

Intelligent Agents



CHAPTER 2
Oliver Schulte
Summer2011

Outline

2

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

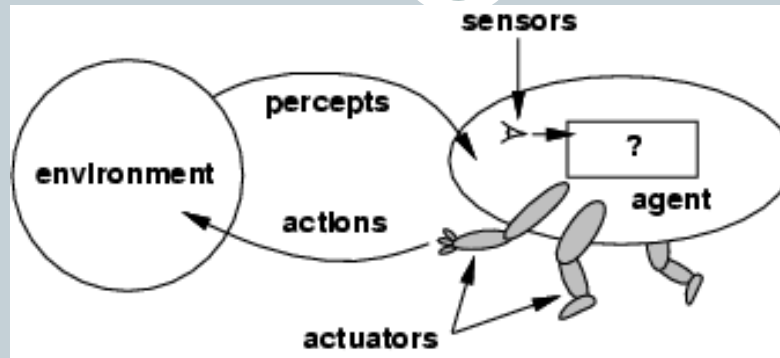
Agents

3

- 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

4



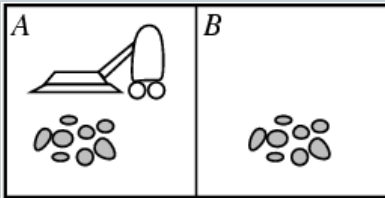
- The **agent function** maps from percept histories to actions:

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

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

Vacuum-cleaner world

5



Demo:

<http://www.ai.sri.com/~oreilly/aima3ejava/aima3ejavademos.html>

- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left, Right, Suck, NoOp*
- *Agent's function* → *look-up table*
 - *For many agents this is a very large table*

| Percept sequence | Action |
|------------------------|--------------|
| [A, Clean] | <i>Right</i> |
| [A, Dirty] | <i>Suck</i> |
| [B, Clean] | <i>Left</i> |
| [B, Dirty] | <i>Suck</i> |
| [A, Clean], [A, Clean] | <i>Right</i> |
| [A, Clean], [A, Dirty] | <i>Suck</i> |
| ⋮ | ⋮ |

Rational agents

6

- **Rationality**
 - Performance measuring success
 - Agents prior knowledge of environment
 - Actions that agent can perform
 - Agent's percept sequence to date
- **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.
-

Examples of Rational Choice

7

- See File: [intro-choice.doc](#)

Rationality

8

- Rational is different from omniscience
 - Percepts may not supply all relevant information
 - E.g., in card game, don't know cards of others.
- Rational is different from being perfect
 - Rationality maximizes expected outcome while perfection maximizes actual outcome.

Autonomy in Agents



The **autonomy** of an agent is the extent to which its behaviour is determined by its own experience, rather than knowledge of designer.

- **Extremes**
 - No autonomy – ignores environment/data
 - Complete autonomy – must act randomly/no program
- **Example: baby learning to crawl**
- **Ideal: design agents to have some autonomy**
 - Possibly become more autonomous with experience

PEAS

10

- PEAS: Performance measure, Environment, Actuators, Sensors
- 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

11

- 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

12

- 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

13

- **Fully observable** (vs. partially observable)
- **Deterministic** (vs. stochastic)
- **Episodic** (vs. sequential)
- **Static** (vs. dynamic)
- **Discrete** (vs. continuous)
- **Single agent** (vs. multiagent):

Fully observable (vs. partially observable)

14

- Is everything an agent requires to choose its actions available to it via its sensors? Perfect or Full information.
 - If so, the environment is fully accessible
- If not, parts of the environment are inaccessible
 - Agent must make informed guesses about world.
- In decision theory: perfect information vs. imperfect information.

| | | | | | |
|-------------------|--------------|-------------------|--------------------|---------------------------|-----------------------|
| Cross Word | Poker | Backgammon | Taxi driver | Part picking robot | Image analysis |
| Fully | Partially | Partially | Partially | Fully | Fully |

Deterministic (vs. stochastic)

15

- Does the change in world state
 - Depend only on current state and agent's action?
- Non-deterministic environments
 - Have aspects beyond the control of the agent
 - Utility functions have to guess at changes in world

| | | | | | |
|-------------------|--------------|-------------------|--------------------|---------------------------|-----------------------|
| Cross Word | Poker | Backgammon | Taxi driver | Part picking robot | Image analysis |
| Deterministic | Stochastic | Stochastic | Stochastic | Stochastic | Deterministic |

Episodic (vs. sequential):

16

- Is the choice of current action
 - Dependent on previous actions?
 - If not, then the environment is episodic
- In non-episodic environments:
 - Agent has to plan ahead:
 - ✦ Current choice will affect future actions

Cross Word

Poker

Backgammon

Taxi driver

Part picking robot

Image analysis

Sequential Sequential

Sequential

Sequential

Episodic

Episodic

Static (vs. dynamic):

17

- **Static environments don't change**
 - While the agent is deliberating over what to do
- **Dynamic environments do change**
 - So agent should/could consult the world when choosing actions
 - Alternatively: anticipate the change during deliberation OR make decision very fast
- **Semidynamic: If the environment itself does not change with the passage of time but the agent's performance score does.**

| | | | | | |
|-------------------|--------------|-------------------|--------------------|---------------------------|-----------------------|
| Cross Word | Poker | Backgammon | Taxi driver | Part picking robot | Image analysis |
| Static | Static | Static | Dynamic | Dynamic | Semi |

Another example: off-line route planning vs. on-board navigation system

Discrete (vs. continuous)

18

- A limited number of distinct, clearly defined percepts and actions vs. a range of values (continuous)

| | | | | | |
|-------------------|--------------|-------------------|--------------------|---------------------------|-----------------------|
| Cross Word | Poker | Backgammon | Taxi driver | Part picking robot | Image analysis |
| Discrete | Discrete | Discrete | Conti | Conti | Conti |

Single agent (vs. multiagent):

19

- An agent operating by itself in an environment or there are many agents working together

Cross Word

Single

Poker

Multi

Backgammon

Multi

Taxi driver

Multi

Part picking robot

Single

Image analysis

Single

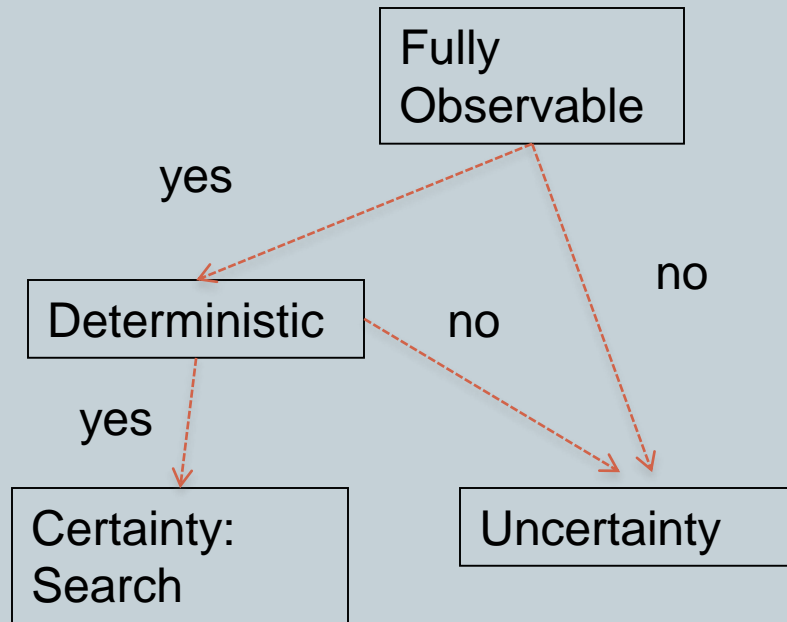
Summary.



| | Observable | Deterministic | Episodic | Static | Discrete | Agents |
|---------------------------|-------------------|----------------------|-----------------|---------------|-----------------|---------------|
| Cross Word | Fully | Deterministic | Sequential | Static | Discrete | Single |
| Poker | Fully | Stochastic | Sequential | Static | Discrete | Multi |
| Backgammon | Partially | Stochastic | Sequential | Static | Discrete | Multi |
| Taxi driver | Partially | Stochastic | Sequential | Dynamic | Conti | Multi |
| Part picking robot | Partially | Stochastic | Episodic | Dynamic | Conti | Single |
| Image analysis | Fully | Deterministic | Episodic | Semi | Conti | Single |

Choice under (Un)certainty

21



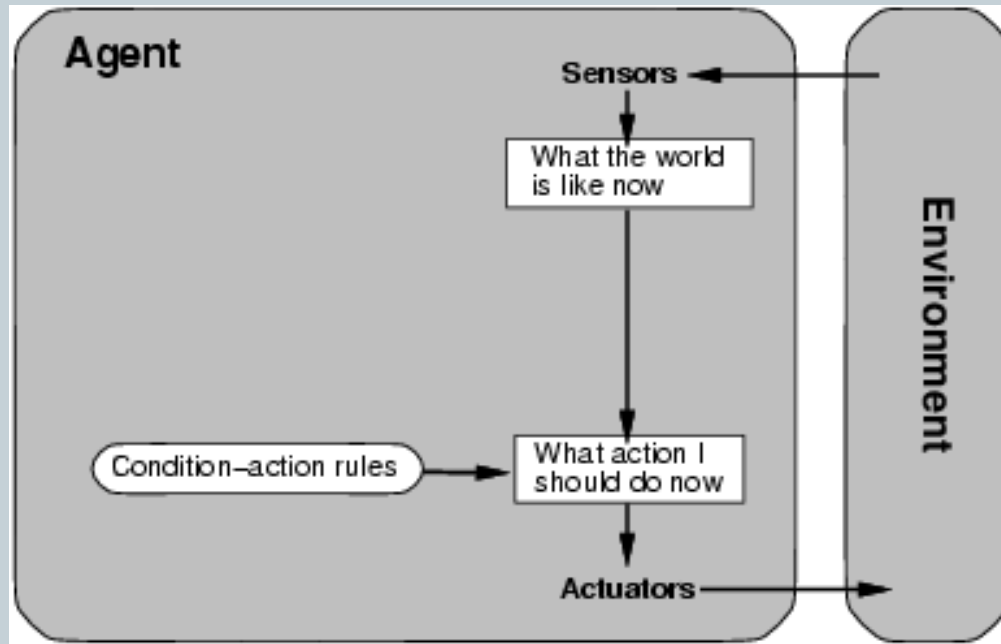
Agent types

22

- Four basic types in order of increasing generality:
 - Simple reflex agents
 - Reflex agents with state/model
 - Goal-based agents
 - Utility-based agents
 - All these can be turned into learning agents
 - <http://www.ai.sri.com/~oreilly/aima3ejava/aima3ejavademos.html>

Simple reflex agents

23



```
function REFLEX-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
```

Simple reflex agents

24

- Simple but very limited intelligence.
- **Action does not depend on percept history, only on current percept.**
- Therefore no memory requirements.
- Infinite loops
 - Suppose vacuum cleaner does not observe location. What do you do given location = clean? Left of A or right on B -> infinite loop.
 - Fly buzzing around window or light.
 - Possible Solution: Randomize action.
 - Thermostat.
- Chess – openings, endings
 - Lookup table (not a good idea in general)
 - ✦ 35^{100} entries required for the entire game

States: Beyond Reflexes

25

- Recall the **agent function** that maps from percept histories to actions:

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

- An agent program can implement an agent function by maintaining an **internal state**.
- The internal state can contain information about the state of the external environment.
- The state depends on the history of percepts and on the history of actions taken:

$$[f: \mathcal{P}^*, \mathcal{A}^* \rightarrow \mathcal{S} \rightarrow \mathcal{A}] \text{ where } \mathcal{S} \text{ is the set of states.}$$

- If each internal state includes all information relevant to information making, the state space is **Markovian**.

States and Memory: Game Theory

26

- If each state includes the information about the percepts and actions that led to it, the state space has **perfect recall**.
- **Perfect Information** = Perfect Recall + Full Observability.

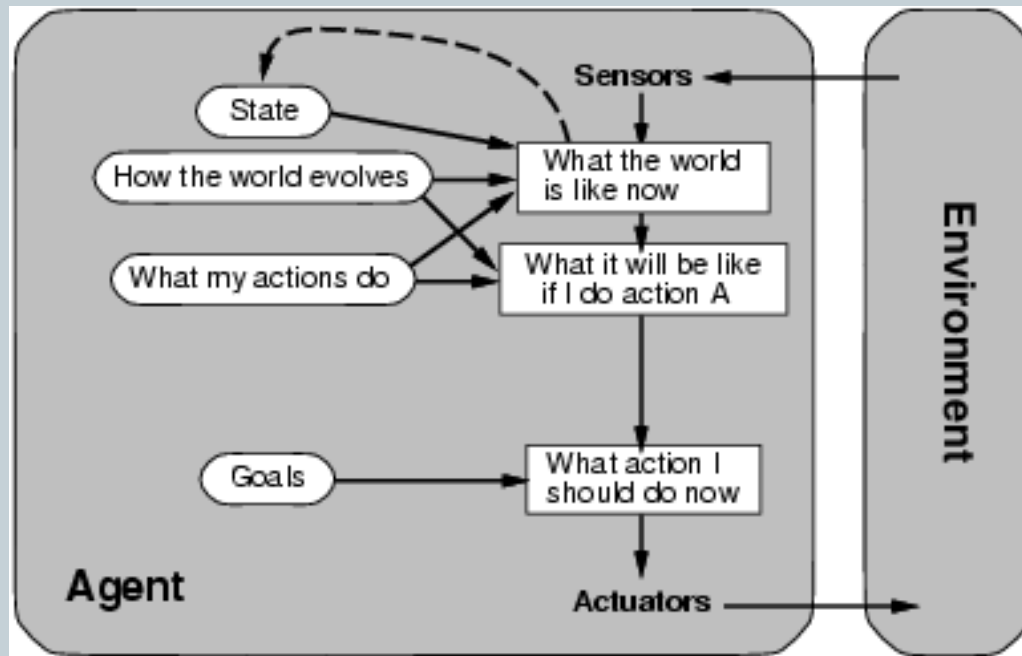
Goal-based agents

27

- knowing state and environment? Enough?
 - Taxi can go left, right, straight
- Have a goal
 - A destination to get to
- Uses knowledge about a goal to guide its actions
 - E.g., Search, planning

Goal-based agents

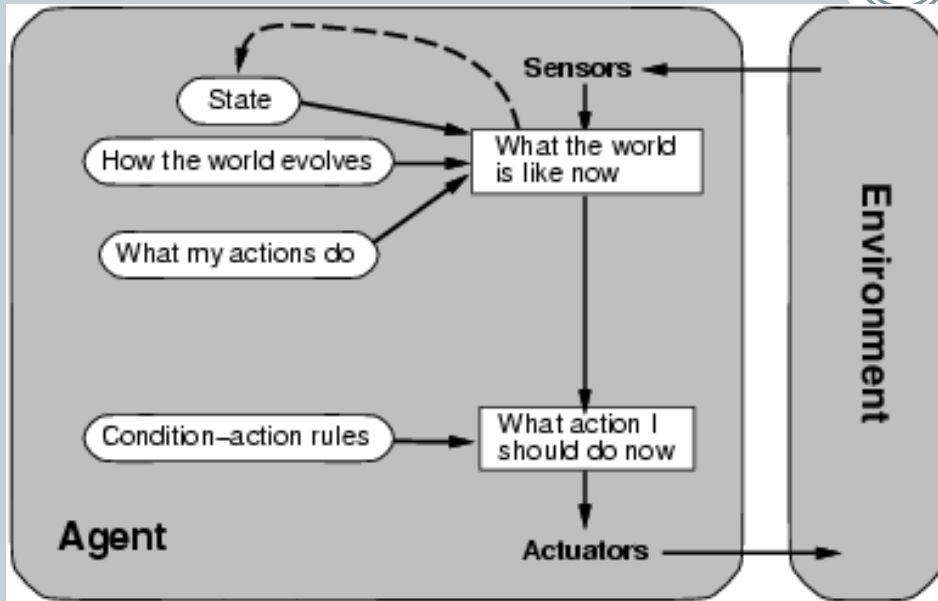
28



- Reflex agent breaks when it sees brake lights. Goal based agent reasons
 - Brake light -> car in front is stopping -> I should stop -> I should use brake

Model-based reflex agents

29



- Know how world evolves
 - Overtaking car gets closer from behind
- How agents actions affect the world
 - Wheel turned clockwise takes you right
- Model base agents update their state

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
```

```
  static: state, a description of the current world state  
          rules, a set of condition-action rules
```

```
  state ← UPDATE-STATE(state, percept)
```

```
  rule ← RULE-MATCH(state, rules)
```

```
  action ← RULE-ACTION[rule]
```

```
  state ← UPDATE-STATE(state, action)
```

```
  return action
```

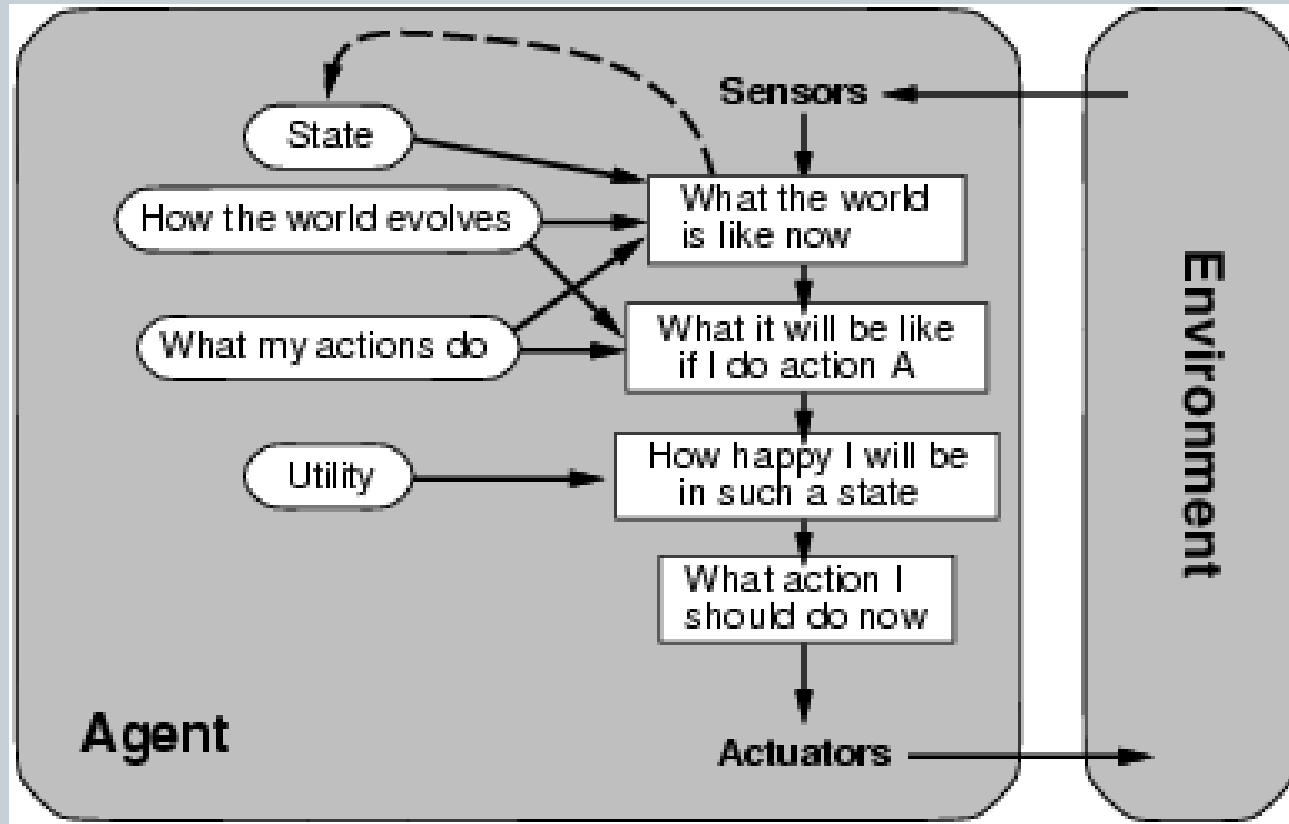
Utility-based agents

30

- Goals are not always enough
 - Many action sequences get taxi to destination
 - Consider other things. How fast, how safe.....
- A utility function maps a state onto a real number which describes the associated degree of “happiness”, “goodness”, “success”.
- Where does the utility measure come from?
 - Economics: money.
 - Biology: number of offspring.
 - Your life?

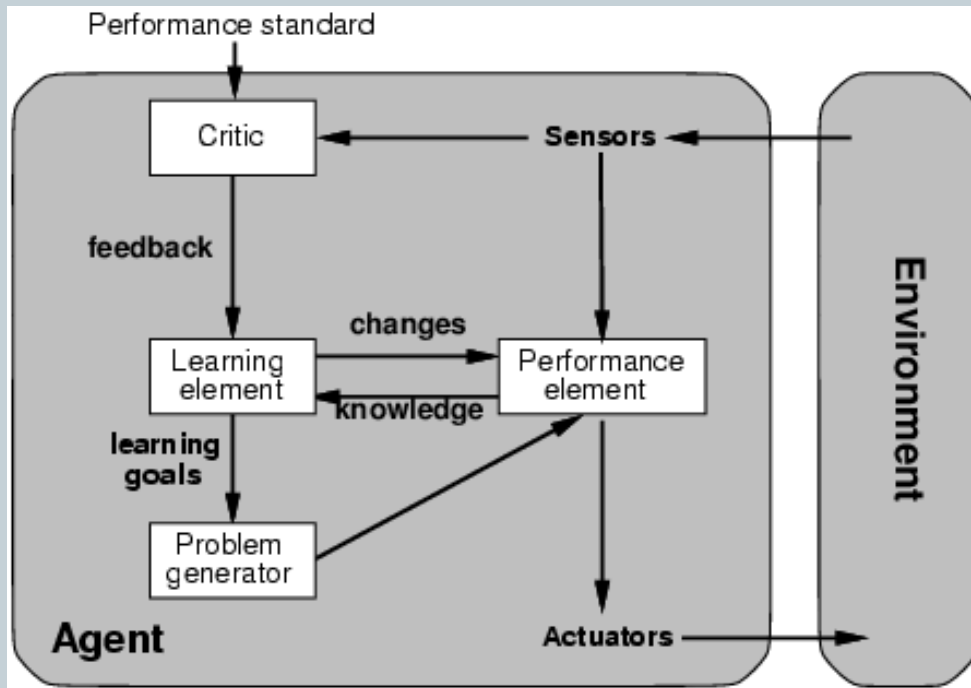
Utility-based agents

31



Learning agents

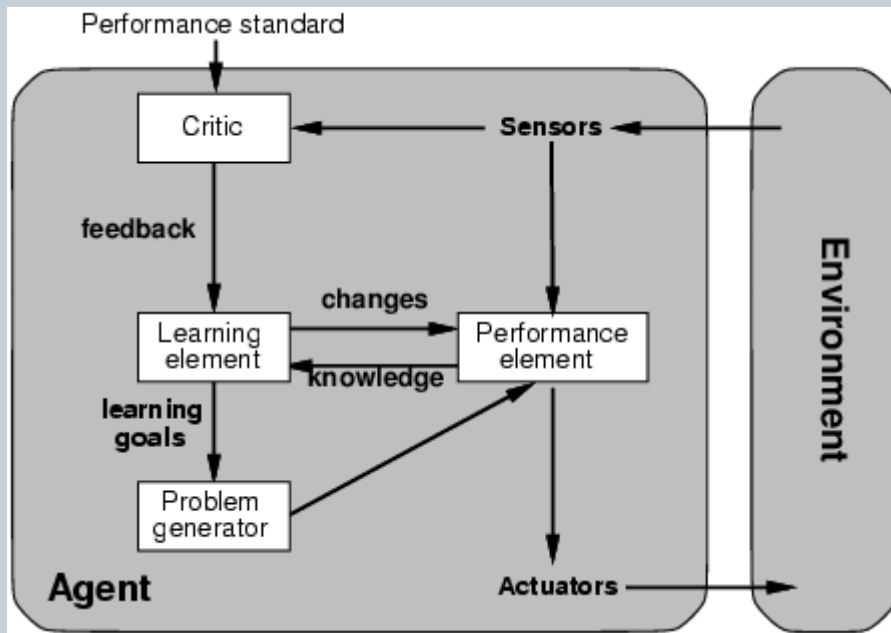
32



- Performance element is what was previously the whole agent
 - Input sensor
 - Output action
- Learning element
 - Modifies performance element.

Learning agents

33



- Critic: how the agent is doing
 - Input: checkmate?
 - Fixed
- Problem generator
 - Tries to solve the problem differently instead of optimizing.
 - Suggests **exploring** new actions -> new problems.

Learning agents(Taxi driver)

34

- Performance element
 - ✦ How it currently drives
- Taxi driver Makes quick left turn across 3 lanes
 - ✦ Critics observe shocking language by passenger and other drivers and informs bad action
 - ✦ Learning element tries to modify performance elements for future
 - ✦ Problem generator suggests experiment out something called Brakes on different Road conditions
- Exploration vs. Exploitation
 - ✦ Learning experience can be costly in the short run
 - ✦ shocking language from other drivers
 - ✦ Less tip
 - ✦ Fewer passengers