323 ARTIFICIAL INTELLIGENCE & EXPERT SYSTEMS

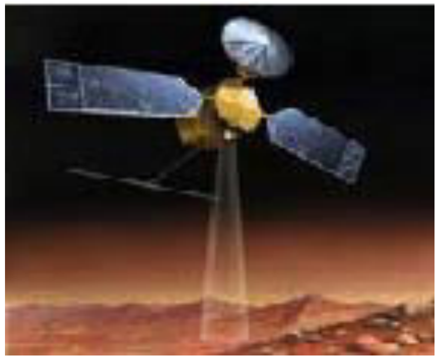
Intelligent Agents

Chapter 2

Ass. Prof. Dr. Zeynep ORMAN

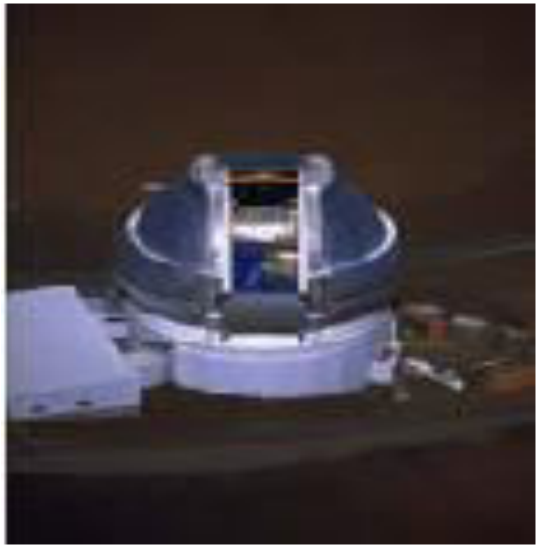
AI Applications

- Autonomous Planning & Scheduling:
  - Autonomous rovers.



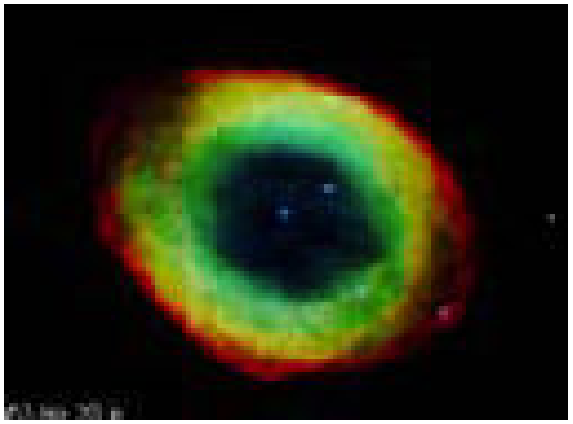
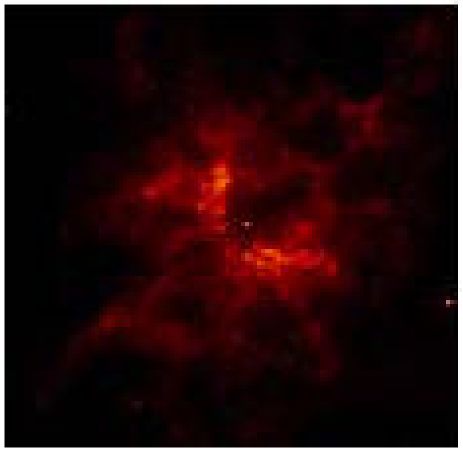
AI Applications

- Autonomous Planning & Scheduling:
  - Telescope scheduling



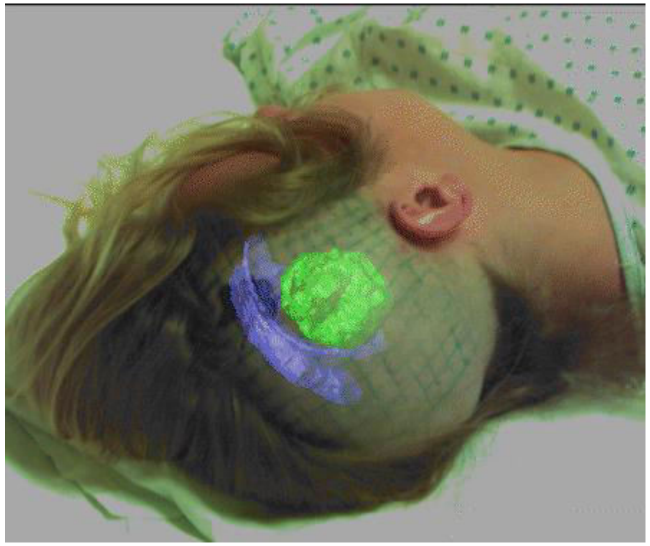
AI Applications

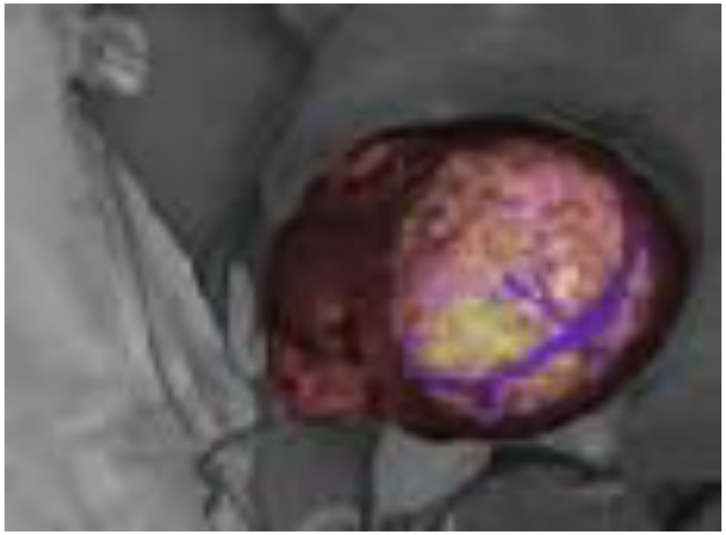
- Autonomous Planning & Scheduling:
  - Analysis of data:

AI Applications

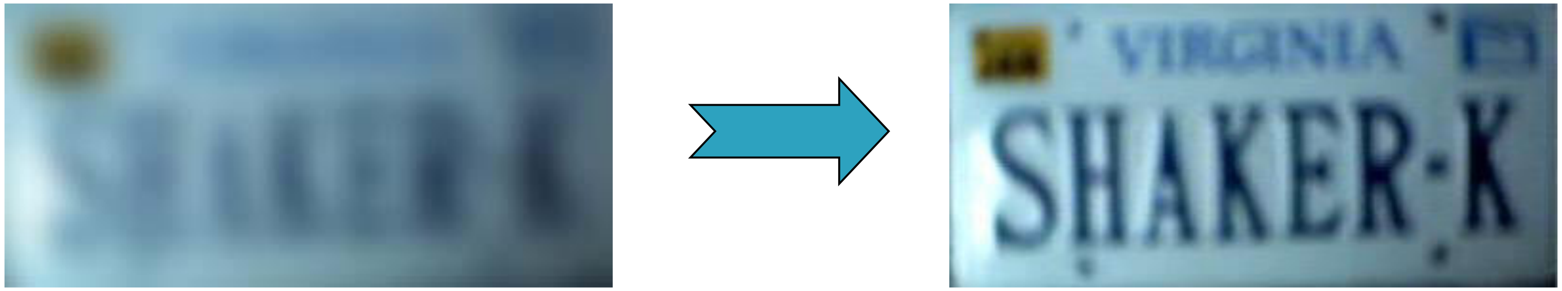
- **Medicine**:
  - Image guided surgery

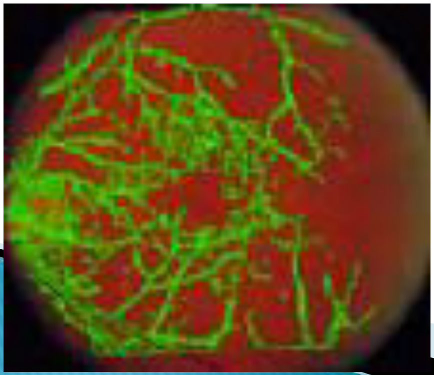


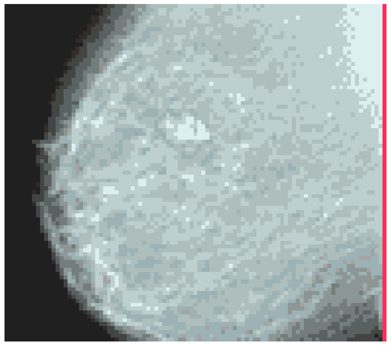


AI Applications

- **Medicine**:
  - Image analysis and enhancement







AI Applications

- **Transportation**:
  - **Autonomous vehicle control:**

AI Applications

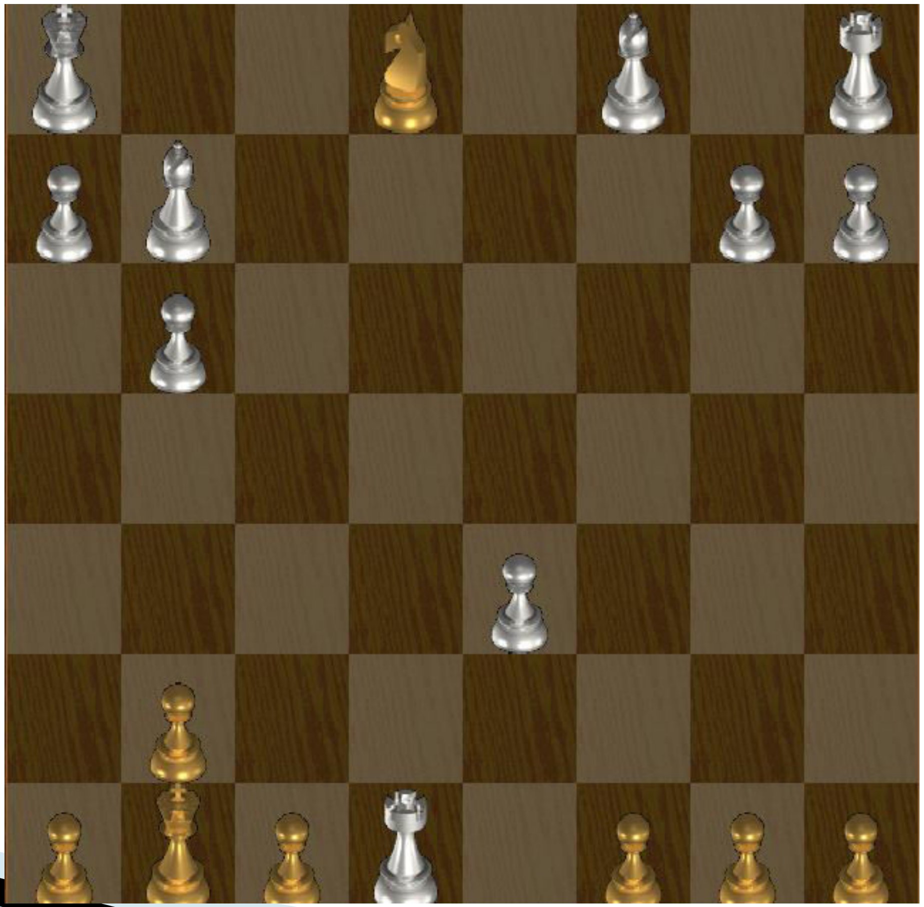
- **Transportation**:
  - **Pedestrian detection:**

AI Applications

Games:

AI Applications

- **Games**:



AI Applications

- **Robotic toys**:

AI Applications

Other application areas:

- **Bioinformatics:**
  - Gene expression data analysis
  - Prediction of protein structure
- **Text classification, document sortin**g:
  - Web pages, e-mails
  - Articles in the news
- **Video, image classification**
- **Music composition, picture drawing**
- **Natural Language Processing .**
- **Perception.**

Outline

- Define an agent
- Define an intelligent agent
- Define a rational agent
- Explain bounded rationality
- Discuss different types of environments
- Explain different agent architectures

Outline

On completion of this lesson, the student will be able to

- Understand what an agent is and how an agent interacts with the environment.
- Given a problem situation, the student should be able to,
  - identify the percepts available to the agent and
  - the actions that the agent can execute
- Understand the performance measures used to evaluate an agent.

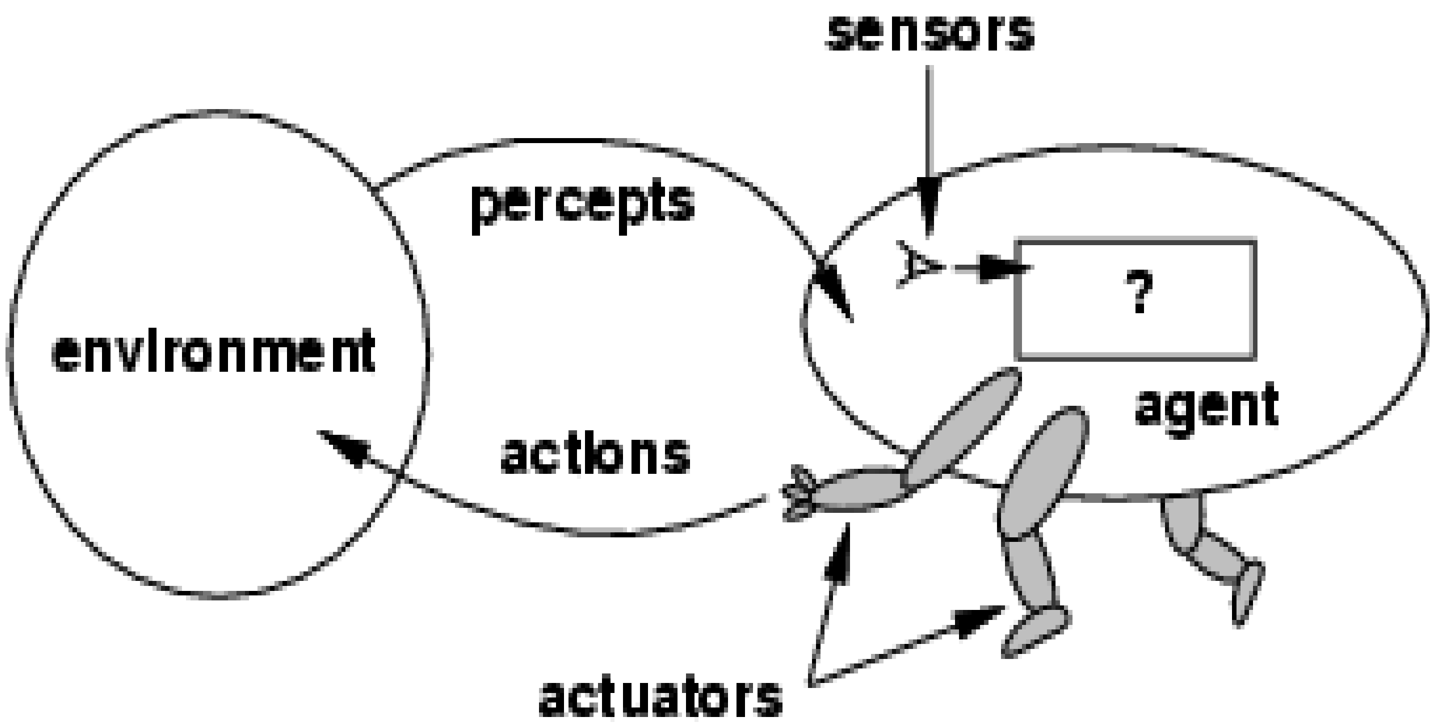
Outline

- Understand the definition of a rational agent
- Understand the concept of bounded rationality.
- Be familiar with
  - different agent architectures
  - reflex agents
  - state based agents
  - goal-based agents
  - utility based agents
  - learning agents

Outline

- Be able to analyze a problem situation and be able to
  - identify the characteristics of the environment.
  - recommend the architecture of the desired agent.

Agents and environments



Agents

- Agents operate in an environment
- Perceives its environment through sensors
- Acts upon its environment through actuators / effectors
- Have goals
  - objectives which the agent has to satisy

Sensors and Effectors

- An agent perceives its environment through sensors
  - The complete set of inputs at a given time is called a percept.
  - The current percept, or a sequence of percepts can influence the actions of an agent.
- It can change the environment through effectors / actuators
  - An operation involving an actuator is called an action.
  - Actions can be grouped into action sequences.

Agents

- Have sensors, actuators
- Have goals
- An agent program implements mapping from percept sequence to actions.
- Performance measure to evaluate agents.
- Autonomous agent decide autonomously which action to take in the current situation to maximize progress towards its goal.

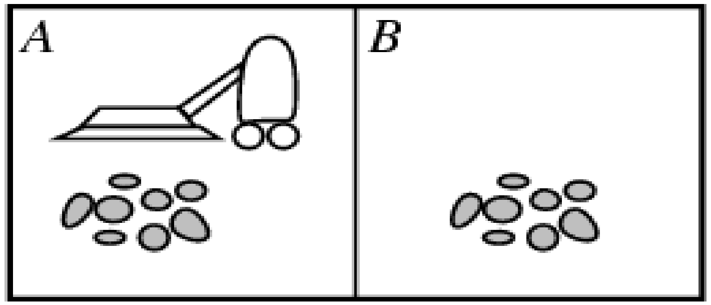
Agent function & program

- 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



- There are two locations: squares A and B.
- The vacuum agent perceives which square it is in and whether there is dirt in the square.
- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

A vacuum-cleaner agent

Partial tabulation of the agent function :

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

A simple agent program for this agent function :

```
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
```

Agent Function and Program

- **Agent Function:** Mathematically speaking, we say that an agent's behavior is described by the agent function that maps any given percept sequence to an action.

- **Agent Program:** The implementation of the agent function for an intelligent agent is called the agent program.

Performance

- Behavior and performance of IAs in terms of agent function
  - Perception history (sequence) to Action Mapping
  - Ideal Mapping: specifies which actions an agent should take at any point in time.

- Performance measure: subjective measure to characterize how successful an agent is (e.g. speed, power usage, accuracy, money, etc.)

Example of 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, skin, taste buds, etc. for sensors;
  - hands, legs, mouth, fingers, etc. for actuators
- **Robotic agent:**
  - Cameras, infrared range finders, bumpers, etc. for sensors;
  - various motors like wheels, lights, speakers, etc. for actuators

Example of Agents

- **Software agent** (Softbot)
  - Functions as sensors
  - Functions as actuators

Types of Agents: Robots

Xavier robot (CMU)
http://www.cs.cmu.edu/~Xavier/

COG (MIT)
http://www.ai.mit.edu/projects/humanoid-robotics-group/cog/

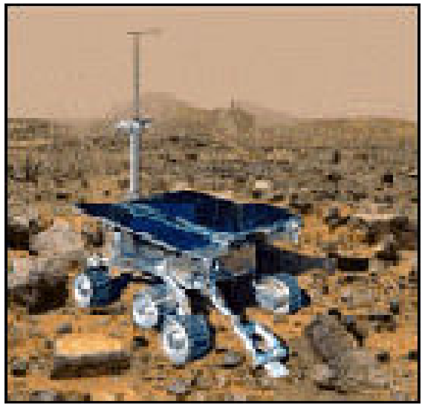
Types of Agents: Robots

AIBO
(Sony)

AIBO robotic pets are considered to be autonomous robots since they are able to learn and mature based on external stimuli from their owner, their environment and from other AIBOs

Types of Agents

- Softbots:
  - Askjeeves.com
- Expert Systems
  - whale watcher
  - cardiologist
- Autonomous spacecraft
- Intelligent buildings



Agents

- Fundamental activities of intelligence
  - Acting
  - Sensing
  - Understanding, reasoning, learning
- In order to act, you must sense. Blind actions are not characterization of intelligence.
- Robotics: sensing and acting; understanding not necessary
- Sensing needs understanding to be useful.

Intelligent Agents

- Intelligent Agent
  - must sense,
  - must act,
  - must be autonomous (to some extent),
  - must be rational

Rational agent

- AI is about building rational agents.
- An agent is something that perceives and acts
- A rational agent always does the right thing.
  - What are the functionalities (goals)?
  - What are the components?
  - How we build them?

Rationality

- Perfect rationality
  - Assumes that the rational agent knows all and will take the action that maximizes its utility.
  - Human beings do not satisfy this definition of rationality.
- Bounded rationality (Herbert Simon, 1972)
  - Because of the limitations of the human mind, humans must use approximate methods to handle many taks.

Rationality

- Rational action: the action that maximizes the expected value of the performance measure given the percept sequence to date.
  - Rational = Best ?
    - Yes, to the best of its knowledge
  - Rational = Optimal ?
    - Yes, to the best of its abilities
    - And its constraints

Rational agents

- **A rational agent** is one that does the right thing, based on what it can perceive and the actions it can perform.
- What is rational at any given time depends on four things:
  - The performance measure that defines the criterion of success.
  - The agent's prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent's percept sequence to date.

Rational agents

- **Performance measure:** An objective criteria 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

- 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.

Omniscience

- A rational agent is not omniscient.
  - It does not know the actual outcome of its actions
  - It may not know certain aspects of its environment.
- Rationality must take into account the limitations of the agent.
  - percept sequence, background knowledge, feasible actions
  - deal with the expected outcome of actions.

Bounded rationality

- Bounded rationality is that
  - Property of an agent that behaves in a manner that is nearly optimal with respect to its goals as its resources will allow.

PEAS

- 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?
  - Environment?
  - Actuators?
  - Sensors?

PEAS for an Automated Taxi

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

PEAS for Internet Shopping Agent

- Performance measure: price, quality, appropriateness, efficiency
- Environment: WWW sites, vendors, shippers
- Actuators: display to user, follow URL, fill in form
- Sensors: HTML pages (text, graphics, scripts)

Environment types

- Fully observable vs. partially observable
  - If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable.
  - An environment can be partially observable because of noisy and inaccurate sensors.
    - e.g. a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares.
    - an automated taxi cannot see what other drivers are thinking.

Environment types

- WHAT ABOUT PLAYING
  - Chess?
  - Poker?

Environment types

- Deterministic vs. stochastic
  - If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise it is stochastic.
    - e.g. taxi driving is clearly stochastic in the sense of one can never predict the behavior of traffic exactly.
    - the vacuum world is deterministic.

Environment types

- Episodic vs. sequential
  - In an episodic task environment, the agent’s experience is divided into atomic “episodes.”
  - Each episode consists of the agent perceiving and then performing a single action.
    - the next episode does not depend on the actions taken in previous episodes.
    - the choice of action in each episode depends only on the episode itself.

Environment types

- Episodic vs. sequential
  - In sequential environments, the current decision may affect all future decisions.
    - chess and taxi driving are sequential
  - Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

Environment types

- Static vs dynamic
  - If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise it is static.
    - static environments are easy to deal with.
  - If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is semidynamic.
    - taxi driving is dynamic
    - chess when played with a clock is semidynamic.
    - crossword puzzles are static.

Environment types

- Discrete vs. continuous
  - A limited number of distinct, clearly defined percepts and actions.
    - e.g. a chess game is a discrete-state environment, it has a finite number of distinct states.
    - taxi driving is a continuous-state problem.
      - the speed and the location of the taxi, and taxi driving actions are continuous

Environment types

- Single agent vs. multiagent
  - An agent operating by itself in an environment.
    - e.g. an agent solving a crossword puzzle by itself is clearly in a single-agent environment.
    - an agent playing chess is in a two-agent environment.

Environment types

- Multiagent environments :
  - competitive multiagent environments
    - e.g. in chess, the agent tries to maximize its performance mesaure while minimizing the other agent's performance mesaure.
  - cooperative multiagent environments
    - e.g. in taxi-driving environment, to avoid collisions, all agents try to maximize the performance measures.
    - but only one car can occupy a parking space.
    - so taxi-diriving is *partially cooperative* and *partially competitive*.

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?				
Deterministic?				
Episodic?				
Static?				
Discrete?				
Single-Agent?				

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?	Yes	Yes	No	No
Deterministic?				
Episodic?				
Static?				
Discrete?				
Single-Agent?				

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?	Yes	Yes	No	No
Deterministic?	Yes	No	Partly	No
Episodic?				
Static?				
Discrete?				
Single-Agent?				

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?	Yes	Yes	No	No
Deterministic?	Yes	No	Partly	No
Episodic?	No	No	No	No
Static?				
Discrete?				
Single-Agent				

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?	Yes	Yes	No	No
Deterministic?	Yes	No	Partly	No
Episodic?	No	No	No	No
Static?	Yes	Semi	Semi	No
Discrete?				
Single-Agent				

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?	Yes	Yes	No	No
Deterministic?	Yes	No	Partly	No
Episodic?	No	No	No	No
Static?	Yes	Semi	Semi	No
Discrete?	Yes	Yes	Yes	No
Single-Agent				

Environment types

	Solitaire	Backgommon	Internet Shopping	Taxi
Observable?	Yes	Yes	No	No
Deterministic?	Yes	No	Partly	No
Episodic?	No	No	No	No
Static?	Yes	Semi	Semi	No
Discrete?	Yes	Yes	Yes	No
Single-Agent	Yes	No	Yes(except auctions)	No

Agent functions and programs

- The agent programs all have the same skeleton :
  - they take the current percept as input from the sensors and
  - return an action to the actuators.
  - if the agent’s actions depend on the entire percept sequence, the agent will have to remember the percepts.

Agent functions and programs

- NOTE : the agent function takes the entire percept history.
- We will describe agent programs with a simple pseudocode language.

Agent functions and programs

• An agent program that keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do.

```
function TABLE-DRIVEN-AGENT( percept) returns an
  action
static:
  percepts, a sequence, initially empty
  table, a table of actions, indexed by percept
  sequences, initially fully specified
append percept to the end of percepts
action ←LOOKUP( percepts, table)
return action
```

Table-driven agent

- This program is invoked for each new percept and returns an action each time.
- To build a rational agent in this way, we must construct a table that contains the actions for percept sequences.
  - P : set of possible percepts
  - T : lifetime of the agent (total number of percepts)
  - The lookup table will contain : $\sum_{t=1}^{T} |P|^t$ entiries.

Table-driven agent

- Problems:
  - Huge table
  - Take a long time to build the table
  - All work done by a designer
  - No physical agent will have the space to store the table.
  - No autonomy – no agent could ever learn all the right table entiries from its experience.
    - All actions are predetermined
  - Even with learning, need a long time to learn the table entries

Table-driven agent

- Mapping is implicitly defined by a program
  - Rule based
  - Neural networks
  - Algorithm

Agent types

- Four basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
- All these can be turned into learning agents.

Simple reflex agents

- The simplest kind of agent is the **simple reflex agent.**
- These agents select actions based only on the *current* percept, ignoring the rest of the percept history.
  - e.g. the vacuum agent is a simple reflex agent, because its decision is based only on the current location and on whether that contains dirt.

Simple reflex agents

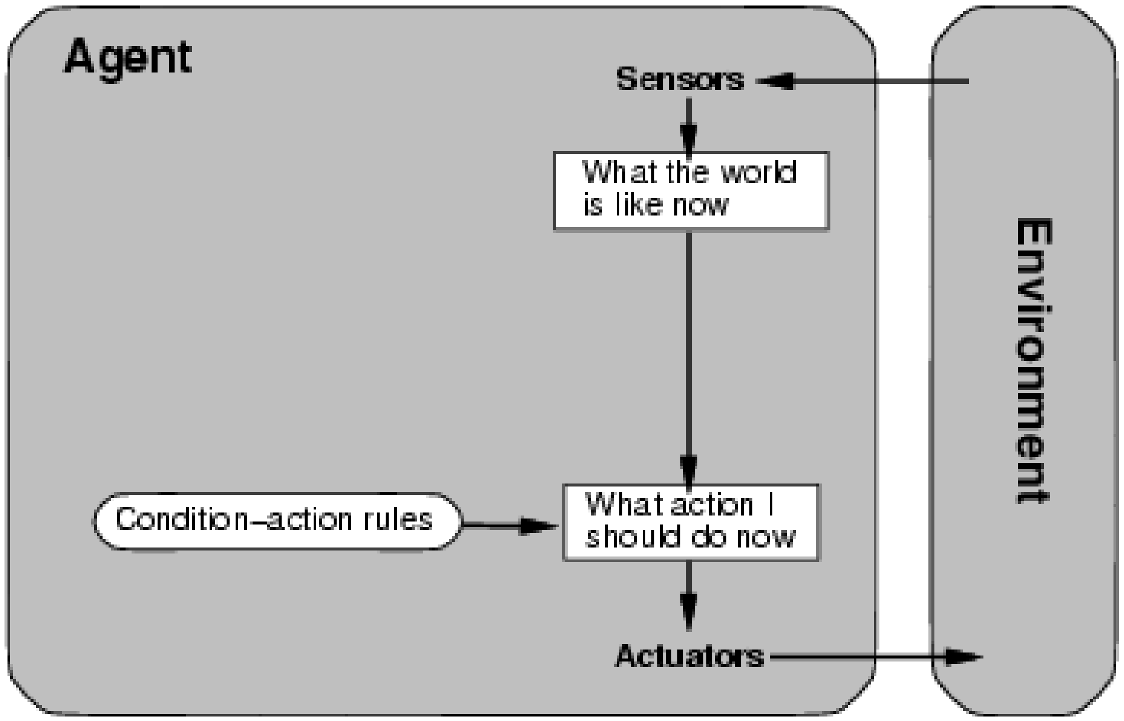
e.g. driver of the automated taxi :

- if the car in front brakes and its brake lights come on, then the driver should notice it and begin braking.

- We call such a connection *condition–action rule :*

if *car–in–front–is–braking* **then** *initiate–breaking*

Simple reflex agents



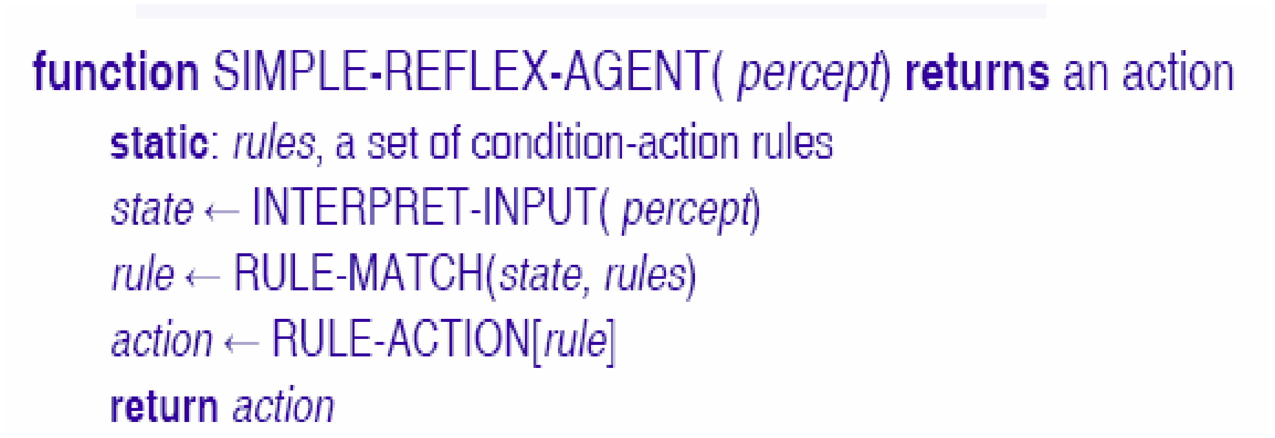
• It shows how the condition-action rules allow the agent to make connection from percept to action.

• rectangles -> current internal state of the agent's decision process

• ovals -> background information used in the process.

Simple reflex agents

The agent program :



```
function SIMPLE-REFLEX-AGENT( percept) returns an action
    static: rules, a set of condition-action rules
    state ← INTERPRET-INPUT( percept)
    rule ← RULE-MATCH(state, rules)
    action ← RULE-ACTION[rule]
    return action
```

- **INTERPRET-INPUT** function generates a description of the current state from the percept.
- **RULE-MATCH** function returns the first rule that matches the given state description.
- **RULE-ACTION** function generates an action from the rule.

Simple reflex agents

- Problems
  - Simple reflex agents have the very good property of being simple but they have limited intelligence.
  - This agents will work only if the correct decision can be made, that is if the environment is fully observable.
  - Too big to generate and to store. (Chess has about 10120 states, for example)
  - Not adaptive to changes in the environment; requires entire table to be updated if changes occur.

Simple reflex agents

- Looping: Infinite loops are often aviodable.
  - e.g. vacuum agent without location sensor and has only a dirt sensor.
  - percepts : [*Dirty*] and [*Clean*]
  - if [*Dirty*], it can *Suck*
  - if [Clean], moving *Left* if it starts in square A fails
    moving *Right* if it starts in square B fails

NOTE : Escape from infinite loops is possible if the agent can **randomize** its actions.

e.g. if the vacuum agent perceives [*Clean*], it might flip a coin to choose between *Left* and *Right*.
So the agent will reach the other square in an average of 2 steps.

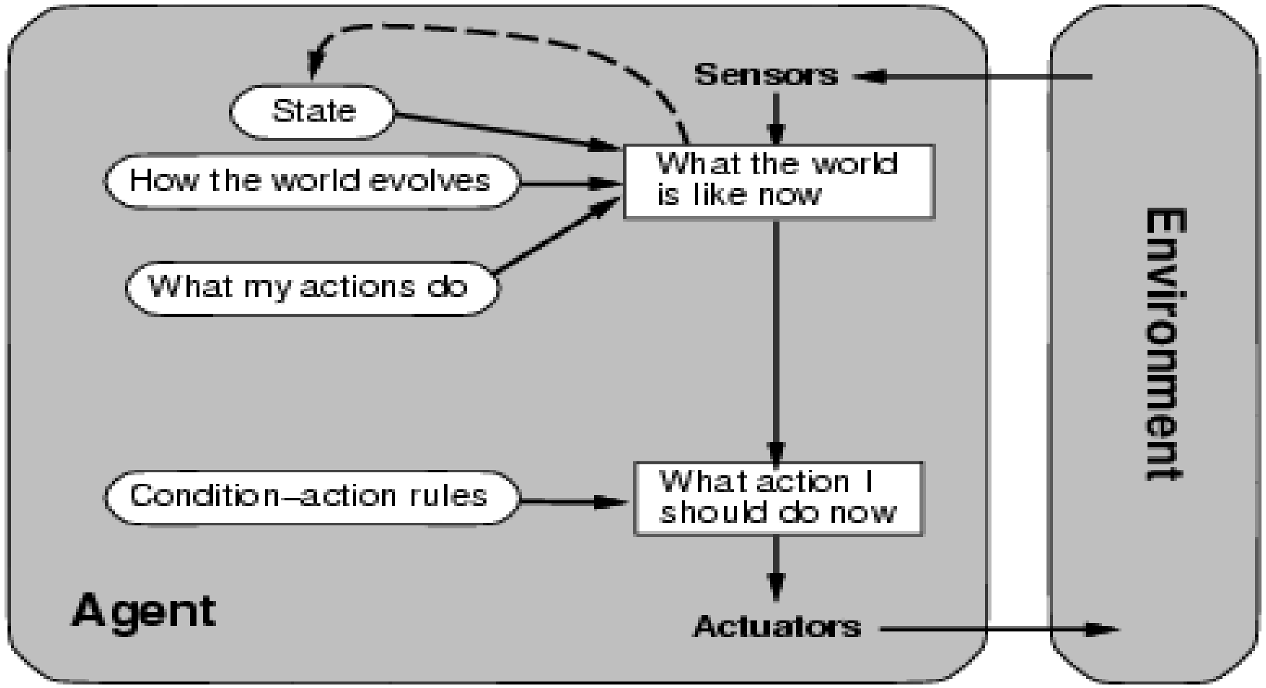
Reflex Agent with State

- The agent should maintain some sort of internal state that depends on the percept history to keep track of the part of the world it cannot see.
- The knowledge about “how the world works” is called a model of the world.
- An agent that uses such a model is called a model-based agent.
- Encode "internal state" of the world to remember the past as contained in earlier percepts.

Reflex Agent with State

- Requires ability to represent change in the world and update the internal state information as time goes by.
- Requires two kinds of knowledge in the agent program :
  - information about how world involves independently of the agent.(e.g. overtaking car will be closer behind than it was a moment ago.)
  - information about how the agent’s own actions affect the world. (e.g. when the agent turns the steering wheel clockwise, the car turns to the right.)

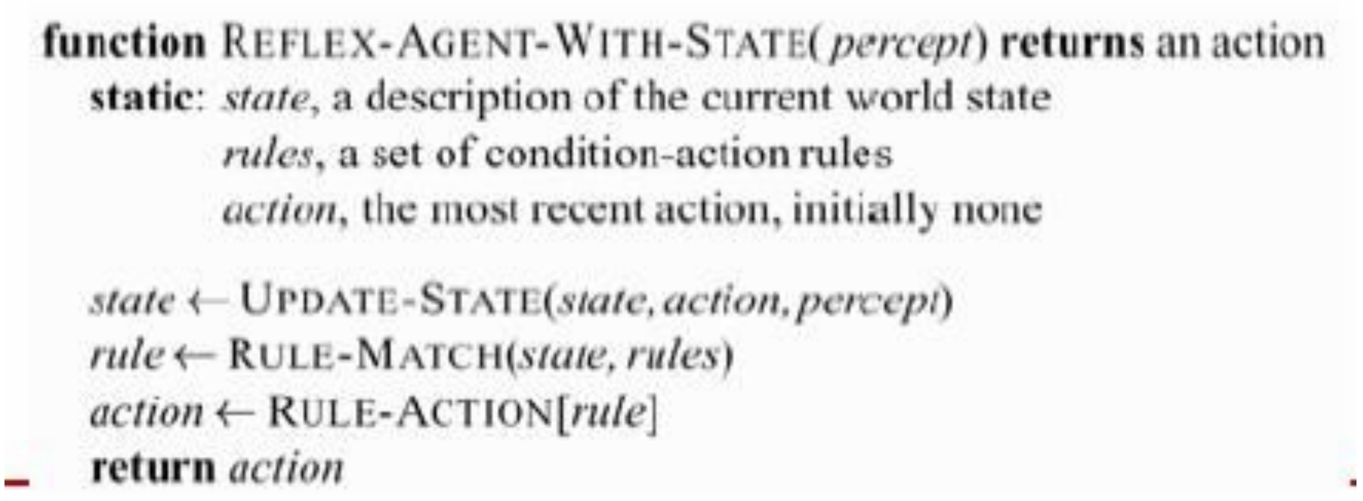
Reflex Agent with State



It shows how the current percept is combined with the old internal state to generate the updated description of the current state.

Reflex Agent with State

The agent program :



```
function REFLEX-AGENT-WITH-STATE(percept) returns an action
    static: state, a description of the current world state
            rules, a set of condition-action rules
            action, the most recent action, initially none

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

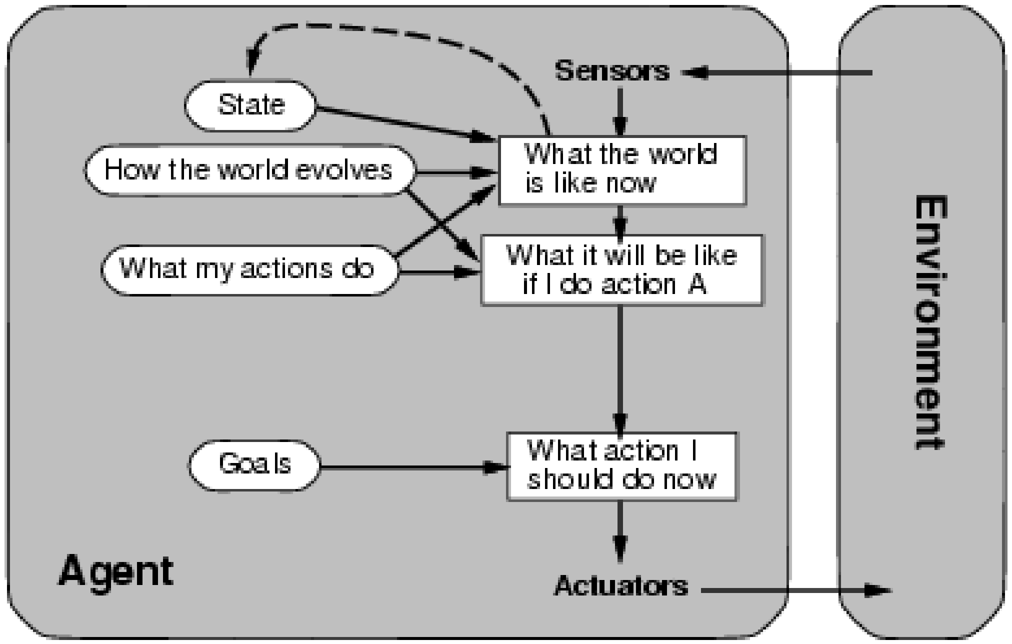
UPDATE-STATE function is responsible for creating the new internal state description.

Goal-based agents

- Knowing about the current state of the environment is not only enough to decide what to do.
  - e.g. at a road junction, the taxi can turn left, right or go straight.
  - the correct decision depends on where the taxi is trying to get to.
- Current state + goal information.

  (being at the passenger's destination)

Goal-based agents



Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will lead to the achievement of its goal.

Goal-based agents

- Choose actions so as to achieve a (given or computed) goal= a description of a desirable situation
- Keeping track of the current state is often not enough--- need to add goals to decide which situations are good
- Deliberative instead of reactive

Goal-based agents

- May have to consider long sequences of possible actions before deciding if goal is achieved--- involves consideration of the future, "what will happen if I do...?“
- **Search** and **planning** are the subfields of AI for finding action sequences that achieve the agent’s goal.

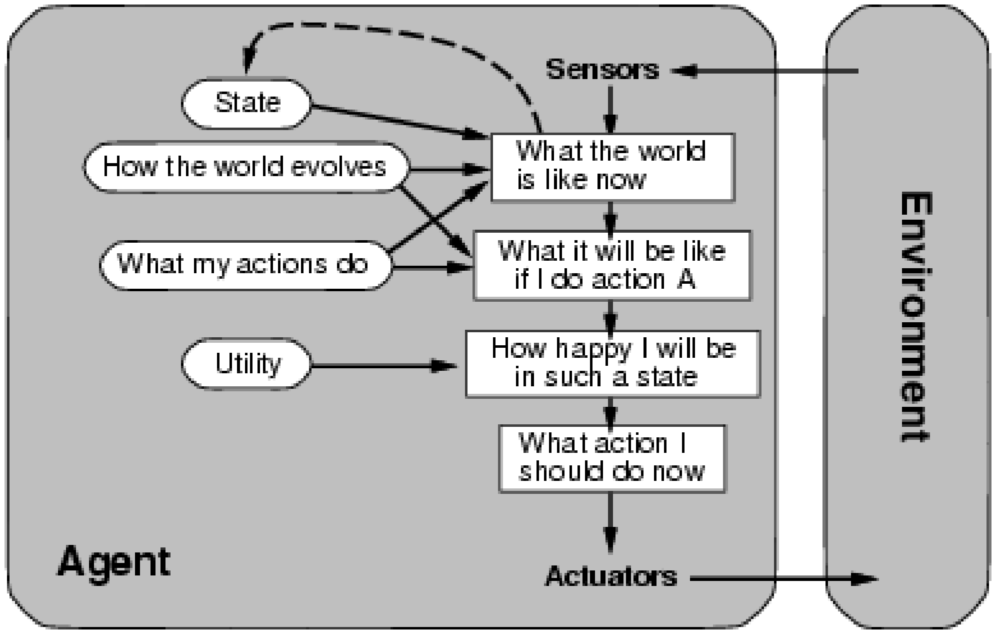
Utility-based agents

- When there are multiple possible alternatives, how to decide which one is best?
  - e.g. there are many action sequences that will get the taxi to its destination.
  - but some are quicker, safer, more reliable or cheaper than others.
- Goals provides a crude distinction between a happy and unhappy state,
- but often need a more general performance measure that describes "degree of happiness"

Utility-based agents

- Utility function U: State --> Reals
    - maps a state (or sequence of states) onto a real number which shows the degree of happiness.
    - indicates a measure of success or happiness when at a given state.
    - when there are conflicting goals (e.g. speed and safety), U selects the appropriate one.

Utility-based agents


Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

It uses a model of the world with a utility function. Then it chooses the action that leads to the best expected utility.

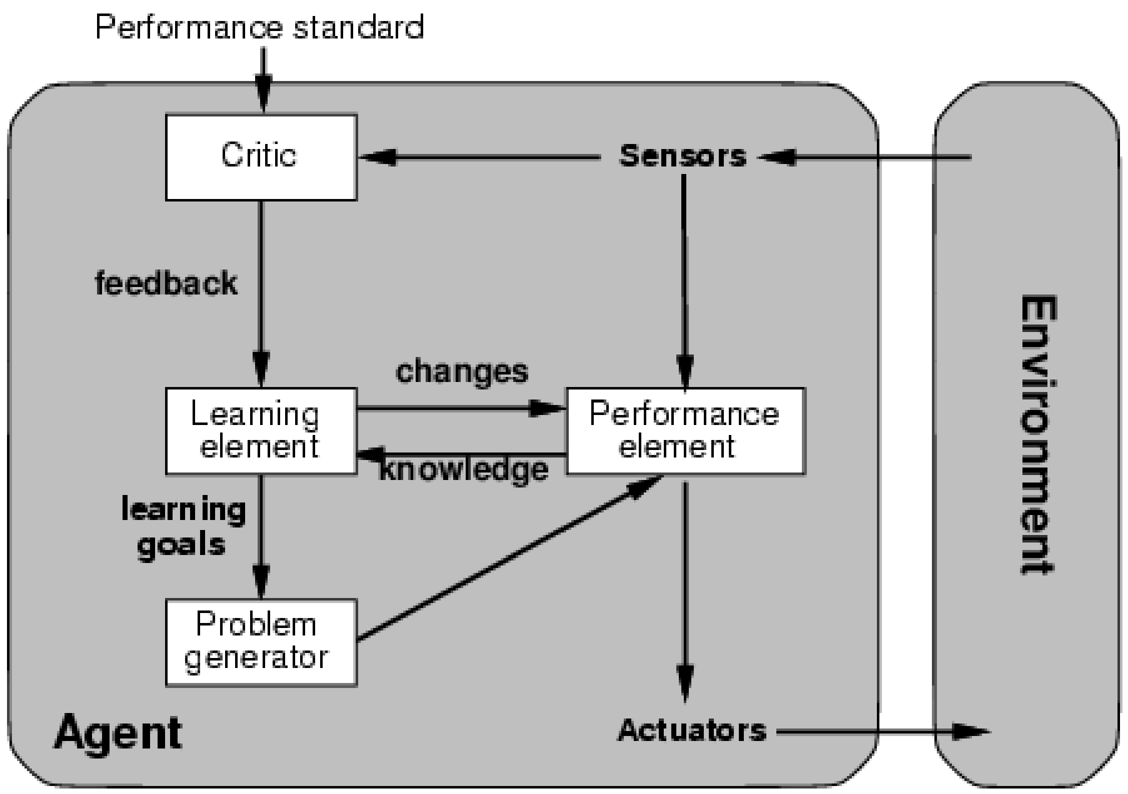
Components of a Learning Agent

- **learning element**, which is responsible for making improvements,
- **performance element**, which is responsible for selecting external actions. The performance element is the entire agent: it takes in percepts and decides on actions.
- **critic**
  - gives feedback to the learning element about how the agent is doing with respect to a fixed performance standard and
  - determines how the performance element should be modified to do better in the future..
- **problem generator** is responsible for suggesting actions that will lead to new and informative experiences.

Problem Generator

- E.g. if learning agents were only use performance element, it would keep doing the actions that are best
- But with the problem generator, it would do perhaps some suboptimal actions in short run and it might discover much better actions for the long run.
- Problem generator's job → to suggest these actions.

Learning agents



SUMMARY

- **Agents** interact with environments through actuators and sensors.
- The **agent function** describes what the agent does in all circumstances.
- The **performance measure** evaluates the behavior of the agent in an environment.
- A **rational agent** maximizes expected performance.
- **Agent programs** implement (some) agent functions
- **PEAS** is used for Performance measure, Environment, Actuators, Sensors.

SUMMARY

- Environments are categorized along several dimensions:

  observable? **deterministic**? **episodic**? **static**? **discrete**? **single-agent**?

- Several basic agent architectures exist:

reex, reex with state, goal-based, utility-based

- All agents can improve their performance through **learning**.

Example

- Find out about the Mars rover
  - What are the percepts for this agent?
  - Characterize the operating environment?
  - What are the actions the agent can take?
  - How can one evaluate the performance of the agent?
  - What sort of agent architecture do you think is most suitable for this agent, and why?