This course is about designing intelligent agents

- Agents and environments
  - The vacuum-cleaner world
- Rationality
  - The concept of rational behavior.
- Environment types
- Agent types
An agent is an entity that perceives and acts in an environment
- environment can be real or virtual

An agent can always perceive its actions, but not necessarily their effects on the environment

Rational agent: optimizes some performance criterion
- For any given task and class of environments we seek the agent (or class of agents) with the best performance.

Problem:
- computational limitations make perfect rationality unachievable
The Vacuum-Cleaner world

- A robot-vacuum-cleaner that operates in a simple world

Environment:
- Virtual house with room A and room B

Percepts:
- The robot can sense pairs $[<\text{location}>,<\text{status}>]$
  - Location: whether it is in room $A$ or $B$
  - Status: whether the room is Clean or Dirty

Actions:
- Left, Right, Suck, NoOp
Rational Agent – Performance Measure

- A **rational agent** is an agent that “does the right thing”
  - intuitively clear, but needs to be measurable in order to be useful for computer implementation

- **Performance Measure:**
  - a function that evaluates sequence of actions/environment states
  - obviously not fixed but task-dependent

- **Vacuum-World performance measures:**
  - reward for the amount of dust cleaned
    - one point per square cleaned up in time $T$
    - can be maximized by dumping dust on the floor again...
  - reward for clean floors
    - one point per clean square per time step
    - possibly with penalty for consumed energy
      - minus one per move?

- **General rule:**
  - design performance measure based on desired environment state
  - not on desired agent behavior
Rational Agent

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date and prior environment knowledge.

- Rational ≠ omniscient
  - An omniscient agent knows the actual outcome of its actions.

- Rational ≠ successful
  - Rationality maximizes expected performance
  - This may not be the optimal outcome
  - Example:
    - the expected monetary outcome of playing in the lottery/casino, etc. is negative (hence it is rational not to play)
    - but if you're lucky, you may win...
PEAS

What is rational at a given time depends on four things:

- **P**: the performance measure that defines the success
- **E**: the agent's prior knowledge of the environment
- **A**: the actions that the agent can perform
- **S**: the agent's percept sequence to date

*Example: Fully automated Taxi*
PEAS

What is rational at a given time depends on four things:

- **P**: the performance measure that defines the success
- **E**: the agent's prior knowledge of the environment
- **A**: the actions that the agent can perform
- **S**: the agent's percept sequence to date

**Example: Fully automated Taxi**

- **Performance**
  - Safety, destination, profits, legality, comfort
- **Environment**
  - Streets/freeways, other traffic, pedestrians, weather, …
- **Actuators**
  - Steering, accelerating, brake, horn, speaker/display,…
- **Sensors**
  - Video, sonar, speedometer, engine sensors, keyboard, GPS, …
PEAS

What is rational at a given time depends on four things:

- **P**: the *performance measure* that defines the success
- **E**: the agent's prior knowledge of the *environment*
- **A**: the *actions* that the agent can perform
- **S**: the agent's *percept sequence* to date

*Example*: Internet Shopping Agent
What is rational at a given time depends on four things:

- **P**: the performance measure that defines the success
- **E**: the agent's prior knowledge of the environment
- **A**: the actions that the agent can perform
- **S**: the agent's percept sequence to date

**Example: Internet Shopping Agent**

- **Performance**
  - price, quality, appropriateness, efficiency
- **Environment**
  - the Web: current and future WWW sites, vendors, shippers
- **Actuators**
  - display to user, follow URL, fill in form
- **Sensors**
  - parsing of HTML pages (text, graphics, scripts)…
PEAS

What is rational at a given time depends on four things:

- **P**: the *performance measure* that defines the success
- **E**: the agent's prior knowledge of the *environment*
- **A**: the *actions* that the agent can perform
- **S**: the agent's *percept sequence* to date

*Example: Chess Program*
PEAS

What is rational at a given time depends on four things:

- **P**: the performance measure that defines the success
- **E**: the agent's prior knowledge of the environment
- **A**: the actions that the agent can perform
- **S**: the agent's percept sequence to date

**Example: Chess Program**

- **Performance**
  - number of games won, ELO rating,...
- **Environment**
  - the chess board
- **Actuators**
  - moves that can be performed
- **Sensors**
  - placement of pieces in current position, whose turn is it?, ...
Environment Types

- **Fully observable**
  - the complete state of the environment can be sensed
  - at least the relevant parts
  - no need to keep track of internal states

- **Partially observable**
  - parts of the environment cannot be sensed

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td></td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td></td>
</tr>
<tr>
<td>Poker</td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td></td>
</tr>
<tr>
<td>Image Analysis</td>
<td></td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td></td>
</tr>
<tr>
<td>Refinery Controller</td>
<td></td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td></td>
</tr>
</tbody>
</table>
Environment Types

- **Deterministic**
  - the next environment state is completely determined by the current state and the executed action
- **Strategic**
  - only the opponents' actions cannot be foreseen
- **Stochastic**

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td>Fully</td>
<td></td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td>Fully</td>
<td></td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td></td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Fully</td>
<td></td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td>Partially</td>
<td></td>
</tr>
<tr>
<td>Refinery Controller</td>
<td>Partially</td>
<td></td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td>Partially</td>
<td></td>
</tr>
</tbody>
</table>
# Environment Types

- **Episodic**
  - the agent’s experience can be divided into atomic steps
  - the agent perceives and then performs a single action
  - the choice of action depends only on the episode itself
- **Sequential**
  - the current decision could influence all future decision

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td>Fully</td>
<td>Deterministic</td>
<td></td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Stochastic</td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Strategic</td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td></td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td></td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td></td>
</tr>
<tr>
<td>Refinery Controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td></td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td></td>
</tr>
</tbody>
</table>
Environment Types

- **Dynamic**
  - the environment may change while the agent deliberates
- **Static**
  - the environment does not change
- **Semidynamic**
  - the environment does not change, but the performance score may

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Strategic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td></td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Episodic</td>
<td></td>
</tr>
<tr>
<td>Refinery Controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td></td>
</tr>
</tbody>
</table>
Environment Types

- **Discrete**
  - finite number of actions / environment states / percepts

- **Continuous**
  - actions, states, percepts are on a continuous scale
  - this distinction applies separately to actions, states, and percepts
  - can be mixed in individual tasks

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td>Static</td>
<td>Semi</td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Semi</td>
<td></td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Static</td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Episodic</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>Refinery Controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td></td>
</tr>
</tbody>
</table>
Environment Types

- **Single-Agent**
  - No other agents (other agents may be part of the environment)

- **Multi-Agent**
  - Does the environment contain other agents whose performance measure depends on my actions?
  - other agents may be co-operative or competitive

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Semi</td>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Episodic</td>
<td>Dynamic</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>Refinery Controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Discrete</td>
<td></td>
</tr>
</tbody>
</table>
# Environment Types

- **Single-Agent**
  - No other agents (other agents may be part of the environment)

- **Multi-Agent**
  - Does the environment contain other agents whose performance measure depends on my *actions*?
  - other agents may be *co-operative* or *competitive*

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudoku</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Single</td>
</tr>
<tr>
<td>Chess With a Clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Semi</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Part-Picking Robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Episodic</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Refinery Controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Interactive Tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
</tbody>
</table>
Environment Types

- The simplest environment is
  - fully observable
  - deterministic
  - episodic
  - static
  - discrete
  - single-agent
Environment Types

- The simplest environment is
  - fully observable
  - deterministic
  - episodic
  - static
  - discrete
  - single-agent

- Most real situations are
  - partially observable
  - stochastic
  - sequential
  - dynamic
  - continuous
  - multi-agent
Agent Function

- The *agent function* maps percept histories to actions.

\[ f : P^* \rightarrow A \]

- The agent function will internally be represented by the **agent program**.
- The agent program runs on the physical architecture to produce \( f \).
A Simple Vacuum Cleaner Agent

- Table of mappings from Percept sequences to actions:

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B, Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
A Simple General Agent

**function** `TABLE-DRIVEN-AGENT(percept)` **returns** an action

**static:** `percepts`, a sequence initially empty

  `table`, a table of actions, indexed by percept sequence

append `percept` to the end of `percepts`

`action ← LOOKUP(percepts, table)`

**return** `action`

- has a table of all possible percept histories
- looks up the right response in the table
- Clearly infeasible:
  - if there are $|P|$ percepts and a life-time of $T$ time steps, we need a look-up table of size $\sum_{t=1}^{T} |P^t|$
- For example: chess:
  - about 36 moves per position, average game-length 40 moves
    $\rightarrow 5105426007029058700898070779698222806522450657188621232590965$
A Simple Vacuum Cleaner Agent

- **Strategy**

  “If current room is dirty then suck, otherwise move to the other room.”

- **As an agent program**

  ```python
  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
  ```

- **Obvious Questions:**
  - Is this the right agent?
  - Is this a good agent?
  - Is there a right agent?
Agent Programs

The key challenge for AI is to write programs that produce rational behavior from a small amount of code rather than a large number of table entries.

- Writing down the agent functions is not practical for real applications.

- But feasibility is also important:
  - you can write a perfect chess playing agent with a few lines of code
  - it will run forever, though...

- Agent = architecture + program
Agent Types

- Four basic kinds of agent programs will be discussed:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

- All these can be turned into learning agents.
Simple Reflex Agent

- Select action on the basis of only the current percept
  - ignores the percept history
Simple Reflex Agent

- Select action on the basis of only the current percept
  - ignores the percept history
- Implemented through condition-action rules
- Large reduction in possible percept/action situations
  - from $\sum_{t=1}^{T} |P^t|$ to $|P|$ 
- But will make a very bad chess player
  - does not look at the board, only at the opponent's last move (assuming that the sensory input is only the last move, no visual)

Example:

```python
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
```
General Simple Reflex Agent

```plaintext
function SIMPLE-REFLEX-AGENT(percept) returns an action

static: rules, a set of condition-action rules

state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state, rule)
action ← RULE-ACTION[rule]
return action
```

- Note that rules are just used as a concept
  - actual implementation could, e.g., be logical circuitry
- Will only work if the environment is fully observable
  - everything important needs to be determinable from the current sensory input
  - otherwise infinite loops may occur
    - e.g. in the vacuum world without a sensor for the room, the agent does not know whether to move right or left
    - possible solution: randomization
Model-Based Reflex Agent

- Keep track of the state of the world
  - better way to fight partial observability
General Model-Based Reflex Agent

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

\[
\text{state } \leftarrow \text{UPDATE-STATE}(\text{state}, \text{action}, \text{percept})
\]
\[
\text{rule } \leftarrow \text{RULE-MATCH}(\text{state}, \text{rule})
\]
\[
\text{action } \leftarrow \text{RULE-ACTION}[\text{rule}]
\]
\[
\text{return } \text{action}
\]

- Input is not only interpreted, but mapped into an internal state description (a world model)
  - a chess agent could keep track of the current board situation when its percepts are only the moves
- Internal state is also used for interpreting subsequent percepts
- The world model may include effects of own actions!
Goal-Based Agent

- the agent knows what states are desirable
  - it will try to choose an action that leads to a desirable state

```plaintext
- how the world evolves
- what my actions do
- state
- what the world is like now
- what it will be like if I do action A
- goals
- what action I should do now

Sensors

Environment

Actuators

project consequences of actions into the future
compare the expected consequences to goals
```
Goal-Based Agent

- the agent knows what states are desirable
  - it will try to choose an action that leads to a desirable state
- things become difficult when long sequences of actions are required to find the goal
  - typically investigated in search and planning research
- main difference to previous approaches
  - decision-making takes future into account
    - “What will happen if I do such-and-such?”
    - “Will this make me happy?”
- is more flexible since knowledge is represented explicitly and can be manipulated
  - changing the goal does not imply changing the entire set of condition-action rules
Utility-Based Agent

- Goals provide just a binary happy/unhappy distinction
  - utility functions provide a continuous scale
Utility-Based Agent

- Goals provide just a binary happy/unhappy distinction
  - utility functions provide a continuous scale
- Certain goals can be reached in different ways.
  - “Alle Wege führen nach Rom”
  - Some ways are quicker, safer, more reliable, cheaper, ...
    → have a higher utility
- **Utility function**
  - maps a state (or a sequence of states) onto a real number
- Improves on goals:
  - selection between conflicting goals (e.g., speed and safety)
  - selection between goals based on trade-off between likelihood of success and importance of goal
Learning

- All previous agent-programs describe methods for selecting actions
  - yet they do not explain the origin of these programs.
- Learning mechanisms can be used for acquiring programs
  - teach them instead of instructing them
- Advantage
  - robustness of the program toward initially unknown environments.
- Every part of the previous agents can be improved with learning

Learning in intelligent agents can be summarized as a process of modification of each component of the agent to bring the components into closer agreement with the available feedback information, thereby improving the overall performance of the agent.
Learning Agent

- **Critic**
  - Performance standard
  - feedback
- **Learning element**
  - changes
  - learning goals
- **Problem generator**
- **Performance element**
- **Sensors**
- **Environment**
- **Actuators**

Agent
Learning Agent

- **Performance element**
  - makes the action selection (as usual)

- **Critic**
  - decides how well the learner is doing with respect to a fixed *performance standard*
  - necessary because the percepts do not provide any indication of the agent's success
    - e.g., it needs to know that checkmate is bad

- **Learning element**
  - improves the performance element
  - its design depends very much on the performance element

- **Problem generator**
  - responsible for *exploration* of new knowledge
    - sometimes try new, possibly suboptimal actions to acquire knowledge about their consequences
  - otherwise only *exploitation* of (insufficient) current knowledge