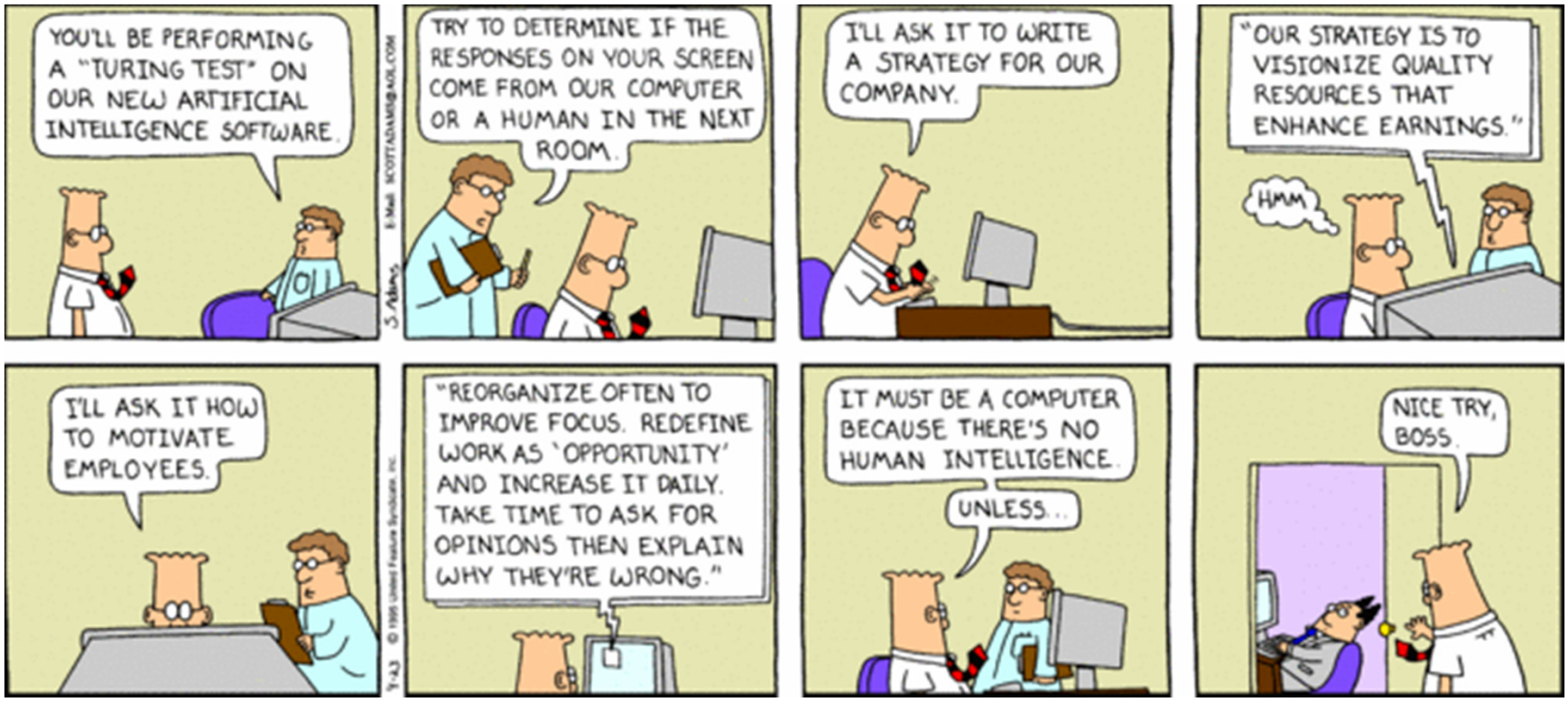
Intelligent Agents

CS4881 Jay Urbain, Ph.D.

Intelligent Agents



Outline

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

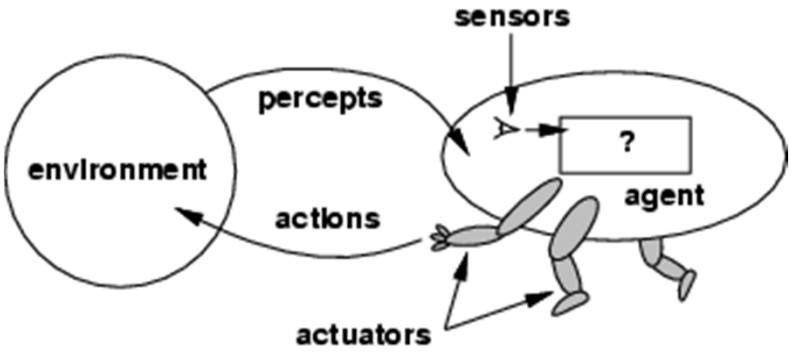
Agents

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

- 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

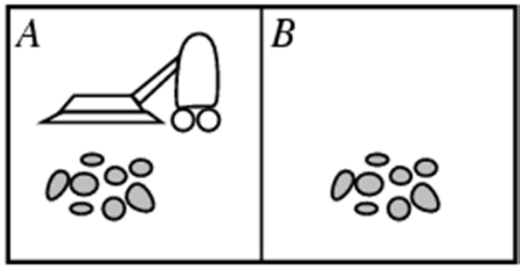


- The agent function maps from percept histories to actions:

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

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

Vacuum-cleaner world – *Roomba!*



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

A vacuum-cleaner agent

Percept Sequence	**Action**
[A, Clean]	*Right*
[A, Dirty]	*Suck*
[B, Clean]	*Left*
[B, Dirty]	*Suck*
[A, Clean] [A, Clean]	*Right*
[A, Clean] [B, Dirty]	*Suck*
…	

Rational agents – do the right thing

- 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.

- *How to define success?*

Rational agents – Performance Measure

- *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.
  - *How to select a performance measure?*

Rational agents – Performance Measure

- *How to select a performance measure?*
- Better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave – *why?*

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
  - whatever built-in knowledge the agent has
  - actions agent can perform

Rational agents

- Rationality is distinct from *omniscience* (all-knowing with infinite knowledge).
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, *exploration*).
- An agent is *autonomous* if its behavior is determined by its own experience (with ability to learn and adapt).
- *Rational agent design needs a definition of the task environment….*

PEAS Description

- PEAS:
  - Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design.
- Consider the task of designing an automated taxi driver:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS - Automated taxi driver

- Must first specify the setting for intelligent agent design
- Consider 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
- *Note: The more restricted the environment, the easier the design problem!*

PEAS - Medical diagnosis system

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

PEAS - Medical diagnosis system

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

PEAS - Part-picking robot

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

PEAS - Part-picking robot

- 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 - Interactive English tutor

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

PEAS - Interactive English tutor

- 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" (agent perceiving then performing a single action), and the choice of action in each episode depends only on each atomic episode itself.

Environment types (cont.)

- Static (vs. dynamic):
  - The environment is unchanged while an agent is deliberating. (semi-dynamic if the environment 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. multi-agent):
  - 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			

- Notes:
  - The environment type largely determines the agent design
  - *The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent*

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

Agent functions and programs

- An agent is completely specified by the *agent function* mapping percept sequences to actions.
- Agent function (or a small equivalence class) is *rational.*
- Aim: How to implement the rational agent function concisely?
- *Table lookup?*

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

  - *Ability to generalize?*

Table-lookup agent

```
function tableDrivenAgent(percept) returns action
  static:   percepts- a sequence, initially empty
            table- a table of actions, indexed by
            percept sequences, initially specified

  percepts.append(percept)
  action <- lookup(percepts, table)
  return action
```

Vacuum-cleaner agent

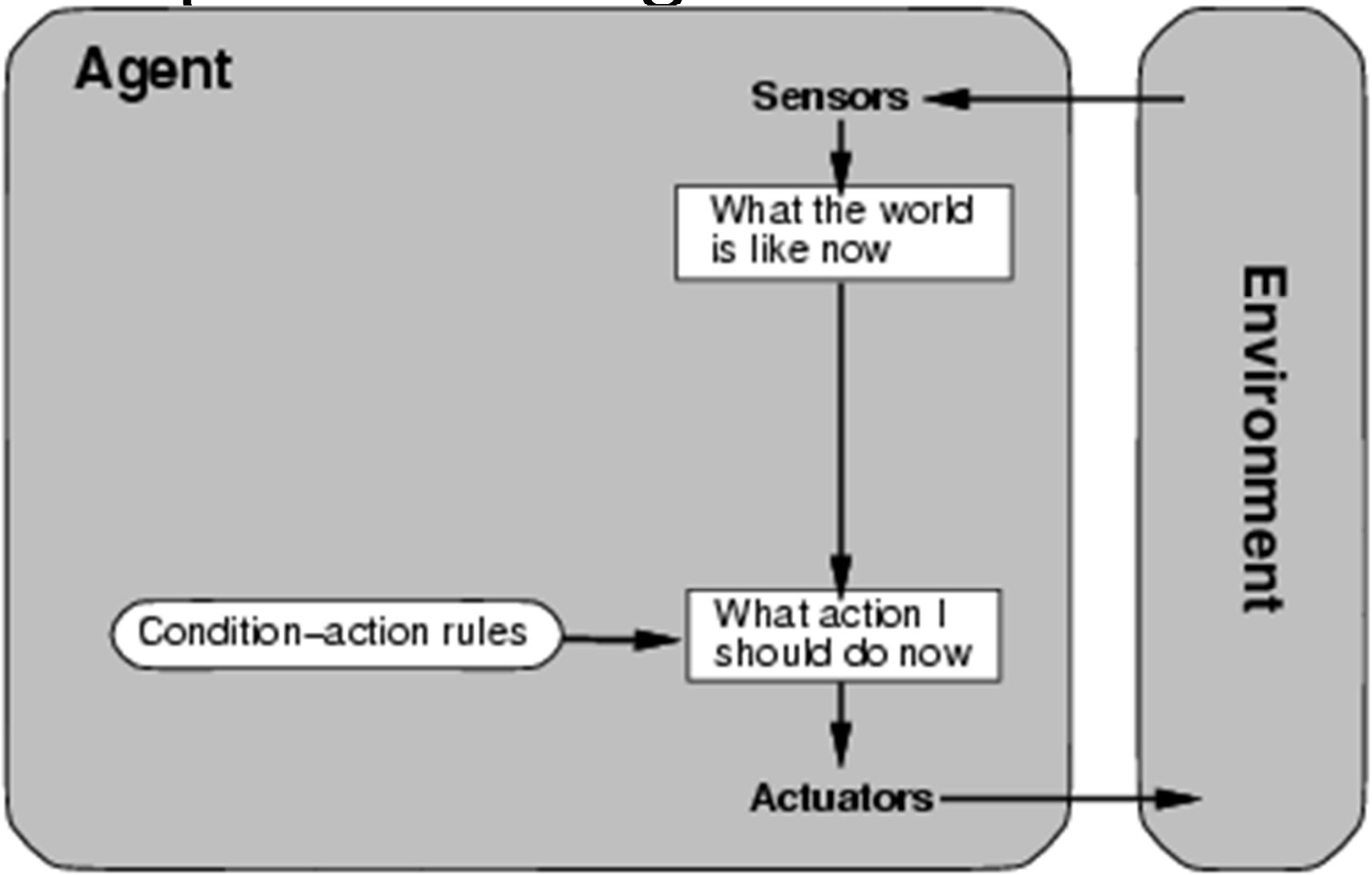
```
function roombaAgent(location, status) returns
   action

   if(status = dirty) return suck
   else if (location = A) return right
   else if (location = B) return left
```

Agent types

- Four basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple reflex agents



Simple reflex agents

```
function simpleReflexAgent(percept) returns action
  static: rules, a set of condition-
      action rules
  state <- interpretInput(percept)
  rule <- ruleMatch(state, rules)
  action <- ruleAction[rule]
  return Action
```

Advantages / disadvantages?

Simple reflex agents

Advantage:

- Simple!

Disadvantages:

- Will work only if the correct decision can be made on the basis of only the current percept, i.e., only if the environment is *fully observable.*

Examples …

- Car braking vs. directional
- Agents deprived of one of their sensors – Roomba w/ no location sensor (add randomness to break out of loop).
- *Is randomness rational?*

Model-based reflex agents

Most effective way to handle *partial observability* is for the agent to keep track of the part of the world it can't see now.

Agent needs to keep some sort of internal state. E.g., as you change lanes you need to remember if there were other cars.

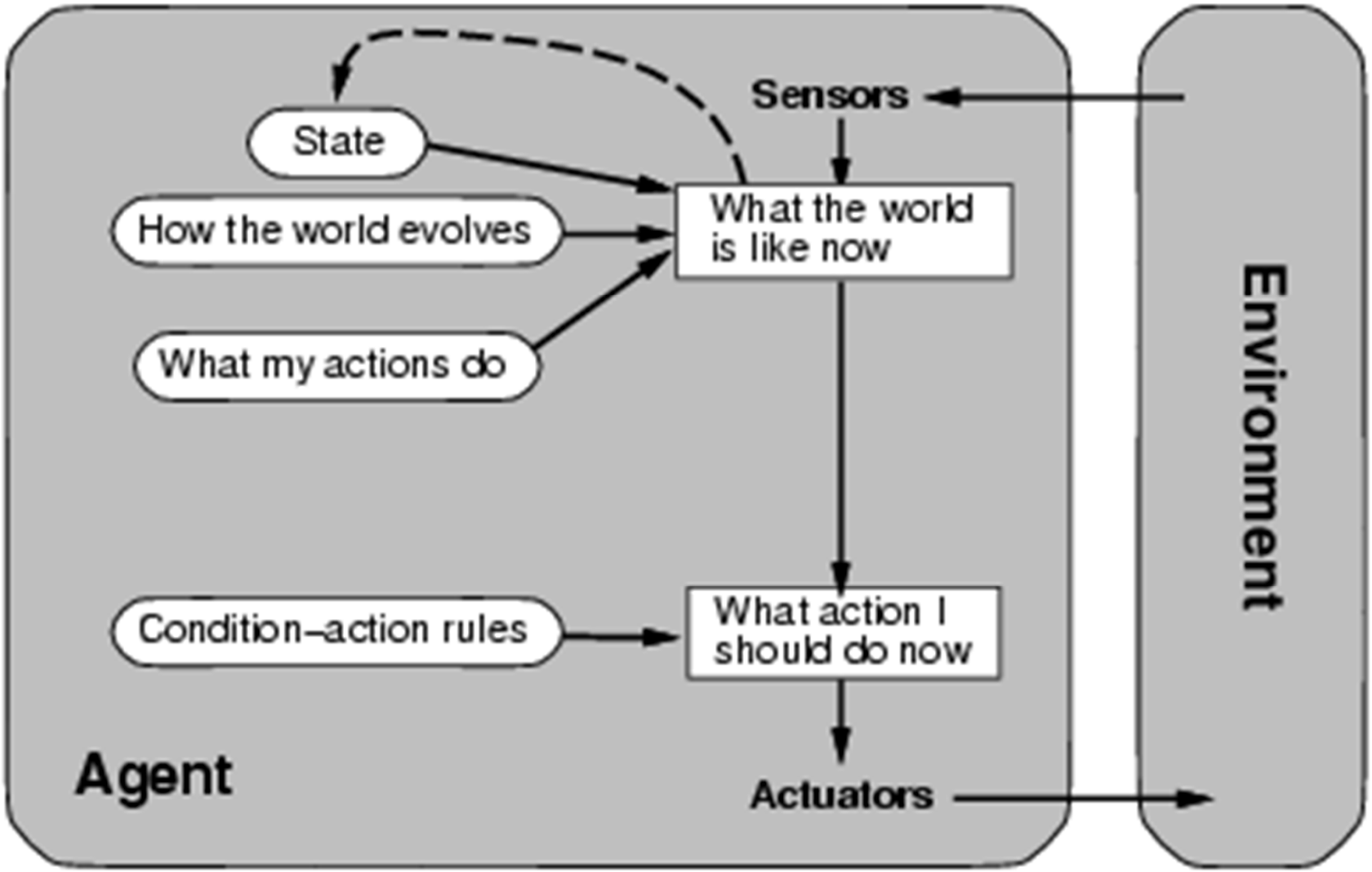
Need to keep track of two kinds of knowledge:

- information about how the world evolves
- information about how the agent's own actions effect the world. E.g., steering

Knowledge about how the world works is called a ***model*** of the world

An agent that uses such a model is a ***Model-based agent***.

Model-based reflex agents



Model-based reflex agents

```
function reflexAgentWithState(percept) returns
  action
  static:
      state, desc of current world state
      rules, a set of condition-action rules
      action, most recent action, initially none

  state <- updateState(state, action, percept)
  rule <- ruleMatch(state, rules)
  action <- ruleAction[rule]
  return Action
```

Model-based reflex agents

Notes on ***updateState():***

- Create new internal state description.
- Interpret the new percept in light of existing knowledge about the state.

- Uses info about how the world evolves to keep track of unseen parts of the world, *and* must know about what the agent's actions do to the state of the world.

Goal-based agents

Knowing about the *current state* of the environment is not always enough to decide what to do.

E.g., a fork in the road… left, right, or straight?

How do you know what to do?

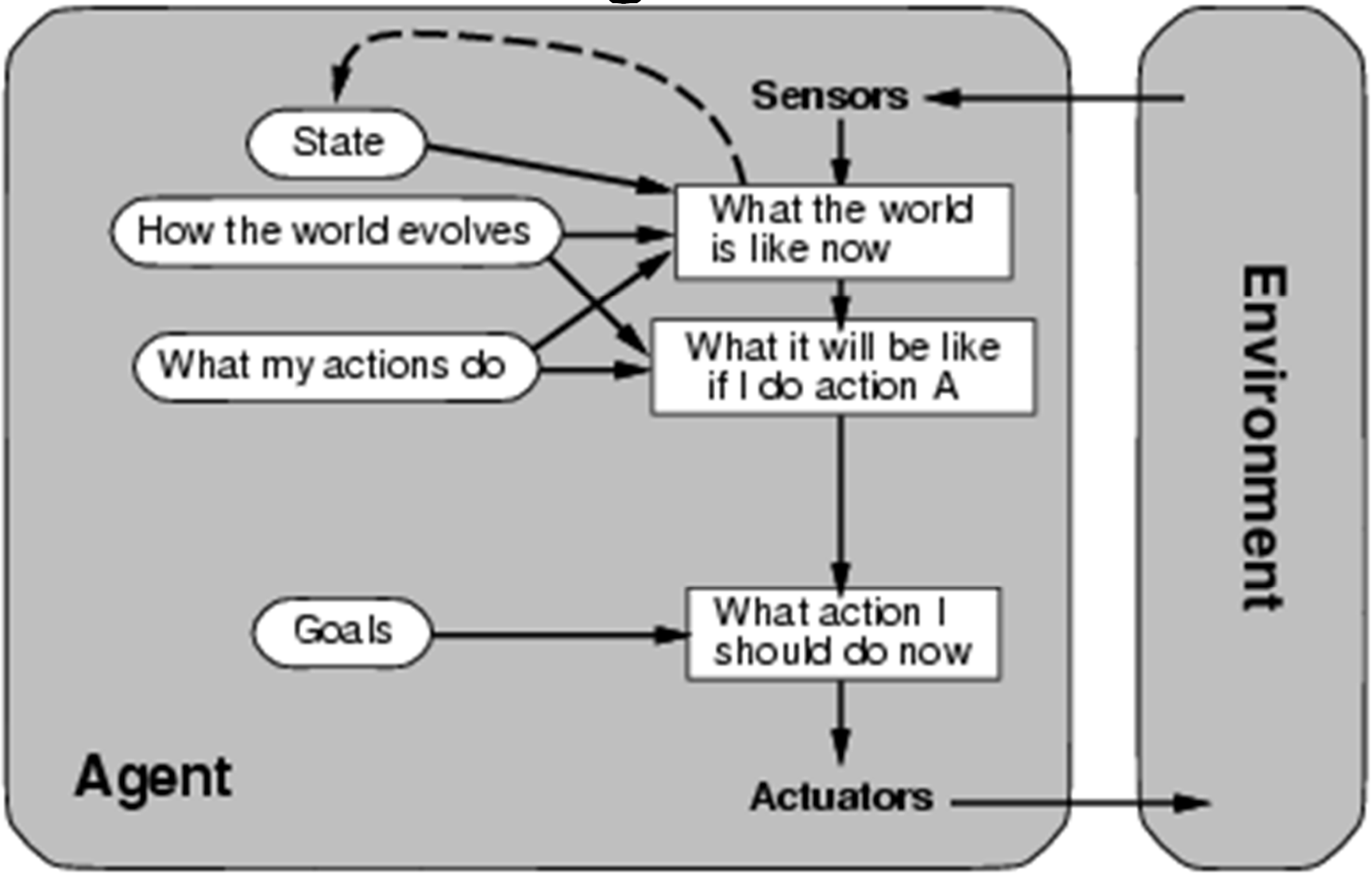
Goal-based agents

How do you know?

- As well as current state description, your agent needs a ***goal*** describing situations that are desirable
  - E.g., being at the passengers destination.
  - Are you trying to get to your mother-in-law's house or avoid it?

- Program can combine this with the info about the results of possible actions.
- Goal-based action is easy when goals are satisfied immediately with a single action.
- Difficult when agent has to consider a sequence of actions – requires ***search*** and ***planning*** (more next week).

Goal-based agents



Goal-based agents

Notes:

Decision making in goal-based agents is fundamentally different from the *condition-action* rules described earlier:

- *decision making involves consideration of the future*
  - What will happen if I do something?
  - Will it make me happy?, i.e., work towards satisfying a goal.

- Goal-based agent appears less efficient, however it is more flexible because knowledge that supports its decisions is represented explicitly and can be modified.
- Agent can more easily update its knowledge of how effectively its brakes will operate causing all relevant behaviors to be updated.

Utility-based agents

- Goals alone are not realy enough to generate high-quality behavior in most environments. E.g.,
  - Many action sequences that will get our taxi to its destination, _*but*_ some are quicker, safer, more reliable, or cheaper.

- Goals just provide a crude distinction between *happy* vs. *unhappy*.

- More general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent if they were achieved
  - If one world state is preferred over another, then it has a higher ***utility***.

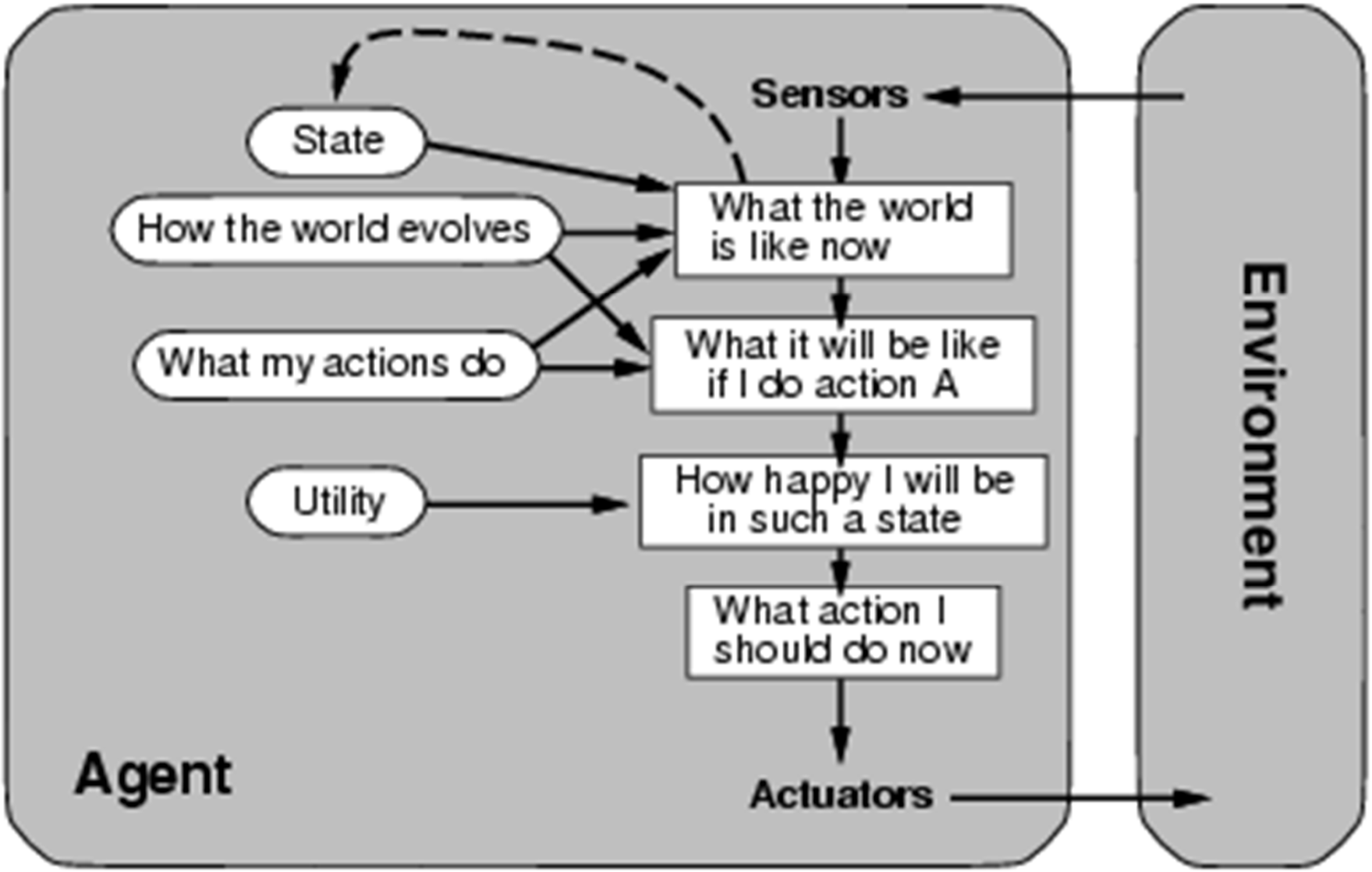
Utility-based agents

- Utility function – maps a state (or sequence of states) onto a real number, which describes the associated degree of happiness.

- Specification of utility function allows rational decisions in two kinds of cases:
  1. When there are conflicting goals (only some of which can be achieved).
  2. When there are several goals (none of which can be achieved with certainty).

- Any rational agent must behave as if it possesses a utility function whose expected value it tries to maximize.
  - ***Agents that possess a utility function can make rational decisions.***
  - ***Turns into local constraint satisfaction problem.***

Utility-based agents



Learning agents

- So far, we've just described agents with various methods for selecting actions.

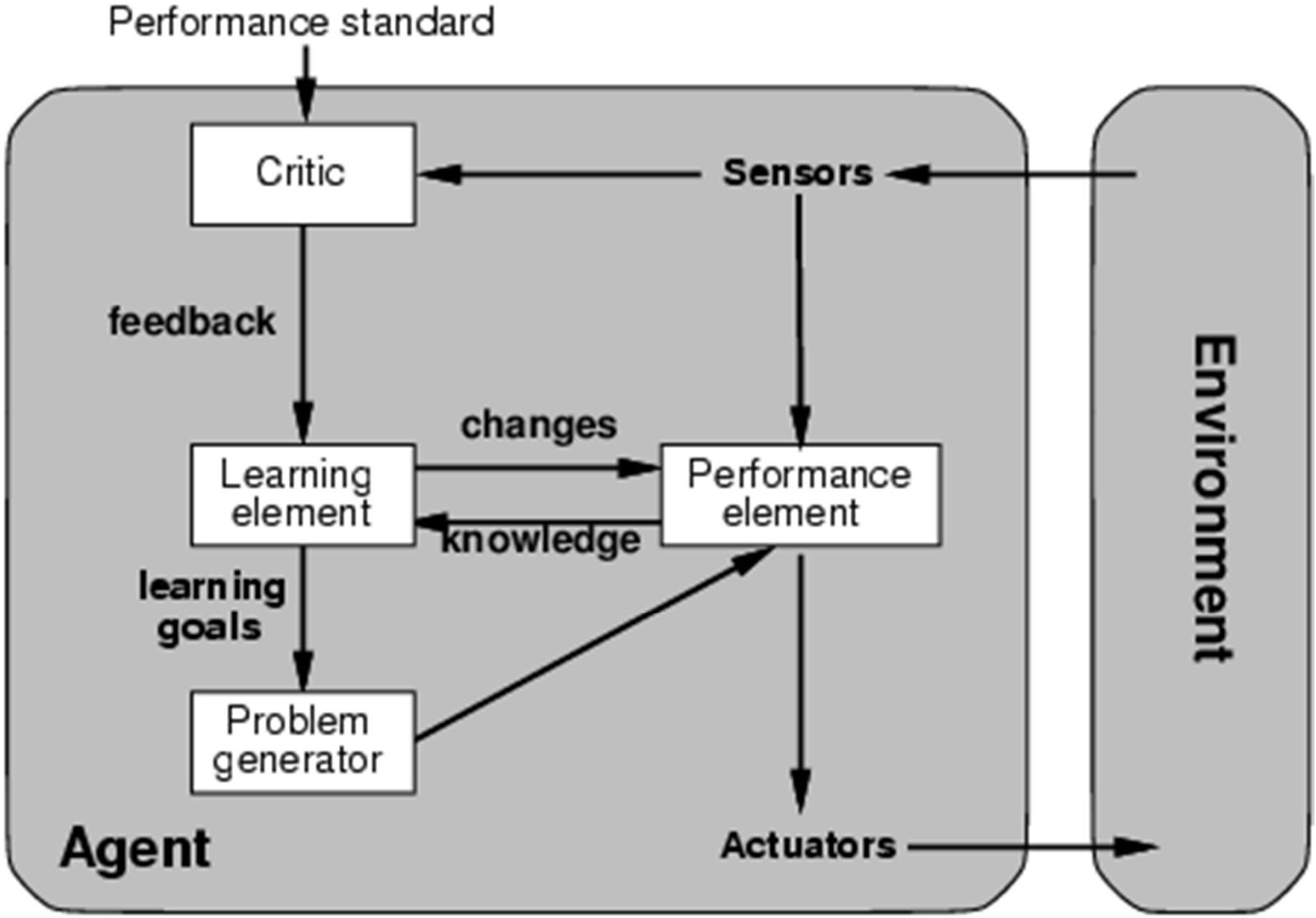
- How do agent programs come into *being*?
  - Generating logic manually becomes intractable for many tasks.
  - Need to build learning machines and then teach them.
  - Preferred method in many areas of AI.

- *What other advantages would a learning agent possess?*

Learning agents

- What other advantages would a learning agent possess?
  - Allows agent to operate in initially unkown environments _*and*_ become more competent with experience
- Learning agent can be divided into conceptual components:
  - Learning element
  - Performance element
  - Critic
  - Problem generator

Learning agents



Learning agents

- Learning element
  - Responsible for making improvements
- Performance element
  - Responsible for selecting external actions
  - Previously considered as the entire agent
- Critic
  - Provides feedback to learning agent on how the agent is doing and determines how the performance element should be modified to do better in the future

Learning agents

- Problem generator
  - Suggests actions that will lead to new and potentially informative experiences, e.g., taxi trying a new route to the airport.
  - If agent is willing to explore a little, and do some sub-optimal actions in the short-run it may discover better actions for the long-run.
  - If the performance element had its way, it would keep doing the actions that are best given what it knows.
  - *What implications does this have for life?*

Learning agents

- Example taxi ride:
  - *Performance element* contains collection of knowledge and procedures for selecting driving actions.
  - Taxi goes out on road and drives using *performance element.*
  - *Critic* observes the world and passes info along to learning element.
  - From these experiences, *learning element* formulates new rules
  - *Problem generator* may identify behavior in need of improvement and suggest experiments

Learning agents

- Learning in intelligent agents can be seen as a process of modification of each component of the agent to bring the components into closer agreement with available feedback.

Summary

- **Agent** – something that percieves and acts in an environment.
- **Agent function** – specifices action taken in response to a percept.
- **Performance measure** – evalutes behavior of the agent in an environment.
- **Rational agent** – acts to maximize performance measure.
- **Task enviroment** – PEAS
- **Agent program** – implements agent function.
- **Simple reflex agents** – respond directly to percepts.
- **Model-based agents** – maintain internal state to track aspects of the world.
- **Goal-based agents** – act to achive their goals.
- **Utility-based agents** - try to maximize their expected happiness.
- All agents can improve their performance though **learning.**