Fundamentals of Artificial Intelligence – Intelligent Agents

Matthias Althoff

TU München

Winter semester 2023/24

Organization

1. Agents and Environments
2. The Concept of Rationality
3. The Nature of Environments
4. The Structure of Agents

The content is covered in the AI book by the section “Intelligent Agents”.

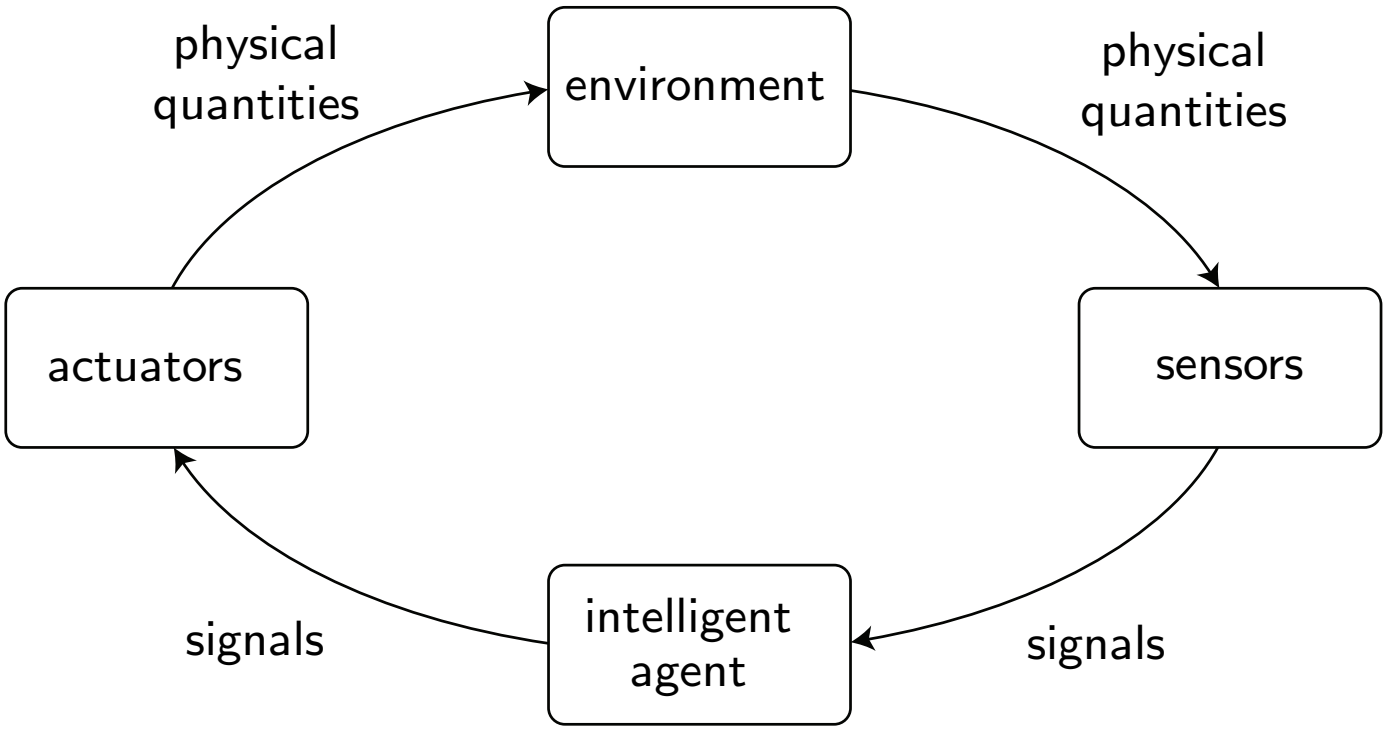
Learning Outcomes

- You can recall the definition and understand the basic concept of intelligent agents.
- You understand the difference between *omniscience*, *learning*, and *autonomy*.
- You can create simple agent functions and agent programs describing the behavior of agents.
- You know how to categorize task environments and can evaluate the difficulty of given tasks.
- You know major categories of types of agents and can group an agent into one of them.

Intelligent Agent: Definition

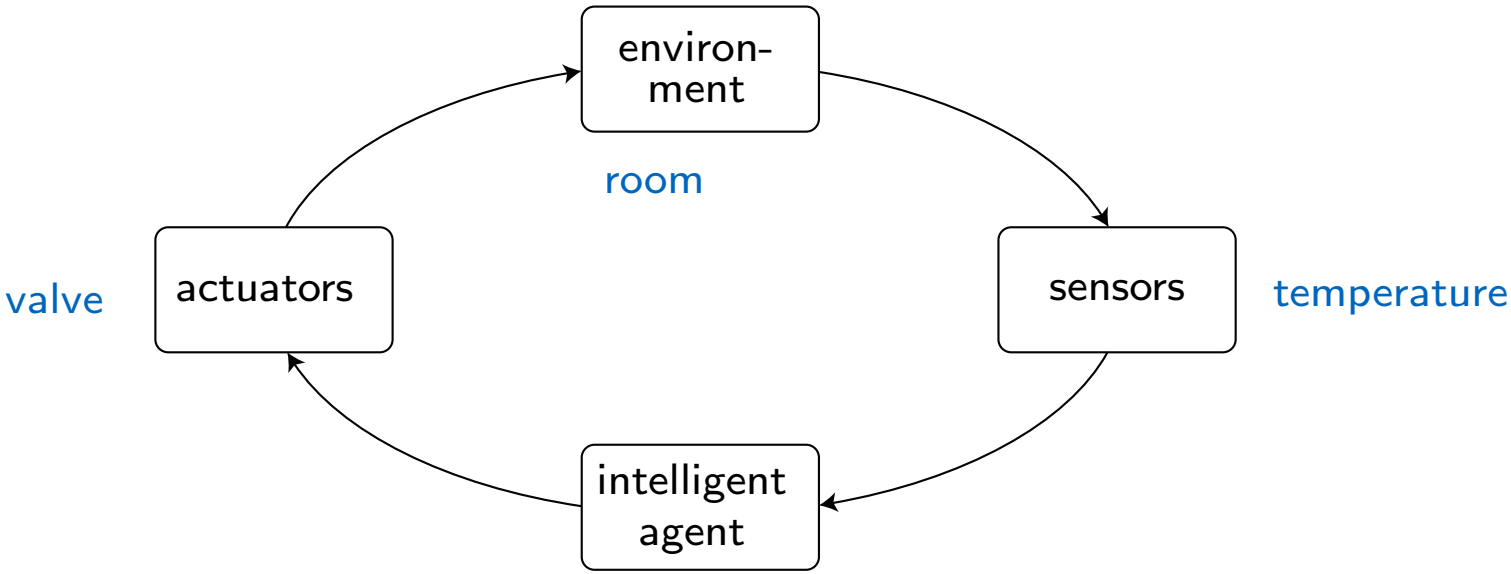
An intelligent agent is anything that

- perceives its environment through sensors and
- acts upon the environment through actuators.



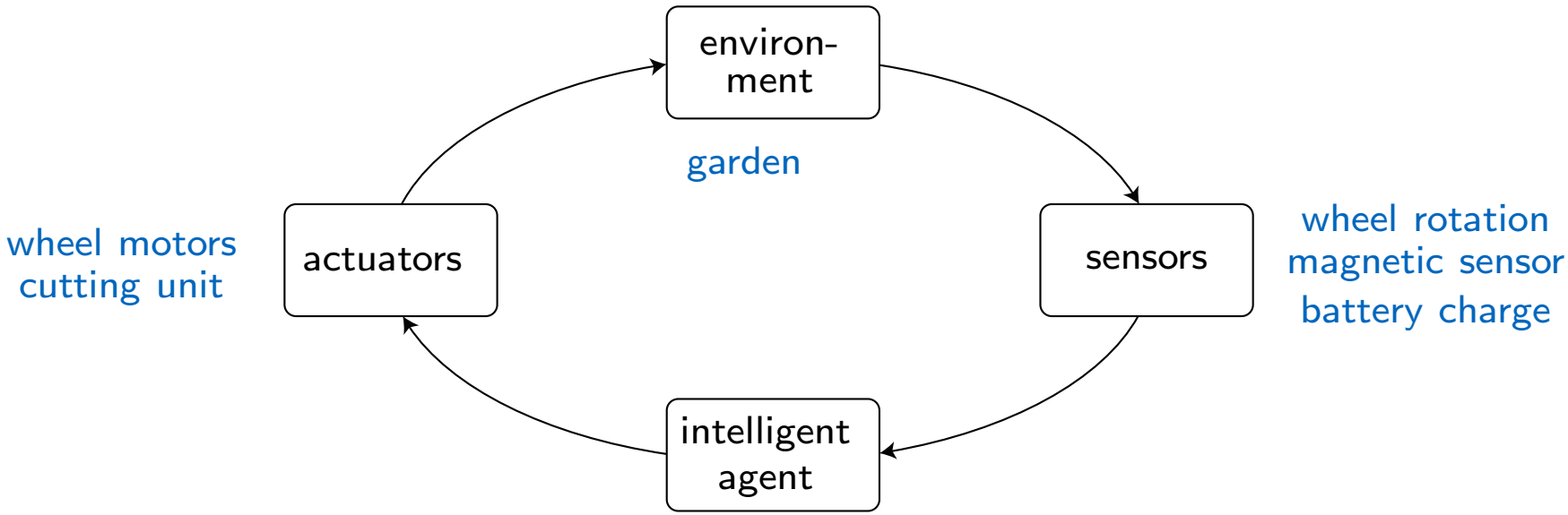
Example: Thermostat

source: Heimeier



Example: Robotic Lawn Mower

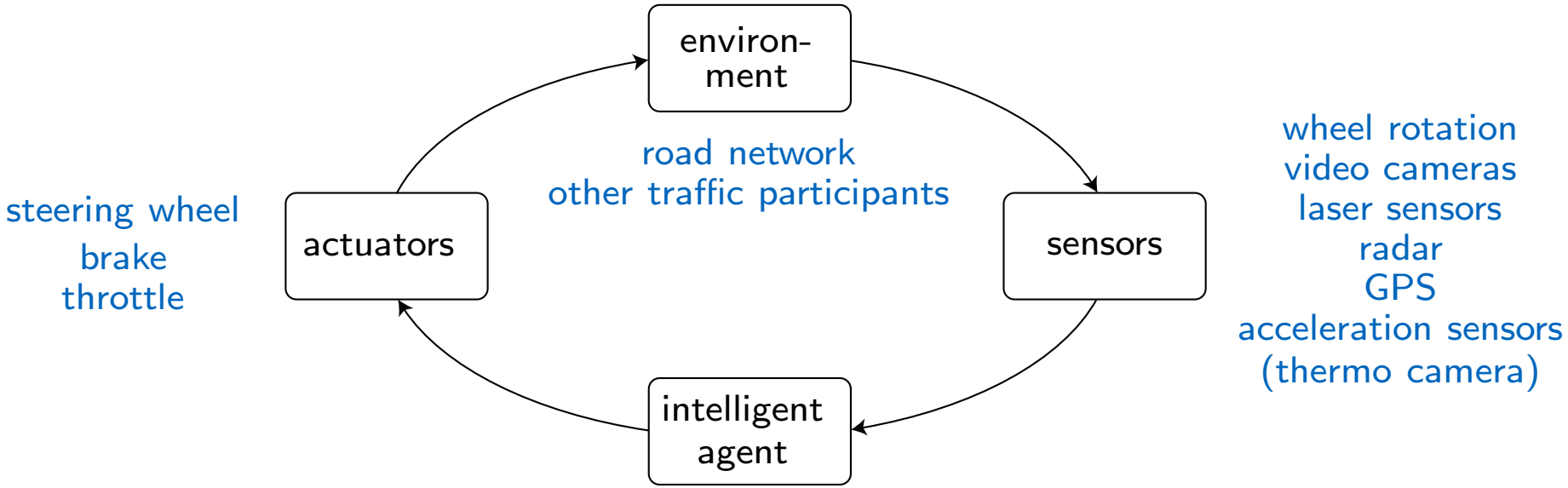
source: Bosch



(The magnetic sensor is required for the border wire)

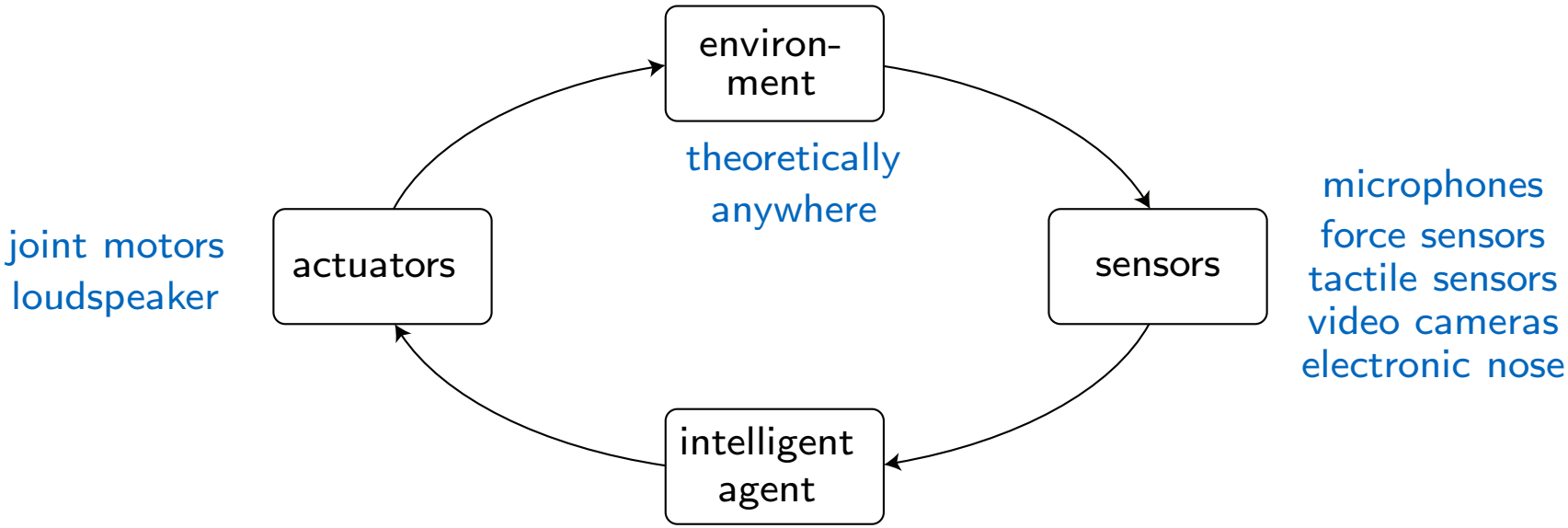
Example: Automated Car

source: Carnegie Mellon University



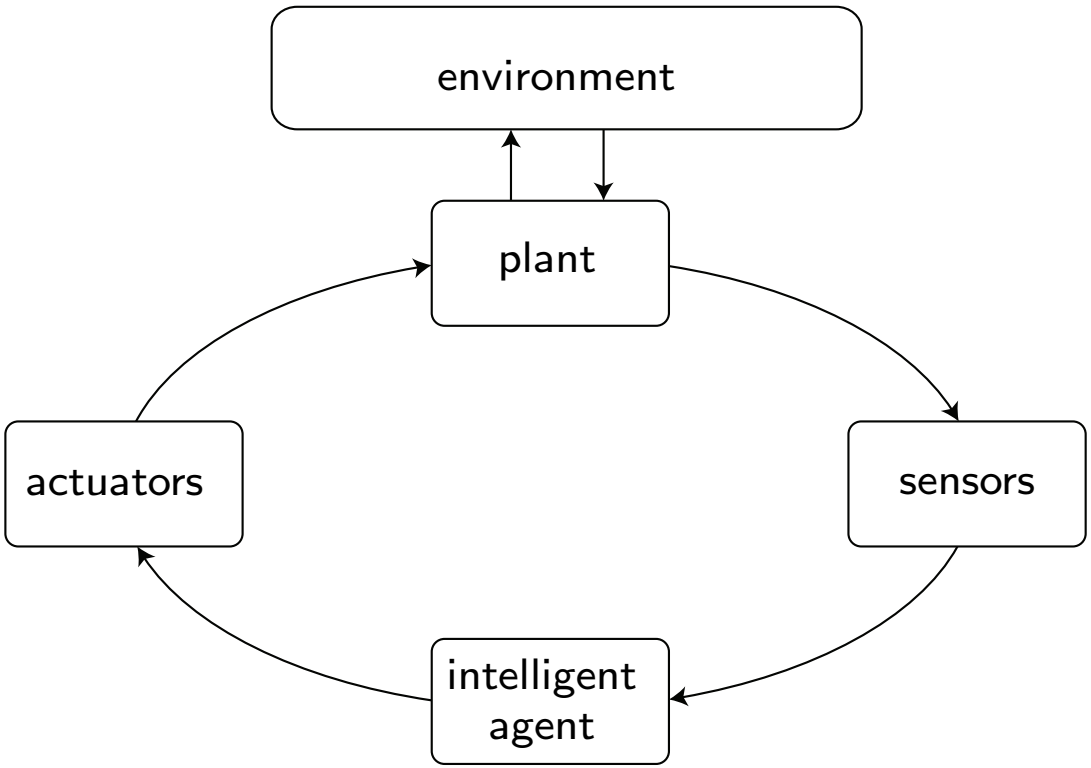
Example: Humanoid

source: Kawada Industries



Difference to a Control System View

In control theory, one typically distinguishes between the system one wants to control and the environment. In the AI setting, this distinction is often not made.

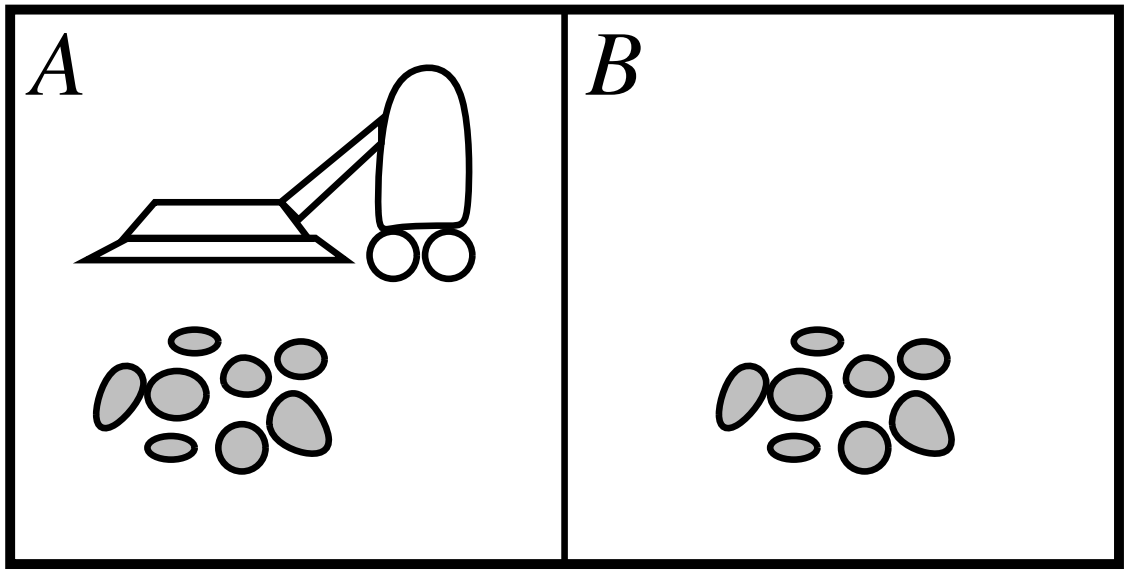


Tweedback Questions

What of the following objects is an agent?

- Pen?
- Plant?
- Cat?
- Carnivorous plant?

Vacuum-Cleaner World



Percepts: location and contents, e.g., [*A*, *Dirty*]
Actions: *Left*, *Right*, *Suck*, *NoOp* (**No Op**eration)

Agent Function

Percept sequence

An agent's percept sequence is the complete history of its perception.
Vacuum cleaner example: $[A, Dirty]$, $[A, Clean]$, $[B, Clean]$, $[A, Clean]$.

The behavior of an agent can be fully described by its agent function:

Agent function

An agent function maps any given percept sequence to an action.

Depending on the length of the percept sequence, the agent can make smarter choices. Why?

Tabular 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*
⋮	⋮

An agent program realizing 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*

Tweedback Questions

What would you prefer?

A **Tabular Agent Function:**
A table is simple! It makes it unnecessary to write a program.

B **Agent Program:**
Why should I fill out a stupid table if I can just write a small program?

Comments on Agent Functions

- **Expressiveness:** Tabular agent functions can theoretically describe the behavior of agents.
- **Practicality:** Tabular functions have no practical use since they are infinite or very large when one only considers finite percept sequences.

  Examples of table sizes:
  - 1 h recording of a camera (640x480 pixel, 24 bit color): $10^{250,000,000,000}$
  - Chess: 10^{150}

  (estimated number of atoms in the universe: 10^{80})
- **Solution:** An agent program is a practical implementation of an agent function.
- **What is a good agent function?** This question is answered by the concept of rational agents (next slide).

Rational Agent (I)

Rationality

A system is rational if it does the "right thing", i.e., has an ideal performance.

- An obvious performance measure is not always available.
- A designer has to find an acceptable measure.

Example vacuum cleaner:

- **Option 1**: Amount of dirt cleaned up in a certain amount of time. *Problem*: An optimal solution could be to clean up a section, dump the dirt on it, clean it up again, and so on.
- **Option 2**: Reward clean floors by providing points for each clean floor at each time step.

Rational Agent (II)

What is rational at any given time depends on

- the performance measure;
- the agent's prior knowledge of the environment;
- the actions that the agent can perform;
- the agent's percept sequence up to now.

Rational Agent

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the prior percept sequence and its built-in knowledge.

Omniscience, Learning, and Autonomy

Omniscient agent

An omniscient agent **knows** the actual outcome of its actions, which is impossible in reality.
Example: Just imagine you know the outcome of betting money on something.
A rational agent ($\neq$ omniscient agent) maximizes **expected** performance.

Learning

Rational agents are able to learn from perception, i.e., they improve their knowledge of the environment over time.

Autonomy

In AI, a rational agent is considered more autonomous if it is less dependent on prior knowledge and uses newly learned abilities instead.

Task Environment (I)

To design a rational agent, we have to specify the task environment. We use the PEAS (**p**erformance, **e**nvironment, **a**ctuators, **s**ensors) description:

Automated taxi

Performance Measure	**Environment**	**Actuators**	**Sensors**
safety,	streets/freeways,	steering,	video,
time,	traffic,	accelerator,	accelerometers,
profits,	pedestrians,	brake,	lidar, ...
legality,	weather, ...	horn,	radar,
comfort,	...	speaker/display,	GPS,
...		...	...

Task Environment (II)

To design a rational agent, we have to specify the task environment. We use the PEAS (**p**erformance, **e**nvironment, **a**ctuators, **s**ensors) description:

Internet shopping agent

Performance Measure	**Environment**	**Actuators**	**Sensors**
price, quality, appropriateness, efficiency	websites, vendors, shippers	display to user, follow URL, fill in form	HTML pages

Properties of Task Environments (I)

Fully observable vs. **partially observable**

An environment is fully observable if the agent can detect the complete state of the environment, and partially observable otherwise.

Example: The vacuum-cleaner world is partially observable since the robot only knows whether the current square is dirty.

Single agent vs. **multi agent**

An environment is a multi agent environment if it contains several agents, and a single agent environment otherwise.

Example: The vacuum-cleaner world is a single agent environment. A chess game is a two-agent environment.

Properties of Task Environments (II)

Deterministic vs. **stochastic**

An environment is deterministic if its next state is fully determined by its current state and the action of the agent, and stochastic otherwise.

Example: The automated taxi driver environment is stochastic since the behavior of other traffic participants is unpredictable. The outcome of a calculator is deterministic.

Episodic vs. **sequential**

An environment is episodic if the actions taken in one episode (in which the robot senses and acts) does not affect later episodes, and sequential otherwise.

Example: Detecting defective parts on a conveyor belt is episodic. Chess and automated taxi driving are sequential.

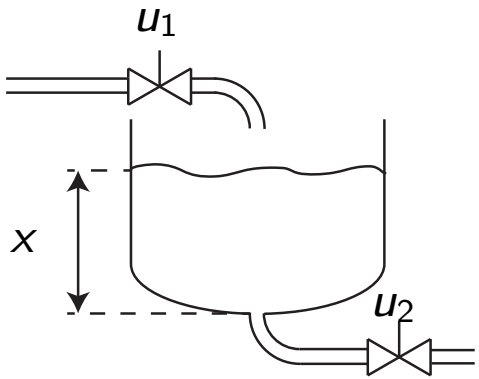
Properties of Task Environments (III)

Discrete vs. **continuous**

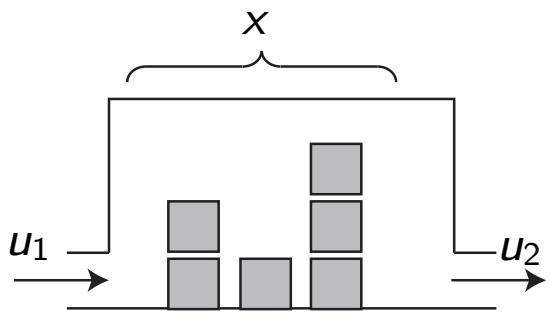
The discrete/continuous distinction applies to the *state* and the *time*:

- continuous state + continuous time: e.g., robot
- continuous state + discrete time: e.g., weather station
- discrete state + continuous time: e.g., traffic light control
- discrete state + discrete time: e.g., chess

Example for discrete and continuous state:



continuous: tank



discrete: warehouse

Properties of Task Environments (IV)

Static vs. **dynamic**

If an environment only changes based on actions of the agent, it is static, and dynamic otherwise.

Example: The automated taxi driver environment is dynamic. A crossword puzzle is static.

Known vs. **unknown**

An environment is known if the agent knows the outcomes (or outcome probabilities) of its actions, and unknown otherwise. In the latter case, the agent has to learn the environment first.

Example: The agent knows all the rules of a card game it should play, thus it is in a known environment.

Examples of Task Environments

Task environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	fully	single	deterministic	seq.	static	discrete
Chess with a clock	fully	multi	deterministic	seq.	semi	discrete
Poker	partially	multi	stochastic	seq.	static	discrete
Backgammon	fully	multi	stochastic	seq.	static	discrete
Taxi driving	partially	multi	stochastic	seq.	dynamic	cont.
Medical diagnosis	partially	single	stochastic	seq.	dynamic	cont.
Image analysis	fully	single	deterministic	episodic	semi	cont.
Part-picking robot	partially	single	stochastic	episodic	dynamic	cont.
Refinery controller	partially	single	stochastic	seq.	dynamic	cont.
Interactive English tutor	partially	multi	stochastic	seq.	dynamic	discrete

The above categorizations can be debated in some cases. In some cases, one can only precisely tell the right category if more detailed information about the particular application is available.

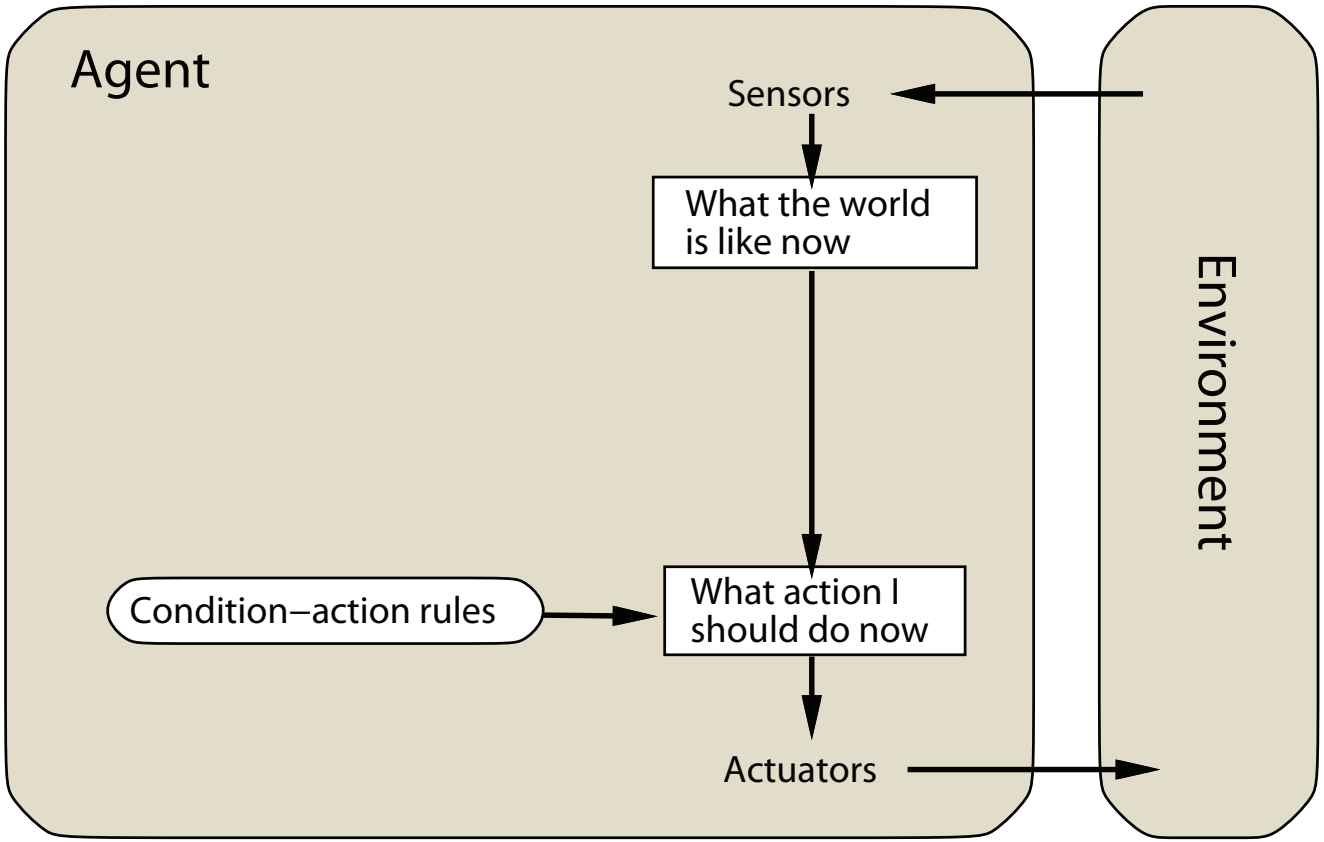
Agent Types

Besides categorizing the task environment, we also categorize agents in four categories with increasing generality:

- simple reflex agents,
- reflex agents with state,
- goal-based agents,
- utility-based agents.

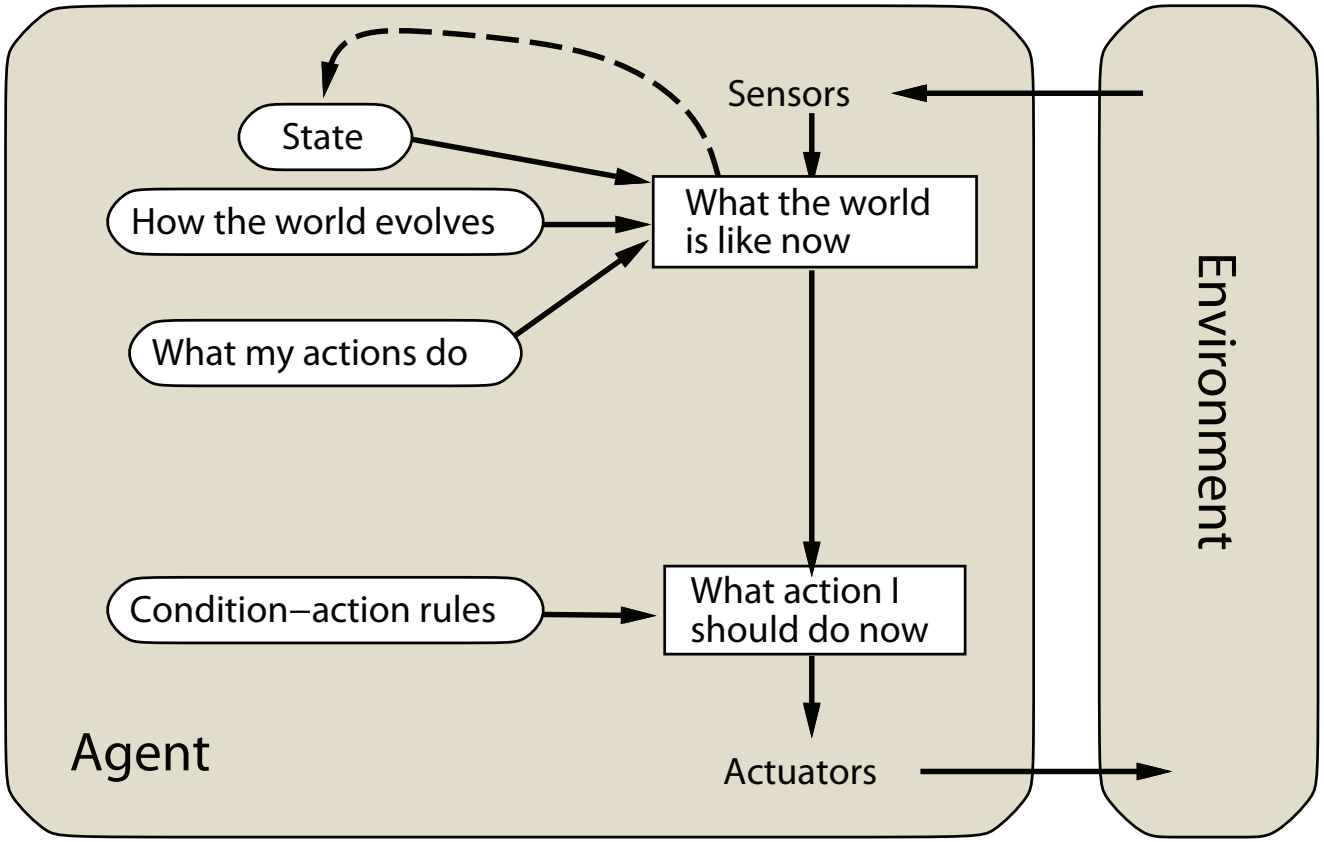
All these can be turned into learning agents.

Simple Reflex Agents



- Agent selects action on the basis of the current percept.
- The vacuum-cleaner program is the one of a simple reflex agent.
- Typically only works in fully observable environments and when all required condition-action rules are implemented. Why?

Model-Based Reflex Agents (I)



- Extension of the simple reflex agent.
- Partial observability is handled by keeping track of the environment the agent cannot perceive.

Model-Based Reflex Agents (II)

New components compared to simple reflex agents:

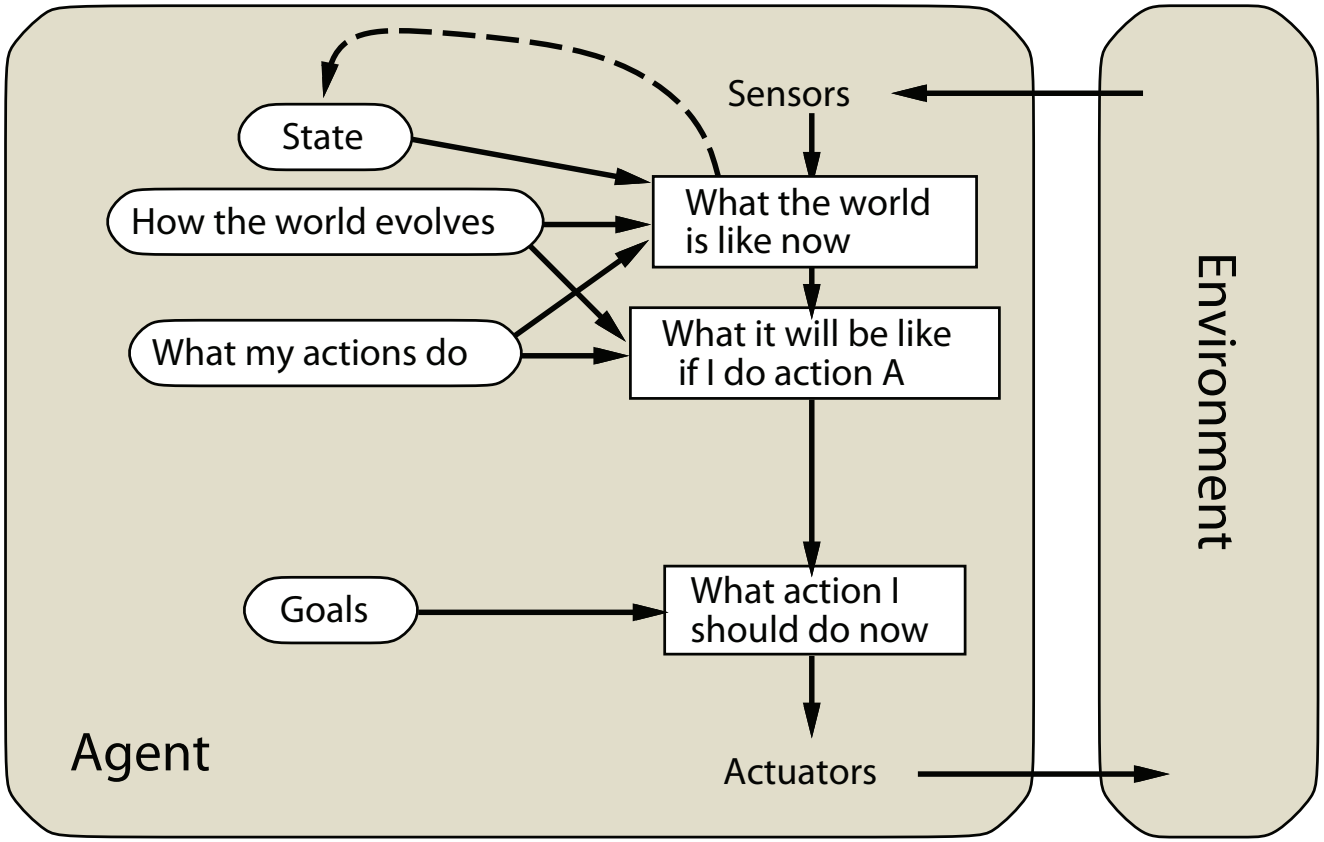
- **Internal state**: The agent keeps an internal state of the previous situation that depends on the percept history and thereby reflects unobservable aspects. It is an art to come up with "good" internal states (see later in the course).
- **How the world evolves**: Example of a pedestrian becoming unobservable:

source: Daimler

If the agent would not keep in mind that a pedestrian stepped behind a vehicle, it would recognize it too late when reappearing.

- **What my actions do**: Continuing the pedestrian example: Braking and accelerating will change the relative positions in the "new world".

Goal-Based Agents (I)



- Extension of the model-based reflex agents.
- In addition to knowing the current state of the environment, the goal of the agent is explicitly considered.

Goal-Based Agents (II)

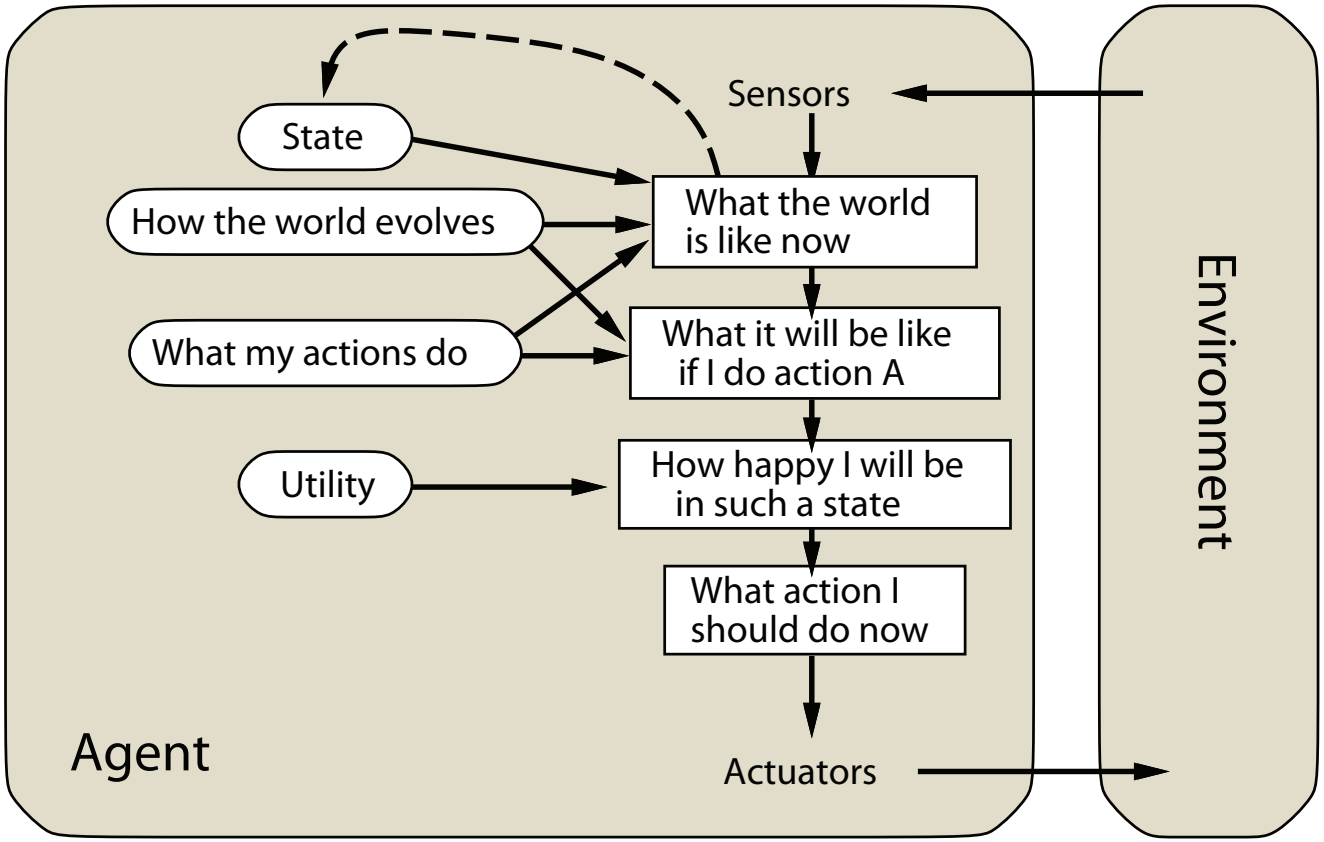
New aspect of goal-based agents: **What it will be like if I do action A?**

Continuing the pedestrian example

- Braking will prevent the vehicle from hitting the pedestrian.
- Continuing with the same velocity will hit the pedestrian.
- The final decision is made by checking which actions help achieving the goal. The system will brake to achieve the goal of arriving accident-free at the destination.

Reaching a goal is achieved by **search** and **planning**, which will be covered in this course.

Utility-Based Agents (I)



- Extension of the goal-based reflex agents.
- In addition to achieving the goal state, it should be reached with maximum utility, i.e., maximizing the "happiness" of the agent.

Utility-Based Agents (II)

New aspect of utility-based agents: **How happy will I be in a state?**

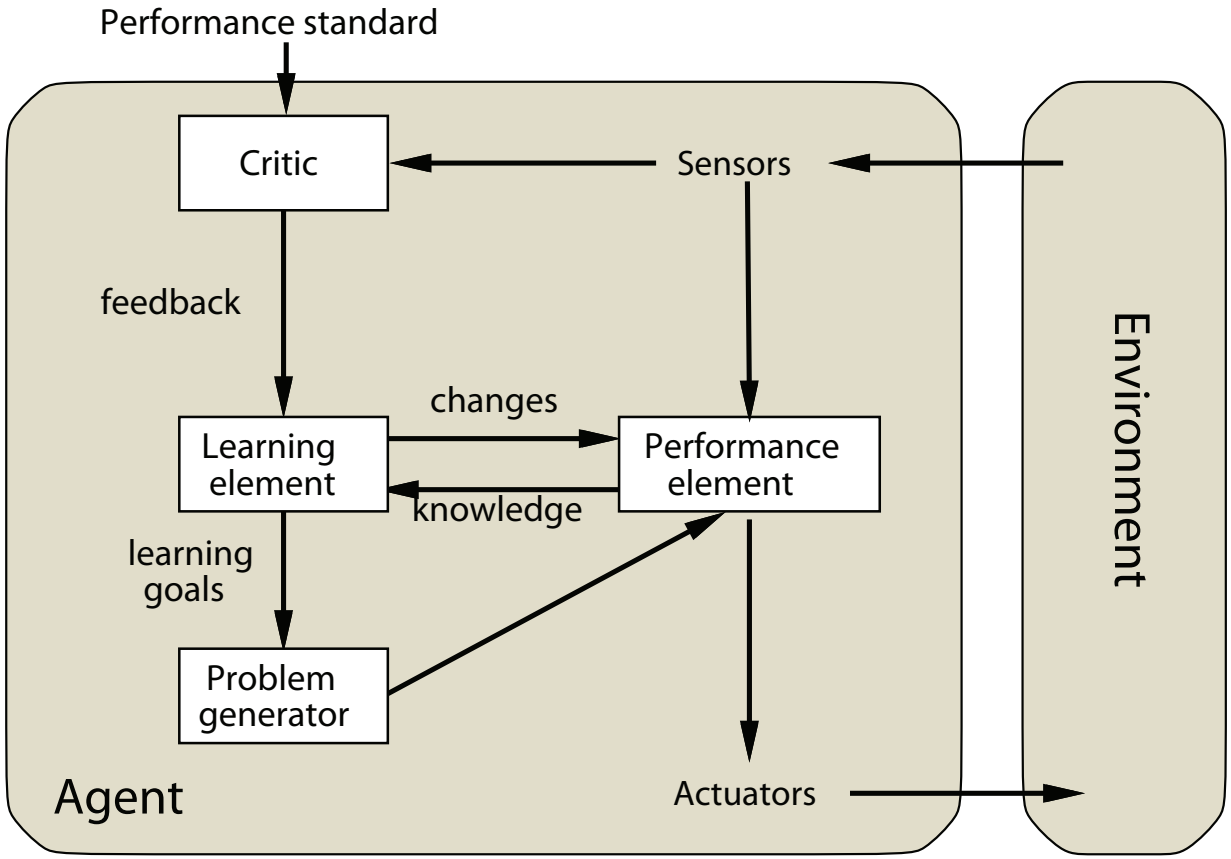
Continuing the pedestrian example

Small modification: The pedestrian has not yet stepped in the street.

- The utility is to arrive at the destination on time, while keeping the risk of an accident low.
- Braking will decrease the risk of hitting the pedestrian.
- Continuing with the same speed helps in getting to the destination on time, but increases the risk of an accident.
- The final decision is made by evaluating the utility function: The outcome will be a tradeoff between arriving on time and safety.

Maximizing the expected utility is especially challenging in stochastic or partially-observable environments, which will be covered in this course.

Learning Agents (I)



- Any previous agent can be extended to a learning agent.
- The block **Performance element** is a placeholder for the whole of any of the previous agents.

Learning Agents (II)

- **Performance element**: Placeholder for any of the previous entire agents; responsible for selecting the actions.
- **Learning element**: Makes improvements based on gained experience.
- **Critic**: Tells the learning element how well the agent is doing with respect to a fixed performance standard.
- **Problem generator**: Suggests actions leading to new and informative experiences to stimulate learning.

Continuing the pedestrian example

- After a near collision with a pedestrian, the critic tells the learning element that the minimum distance was too small.
- The learning agent adapts the rule of when to brake close to a pedestrian.
- The problem generator suggests changing lanes when close to pedestrians.
- The critic will tell the learning agent that changing lanes is an effective method and the learning agent adapts the performance element accordingly.

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 environment sequence.
- A **perfectly rational** agent maximizes expected performance.
- **Agent programs** implement (some) agent functions.
- **PEAS** descriptions define task environments.
- Environments are categorized along several dimensions: **observable**? **deterministic**? **episodic**? **static**? **discrete**? **single-agent**?
- Several basic agent architectures exist: **reflex**, **reflex with state**, **goal-based**, **utility-based**.