# Lec2: Intelligent Agents

# Recap

- What we have learnt so far?

# Roadmap of the course

- Myuni schedule

- About first tutorial

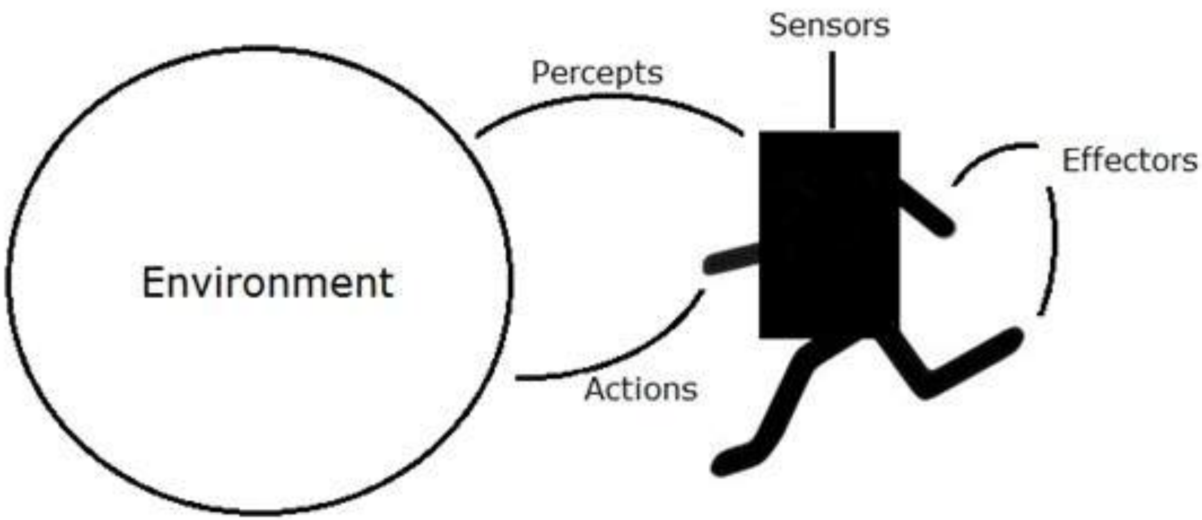
# Outline of this lecture

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

# What is an Agent?

- Agent def:
  - anything that can perceive its **environment** through **sensors** and acts upon that environment through **actuator**/**effectors**.

# Agent




Agents interact with environments through sensors and actuators/ effectors.

# Agent examples

- **human agent**
  - eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- **robotic agent**
  - cameras and infrared range finders for sensors and various motors for actuators
- **software agent**
  - keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.

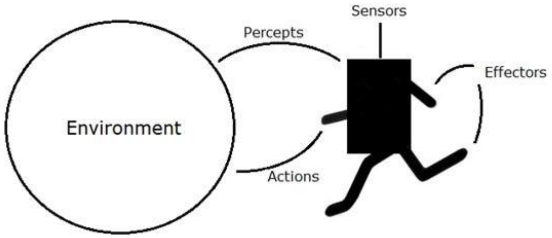
# Agent Terminology

- **Performance Measure of Agent**
  - It is the criteria, which determines how successful an agent is.
- **Behavior of Agent**
  - It is the action that agent performs after any given sequence of percepts.
- **Percept**
  - It is agent's perceptual inputs at a given instance.
- **Percept Sequence**
  - It is the history of all that an agent has perceived till date.
- **Agent Function**
  - It is a map from the precept sequence to an action (math concept). Its implementation is called **Agent program**.

# Agents and environments

- The agent function maps from percept histories to actions:

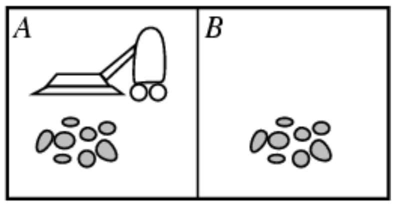
  $[f: \mathcal{P}^\star \rightarrow \mathcal{A}]$




- The agent program runs on the physical architecture to produce $f$

- Agent = architecture + program
  - Architecture needs to be consistent with the program (vice versa)
    - program is going to recommend actions like “Walk”, the architecture had better have “Legs”.
    - A program should not recommend an ordinary car to “Fly”

# Vacuum-cleaner world




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

# Agent program

**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*

**Figure 2.8** The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

- *What is the right way to fill out the table?* What makes an agent good or bad, intelligent or stupid?

# 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/prior knowledge the agent has.*

# Rationality depends on

- **performance measure** that defines the criterion of success.
- **prior knowledge** of the environment.
- **actions** that the agent can perform.
- **percept sequence** to date.

# Rationality != perfection

- Rationality != perfection
  - Rationality maximizes *expected* performance, while perfection maximizes *actual* performance.
  - Example
- Rationality != omniscience
  - Percepts may not supply all relevant information
  - E.g., in card game, don't know cards of others.

# PEAS (to define the problem)

- PEAS: **P**erformance measure, **E**nvironment, **A**ctuators, **S**ensors
- 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

# Environment types

- 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)

- 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)

- 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):

- 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):

- 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 |

# Discrete (vs. continuous)

- 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):

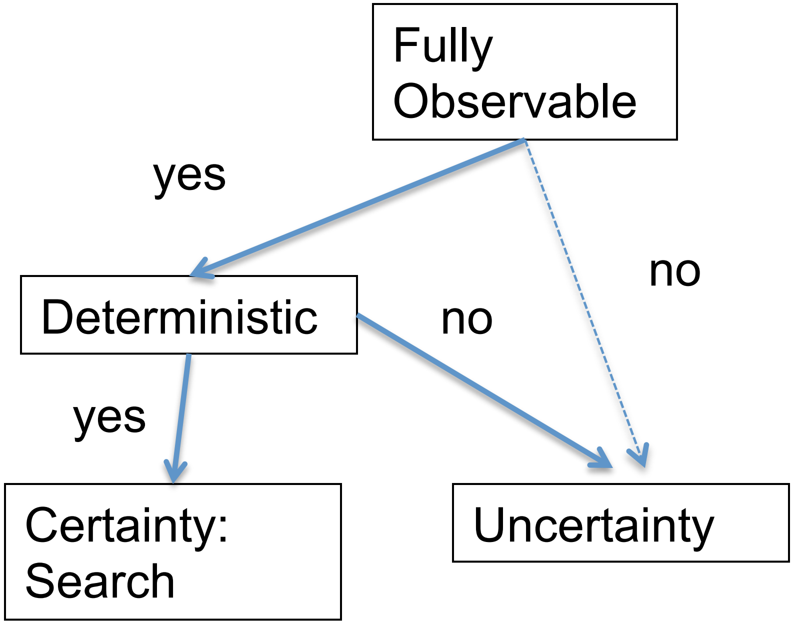
- An agent operating by itself in an environment or there are many agents working together

| Cross Word | Poker | Backgammon | Taxi driver | Part picking robot | Image analysis |
|---|---|---|---|---|---|
| Single | Multi | Multi | Multi | Single | 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



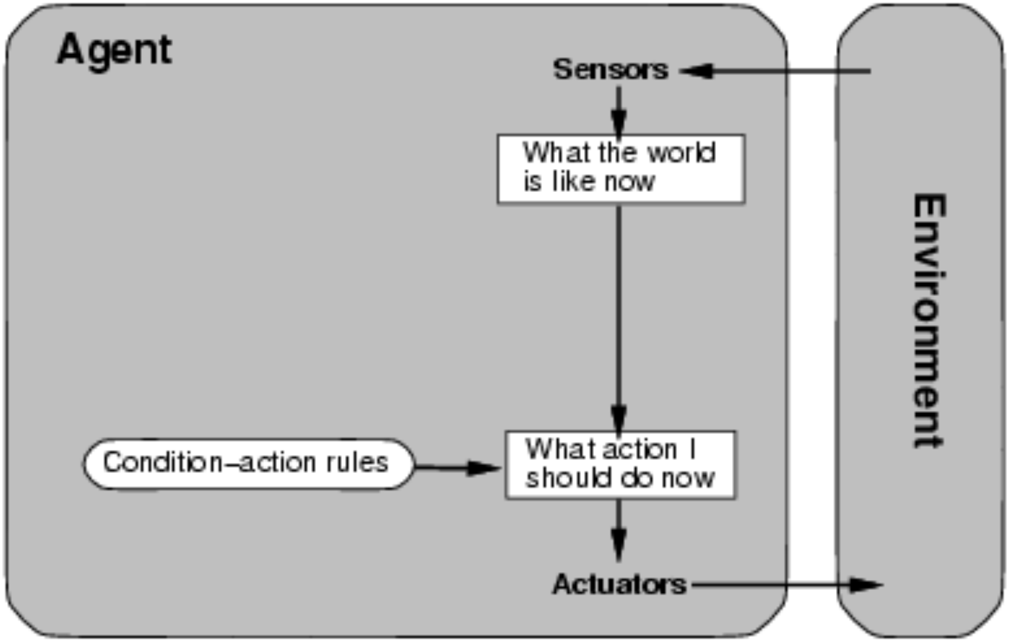
# From PEAS to Agent program

- Now we know how specify a “problem” by PEAS (with various properties).
- Rational agents are the “solutions” to the “problem”
- The job of AI is to design an **agent program** that implements the agent function— the mapping from percepts to actions.

# Agent types

- **Simple reflex agents**
  - select actions on the basis of the *current* percept, ignoring the rest of the percept history.
- **Model-based reflex agents**
  - use a model of the world to choose their actions. They maintain an internal state.
- **Goal-based agents**
  - choose their actions in order to achieve goals.
  - **Search** (Chapters 3 to 5) and **planning** (Chapters 10 and 11) are the subfields of AI devoted to finding action sequences that achieve the agent's goals.
- **Utility-based agents**
  - choose actions based on a preference (utility) for each state. Goals are inadequate when
    - There are conflicting goals, out of which only few can be achieved.
    - Goals have some uncertainty of being achieved and you need to weigh likelihood of success against the importance of a goal.
- All these can be turned into **learning agents**

# Simple reflex agents



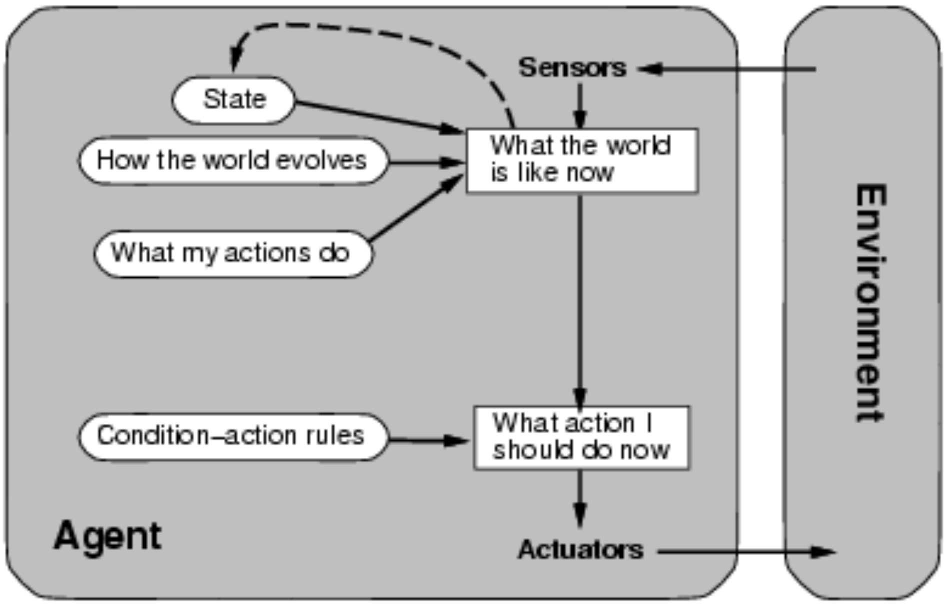

**function** REFLEX-VACUUM-AGENT( [*location*, *status*]) **returns** an action

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

# Simple reflex agents

- 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

# Model-based reflex agents




- 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

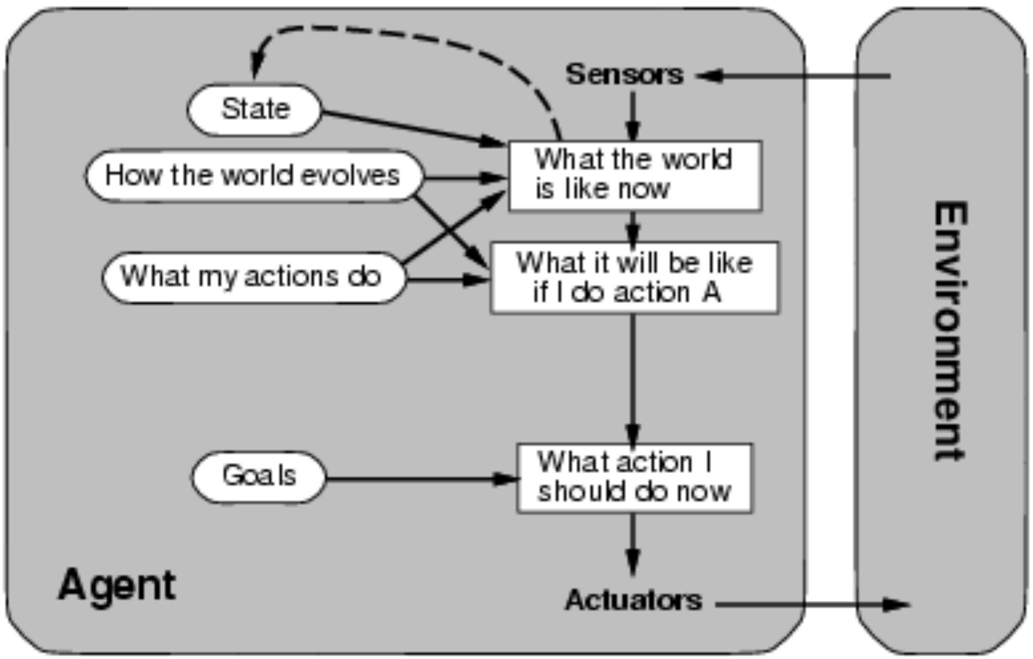
```
function REFLEX-AGENT-WITH-STATE(percept) returns action
   static: state, a description of the current world state
           rules, a set of condition-action rules

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

# Goal-based agents

- 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

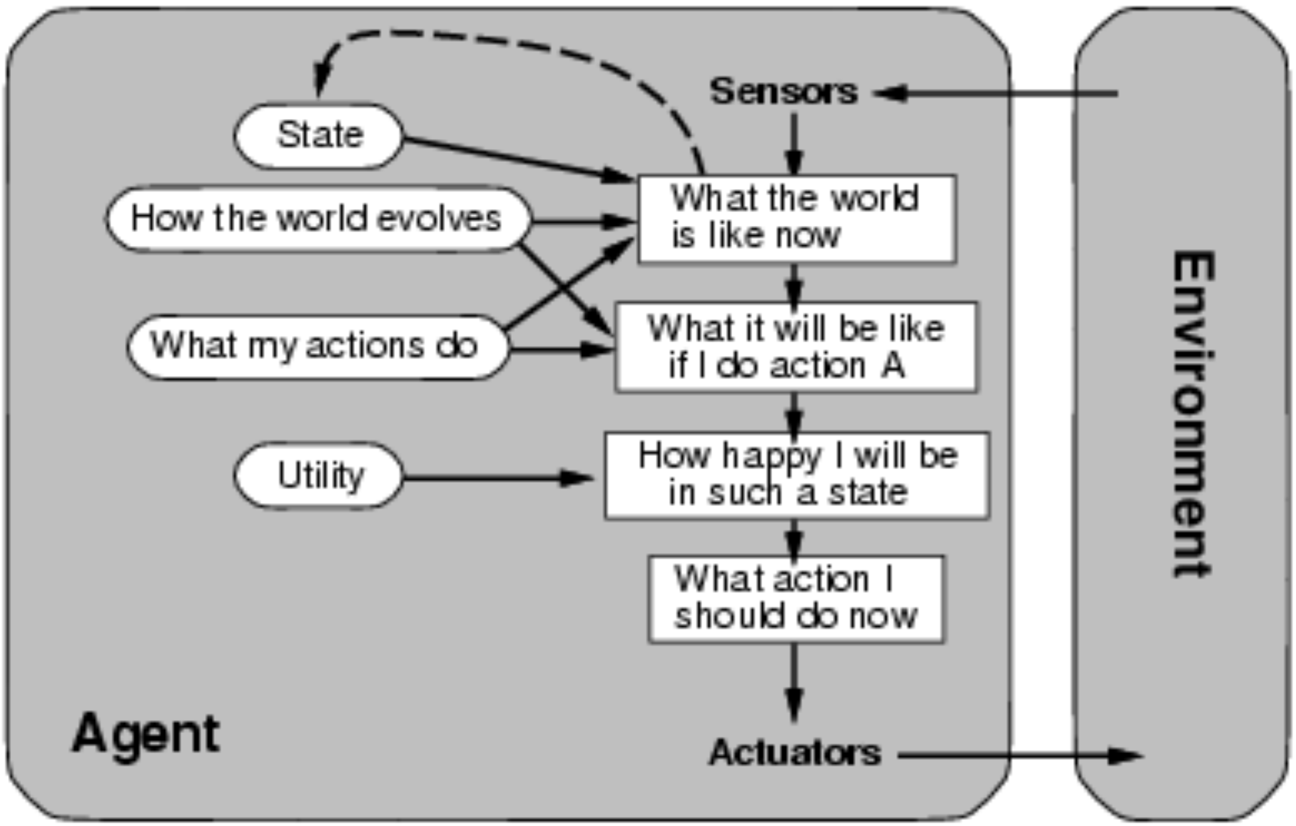



- Reflex agent breaks when it sees brake lights. Goal based agent reasons
  - Brake light -> car in front is stopping -> I should stop -> I should use brake

# Utility-based agents

- 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



# Learning agents

- All agents can improve their performance through **learning**.
- to build learning machines and then to teach them.



36