
Intelligent Systems
SCE408
Intelligent Agents
[Ch-2]

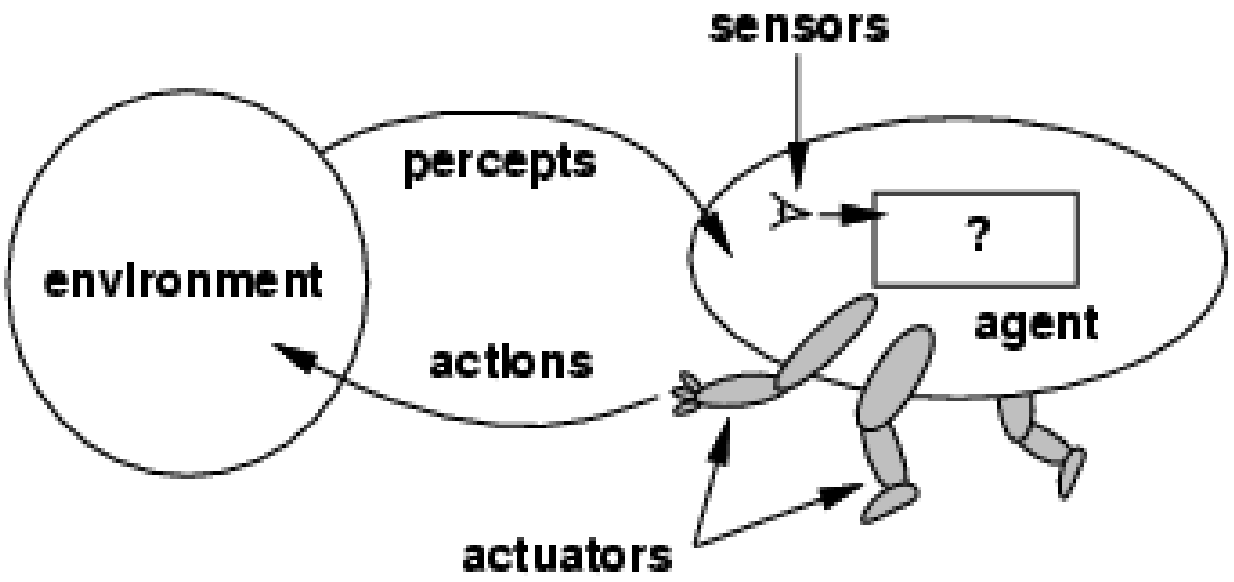
Review

- What is AI?
- What is Turing Test?
- List 6 AI Applications

Contents

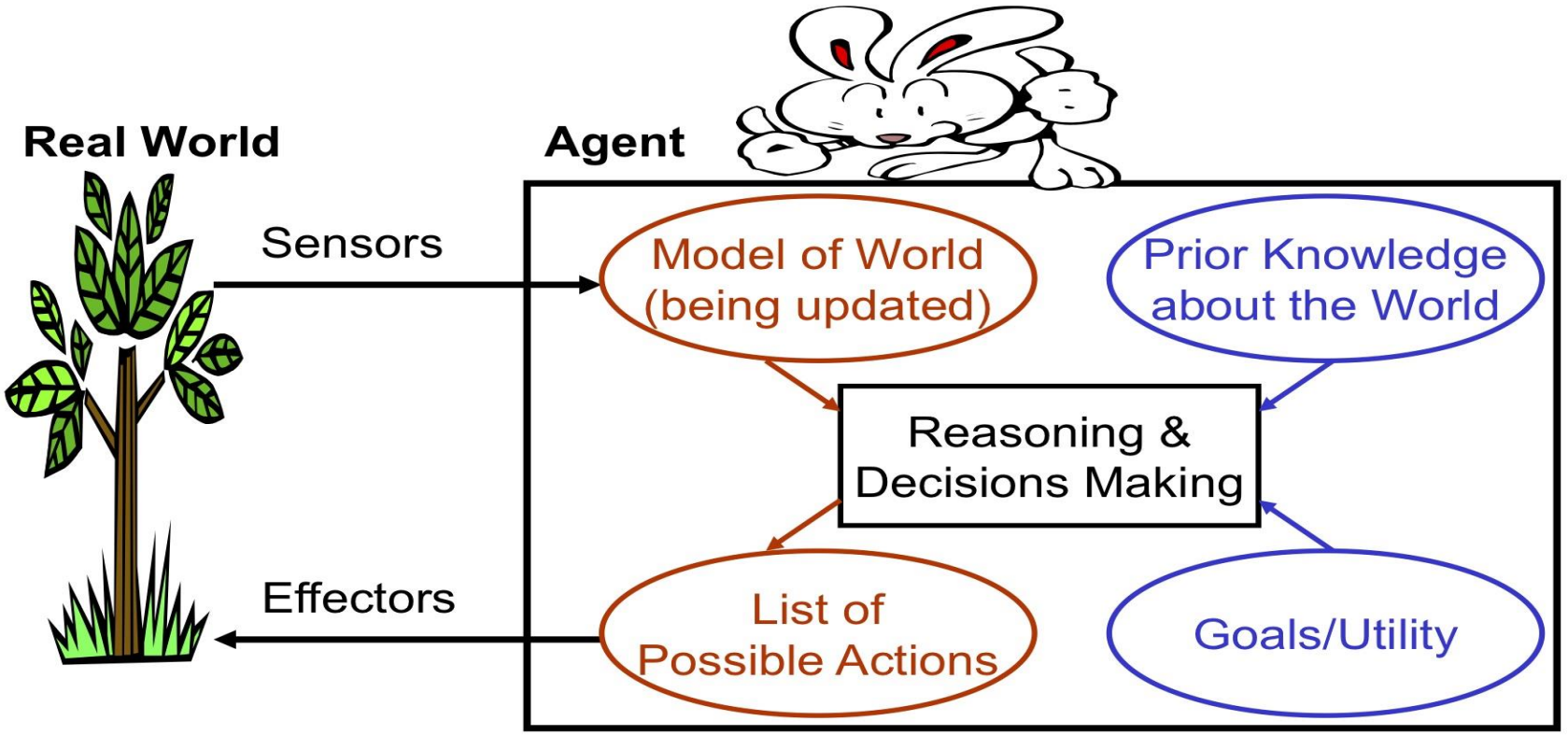
Intelligent Agents

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

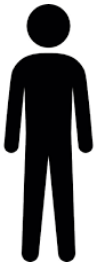


Intelligent Agents

AI Architecture



Intelligent Agents, examples



Human Agent	
Sensors	Eyes, ears, nose, skin,..
Actuators	Hands, legs, mouth,..

Robotic Agent	
Sensors	Cameras, infrared ,GPS, …
Actuators	Various motors, wheels, Arms,....

A software Agent	
Sensors	Reading data: Keystrokes, file contents, received network packages,……
Actuators	Displaying data on the screen, writing files, sending network packets,…

Intelligent Agents structure

➢**An agent's behavior** is described by the **agent function** which maps from percept histories to actions:

$$\mathbf{f}: P^* \rightarrow A$$

➢Agent function is implemented by an agent program which runs on the physical architecture to produce **f**

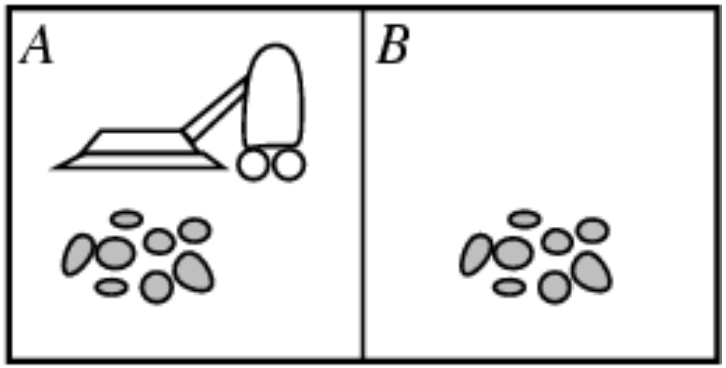
agent = architecture + **Program**

Computing device Running Agent Program, with sensors & actuators

Implements Agent Function, performs mapping of percepts to actions

Intelligent Agents, example

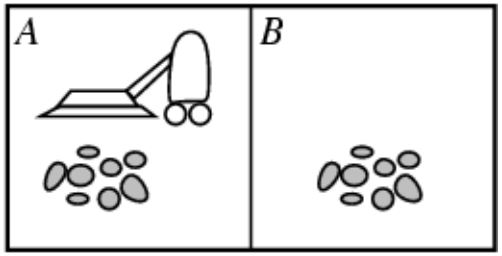
Vacuum-cleaner agent



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

Intelligent Agents

Agent function as look up Table:



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*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

Intelligent Agents

Agent function as look up Table:

- An agent actions is completely specified by the lookup table

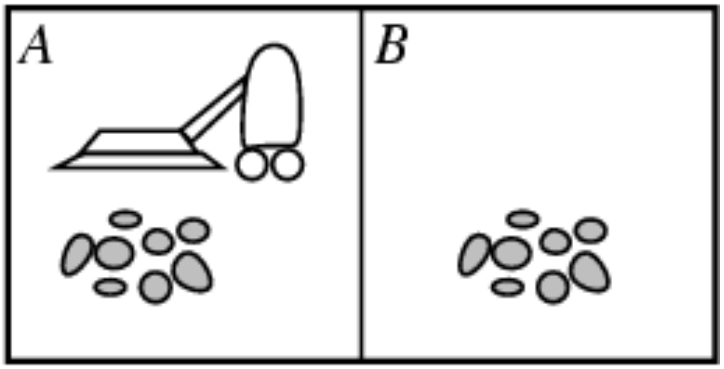
```
function Table-Driven-Agent(percept) returns action
  static: percepts, a sequence, initially empty
          table, a table indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LookUp(percepts, table)
return action
```

- Drawbacks:
  - Huge table
  - Take a long time to build the table
  - No autonomy

Intelligent Agents

Rational Agent

- For each possible **percept** sequence, Ideal rational agent should do whatever **action** expected to **maximize performance** measure, on the basis of built-in **knowledge** agent has



Agent design

Agent Design (PEAS)

- **P**erformance: How agent be assessed?
- **E**nvironment: What elements exists around agent?
- **A**ctuators: How agent change the environment?
- **S**ensors: How agent sense the environment?

Agent design

Agent Design (PEAS)

Automated taxi driver

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

Agent design

Agent Design (PEAS)

Part-picking robot

- Performance: Percentage of parts in correct bins, speed
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

Agent design, Quiz.....

Agent Design (PEAS)

Medical diagnosis system

- Performance: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff,.....
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, patient's answers)

Environment Properties

Environment Properties (ODESDA)

- Observable (or, partially observable)

  An agent's sensors give it access to the complete state of the environment at each point in time

- Deterministic (or, stochastic)

  The next state of the environment is completely determined by the current state and the action executed by the agent

- Episodic (or, sequential)

  The agent's experience is divided into episodes, in each episode the agent receives a percept and then performs a single action & the next episode does not depend on the actions taken in previous episodes

Environment Properties

Environment Properties (ODESDA)

- Static (or, Dynamic)
  The environment is unchanged while an agent is deliberating
- Discrete (or, Continuous)
  A limited number of distinct, clearly defined percepts and actions.
- Agent (single/ multi) (cooperative /competitive)
  Number of agent in the environment

Environment Properties

Environment Properties (ODESDA)

	Chess with a clock	Chess without a clock	Taxi driving
Observable	Yes	Yes	No
Deterministic	Yes	Yes	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: partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Environment Properties

Complete the following

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle Chess with a clock						
Poker						
Taxi driving Medical diagnosis						
Image analysis Part-picking robot						
Interactive English tutor						

Environment Properties

Score Yourself

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Agents Types

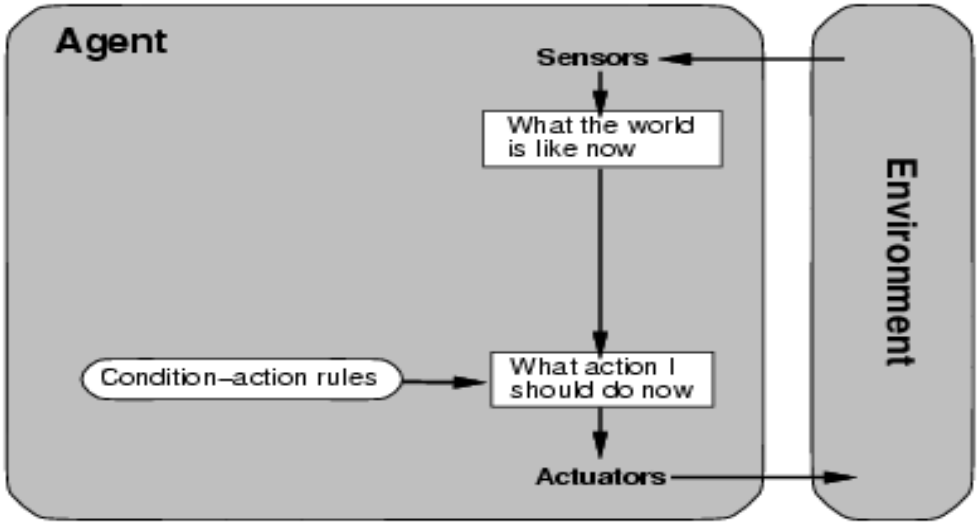
Agent Programs Basic Types:

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

Agents Types

❑ Simple reflex agents

❖Choose actions only based on the current percept
❖Ignore the precept history (no memory)
❖Use condition-action rule–acts according to a rule whose condition matches the current state (percept).



rectangles ← the current internal state; Ovals ← background information

Example:
if ***car-in-front-brakes***
then ***initiate braking***

Very simple !

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*

Agents Types

❑ Model-based reflex agents

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

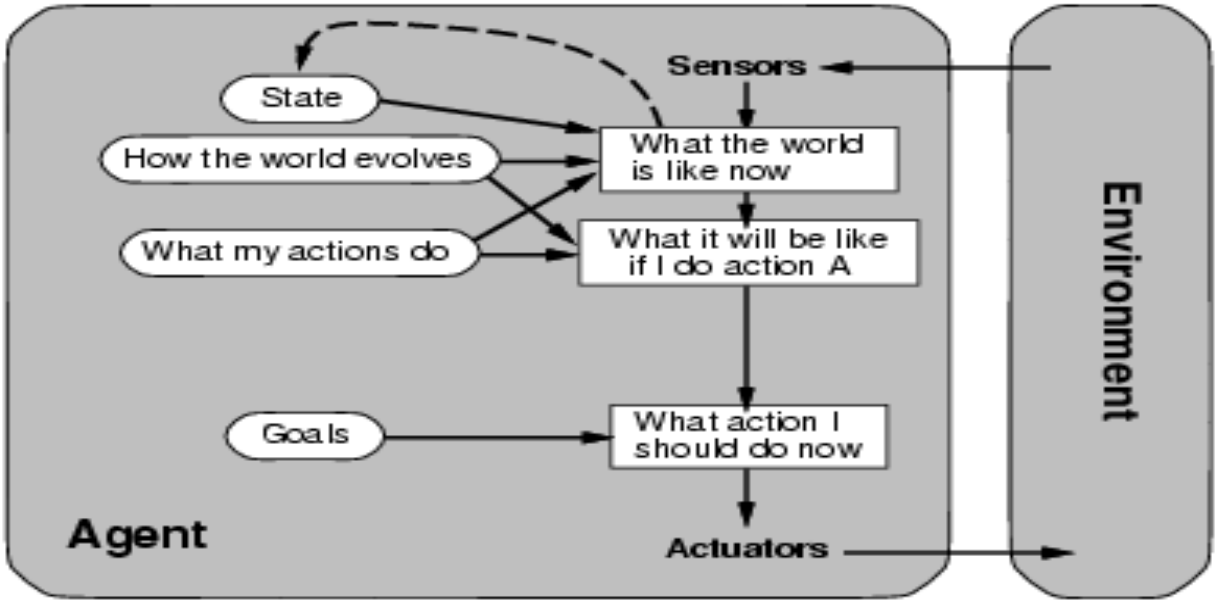
  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

- ❖Action depend on history or unperceived aspects of the world
- ❖Need to maintain internal world model (state)

Without clear goal it is unclear to know what to do!

Agents Types

❑ Goal-based agents



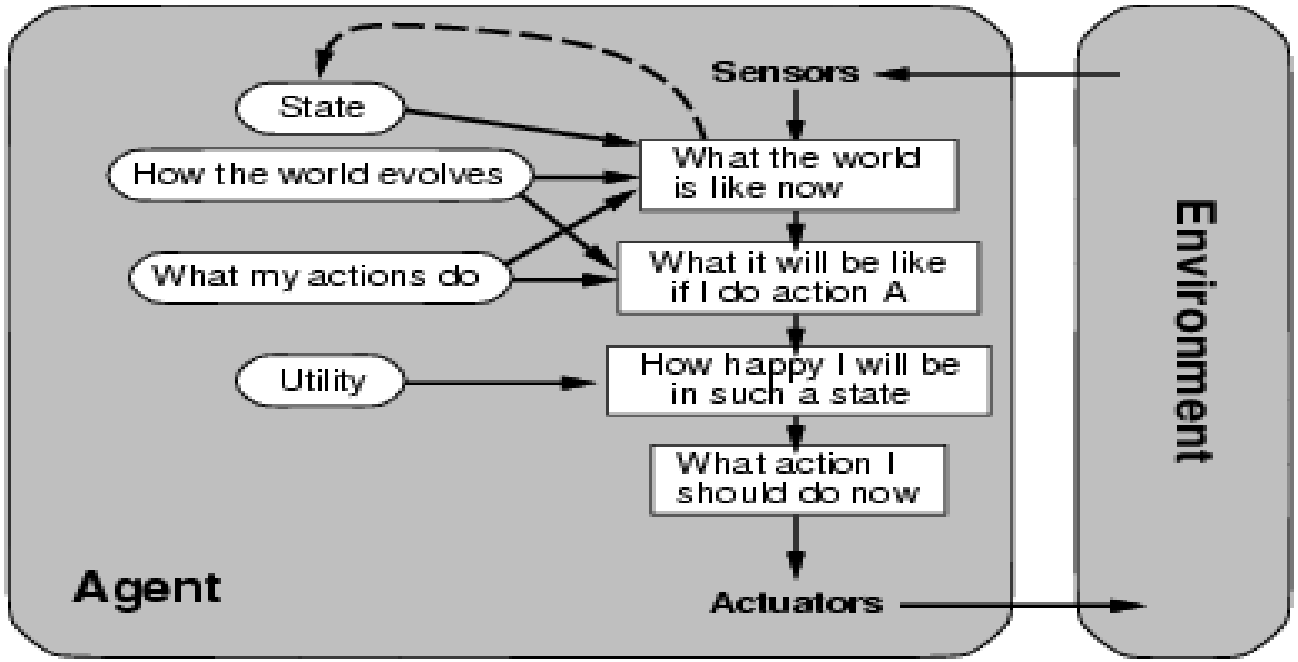
❖Agents of this kind take future events into consideration

❖Agent has some goal information, choose actions according to goal

Some solutions to goal states are better than others!

Agents Types

- Utility-based agents



- Try to Maximize agent expected happiness

Models To Be Studied

- **State-based Models (Search, Planning)**
  - Solutions are defined as a *sequence of steps*
  - Model task as a graph of states and solution as a path
  - A state captures all the relevant information about the past in order to act (optimally) in the future
  - Apps: navigation, games
  - State-space graphs
- **Parametric, Reflex Models (Machine Learning)**
  - Given a set of (input, output) pairs of training data, learn a set of parameters that will map input to output for future data
  - Apps: classification, regression
  - Decision trees, neural networks, SVMs, k-NN

Models to be Studied

- **Variable-based Models (Uncertainty)**
  - Solution is an assignment of values for a set of variables
  - Apps: Sudoku, speech recognition, face recognition
  - Constraint satisfaction, Bayesian networks, Hidden Markov Models, CNNs
- **Logic-based Models (Logic)**
  - Symbolic representation of classes of objects
  - Deductive reasoning
  - Apps: Question answering systems, natural language understanding
  - Propositional logic, First-order logic


Thank You !