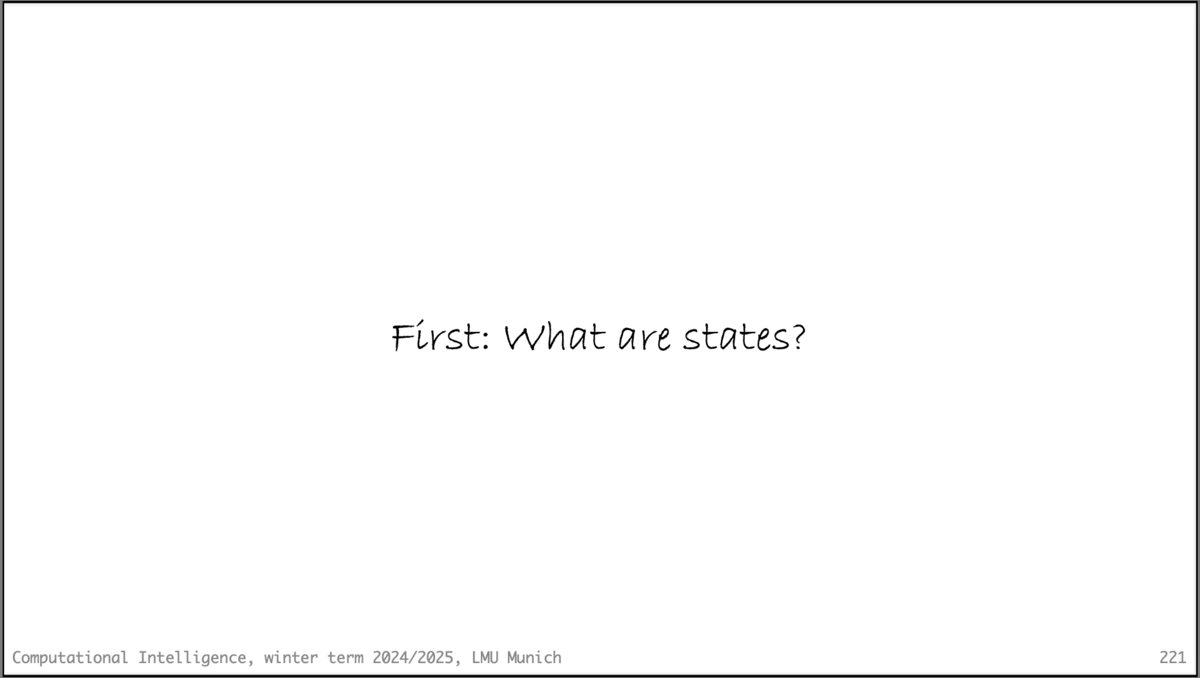
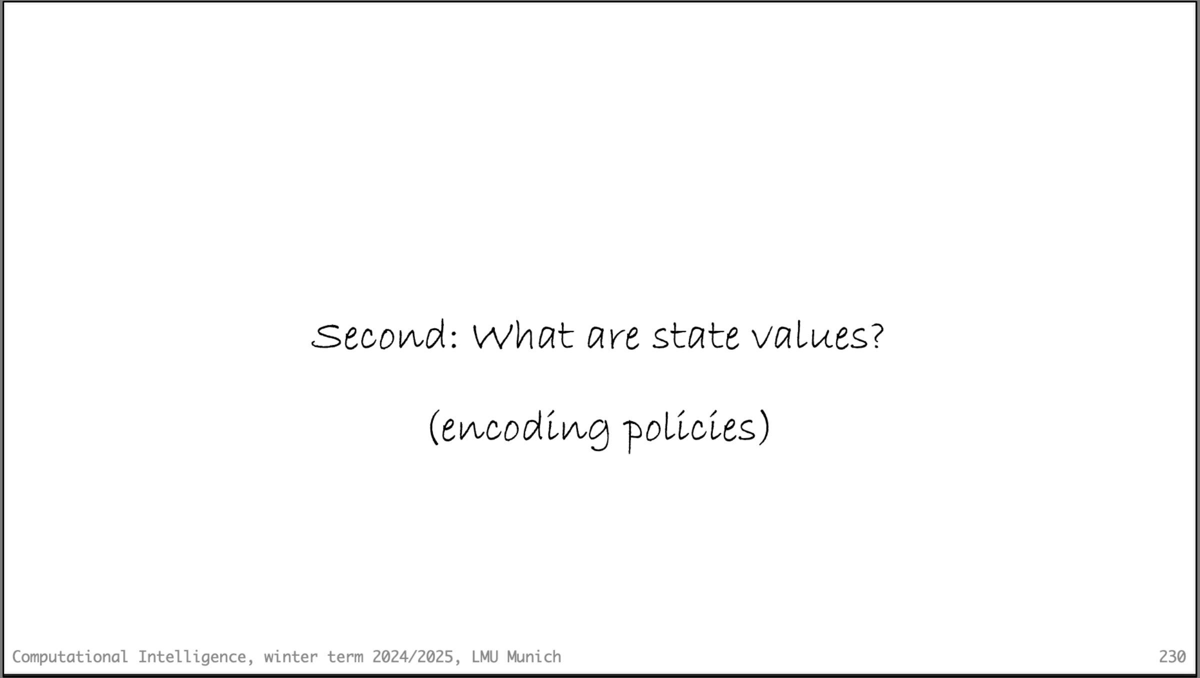
# Computational Intelligence

LMU Munich
winter term 2024/2025

Thomas Gabor
Claudia Linnhoff-Popien



First: What are states?

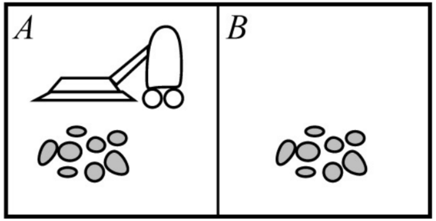


## Second: What are state values?

(encoding policies)

running example #1

## The Vacuum World



reward function: +1 if dirty room becomes clean

$\Rightarrow$ states after we cleaned often are good

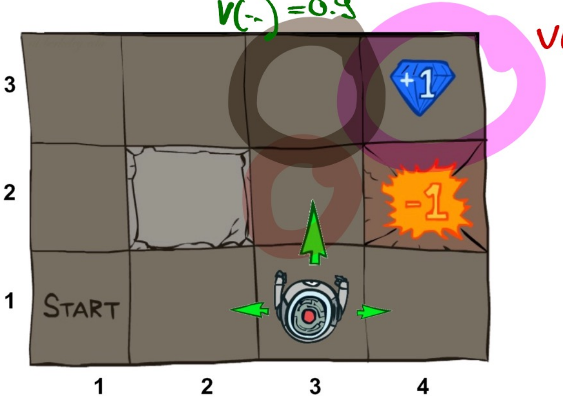
Russel, Norvig. Artificial Intelligence – A Modern Approach. Third Edition. 2016.

running example #2

## The Basic Grid World

$V(\cdot\cdot) = 0.9$

$V(\cdot\cdot) = +1$



reward function : get reward according to grid field

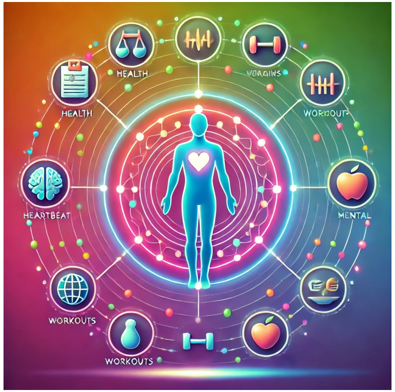
running example #3

## Resource/Stock Trading

chatgpt.com

running example #4

## Personal Life Assistant



chatgpt.com

## Third: Where do we get a state value function?

(finding policies)

**Theorem 2** (Bellman equation)**.** Let $(\mathcal{S}, \mathcal{A}, \mathcal{T}, P, R)$ be a Markov decision process. Let $R : \mathcal{S} \times \mathcal{A} \to \mathcal{T}$ be the expected reward of executing an action in a given state, i.e., $R(s, a) = \mathbb{E}[R(s, a, s')]$ where $s' \sim P(s'|s, a)$. Let $\gamma \in [0; 1) \subseteq \mathbb{R}$ be a temporal discount factor.
The expected reward of a policy $\pi$ being executed starting from state $s$ is given via $\pi$'s value function

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) \cdot V^{\pi}(s').$$

The value function of the optimal policy $\pi^*$ is given via

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left( R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot V^{\pi^*}(s') \right).$$

**Theorem 2** (Bellman equation)**.** Let $(\mathcal{S}, \mathcal{A}, \mathcal{T}, P, R)$ be a Markov decision process. Let $R : \mathcal{S} \times \mathcal{A} \to \mathcal{T}$ be the expected reward of executing an action in a given state, i.e., $R(s, a) = \mathbb{E}[R(s, a, s')]$ where $s' \sim P(s'|s, a)$. Let $\gamma \in [0; 1) \subseteq \mathbb{R}$ be a temporal discount factor.
The expected reward of a policy $\pi$ being executed starting from state $s$ is given via $\pi$'s value function

$$V^\pi(s) = R(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) \cdot V^\pi(s').$$

The value function of the optimal policy $\pi^*$ is given via

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left( R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot V^{\pi^*}(s') \right).$$

**Algorithm 7** (optimal policy)**.** Let $V^* : \mathcal{S} \to \mathcal{T}$ be the *true value function* of a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, P, R)$. The optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ is given via

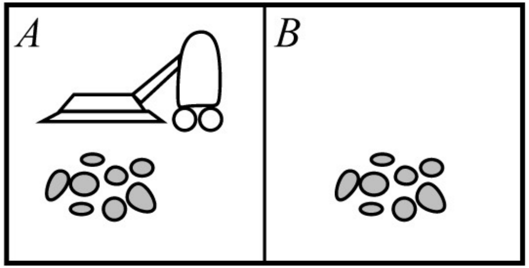
$$\pi^*(s_t) = \arg\max_{a \in \mathcal{A}} V^*(s')$$

where $s' \sim P(s'|s_t, a)$ is the follow-up state when executing action $a$ in state $s_t$.

Let's try that!

running example #1

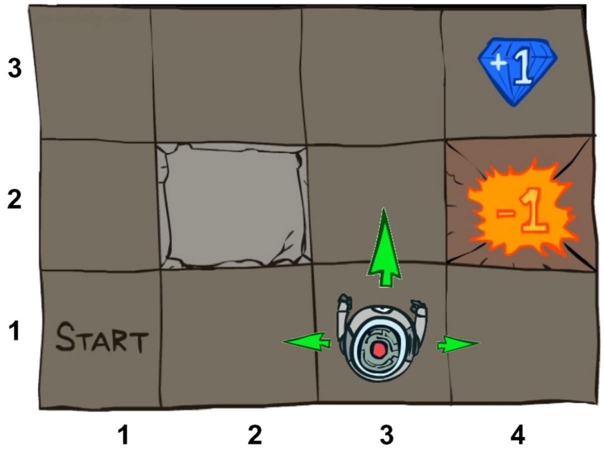
## The Vacuum World



Russel, Norvig. Artificial Intelligence – A Modern Approach. Third Edition. 2016.

running example #2

## The Basic Grid World



https://inst.eecs.berkeley.edu/~cs188/fa22/

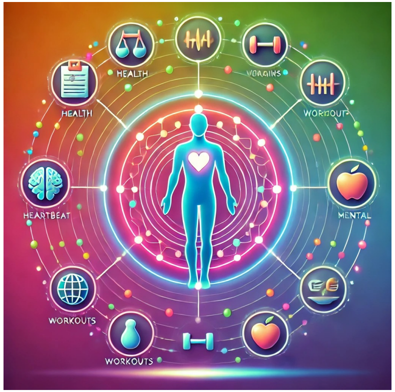
running example #3

## Resource/Stock Trading

chatgpt.com

running example #4

# Personal Life Assistant



chatgpt.com

## Putting it together…

**Definition 11** (training of a neural network)**.** Let $\mathcal{N} : \mathbb{R}^p \to \mathbb{R}^q$ be a neural network with $n$ weights $\overline{\mathcal{N}} = \mathbf{w} \mathbin{+\!\!\!+} \mathbf{b} \in \mathbb{R}^n$ as in Definition 8. Note that thus $|\overline{\mathcal{N}}| = n$. Let $\tau : \mathbb{R}^n \to \mathbb{R}$ be a target function as in Definition 2. Note that thus $\mathcal{T} = \mathbb{R}$. The process of optimizing the network weights $\overline{\mathcal{N}}$ so that $\tau(\overline{\mathcal{N}})$ becomes minimal is called training.

- Let $\mathbb{T} = \{(\mathbf{x}_i, \mathbf{y}_i) \ : \ i = 1, ..., N\}$ be a set of $N$ points of training data, where $\mathbf{x}_i \in \mathbb{R}^p, \mathbf{y}_i \in \mathbb{R}^q$ for all $i$.
  If $\tau$ is of the form
  $$\tau(\overline{\mathcal{N}}) = \sum_{i=1}^{N} (\mathcal{N}(\mathbf{x}_i) - \mathbf{y}_i)^2$$
  or a similar form, the process of training $\mathcal{N}$ is called supervised learning.

**Definition 11** (training of a neural network)**.** Let $\mathcal{N} : \mathbb{R}^p \to \mathbb{R}^q$ be a neural network with $n$ weights $\overline{\mathcal{N}} = \mathbf{w} \mathbin{+\!\!+} \mathbf{b} \in \mathbb{R}^n$ as in Definition 8. Note that thus $|\overline{\mathcal{N}}| = n$. Let $\tau : \mathbb{R}^n \to \mathbb{R}$ be a target function as in Definition 2. Note that thus $\mathcal{T} = \mathbb{R}$. The process of optimizing the network weights $\overline{\mathcal{N}}$ so that $\tau(\overline{\mathcal{N}})$ becomes minimal is called training.

- Let $\mathbb{T} = \{(\mathbf{x}_i, \mathbf{y}_i) \;:\; i = 1, ..., N\}$ be a set of $N$ points of training data, where $\mathbf{x}_i \in \mathbb{R}^p, \mathbf{y}_i \in \mathbb{R}^q$ for all $i$.
  If $\tau$ is of the form
  $$\tau(\overline{\mathcal{N}}) = \sum_{i=1}^{N} (\mathcal{N}(\mathbf{x}_i) - \mathbf{y}_i)^2$$
  or a similar form, the process of training $\mathcal{N}$ is called supervised learning.

- Let $(\mathcal{O}, \mathcal{A}, \mathcal{T}, e, R)$ be a decision process (cf. Definition 9) for which policy $\pi_{\overline{\mathcal{N}}} : \mathcal{O} \to \mathcal{A}$ yields (possibly randomized or non-deterministic) rewards $\langle r_t \rangle_{t \in \mathcal{Z}}$. Note that $\pi_{\overline{\mathcal{N}}}$ in some way calls $\mathcal{N}$ to produce its output, for example
  $$\pi_{\overline{\mathcal{N}}}(o) = \mathcal{N}(o)$$
  for $\mathcal{O} \subseteq \mathbb{R}^p, \mathcal{A} \subseteq \mathbb{R}^q$ or if suitable translations exist.
  If $\tau$ is of the form
  $$\tau(\overline{\mathcal{N}}) = -\mathbb{E}\Big[\sum_{t \in \mathcal{Z}} \gamma^t \cdot r_t\Big]$$
  or a similar form, the process of training $\mathcal{N}$ is called policy-based reinforcement learning.

- Let $(\mathcal{S}, \mathcal{A}, \mathcal{T}, P, R)$ be a Markov decision process (cf. Definition 10) for which we run policy $\pi_{\overline{\mathcal{N}}} : \mathcal{S} \to \mathcal{A}$. Note that $\pi_{\overline{\mathcal{N}}}$ in some way calls $\mathcal{N}$ to produce its output, for example

  $$\pi_{\overline{\mathcal{N}}}(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim P(s'|s,a)} \left[ \mathcal{N}(s') \right]$$

  for $\mathcal{S} \times \mathcal{A} \subseteq \mathbb{R}^p$ with $q = 1$ or if suitable translations exist.
  Let $R : \mathcal{S} \times \mathcal{A} \to \mathcal{T}$ be the expected reward of executing an action in a given state, i.e., $R(s,a) = \mathbb{E}[R(s,a,s')]$ where $s' \sim P(s'|s,a)$. Let $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ be a (possibly randomized or non-deterministic) transition function, i.e., $T(s,a) = s'$ where $s' \sim P(s'|s,a)$. Let $\gamma \in [0;1]$ be a discount factor. Let $V_{\pi_{\overline{\mathcal{N}}}} : \mathcal{S} \to \mathbb{R}$ be the total discounted reward that policy $\pi_{\overline{\mathcal{N}}}$ generates when starting in state $s$, i.e.,

  $$V_{\pi_{\overline{\mathcal{N}}}}(s) = R\big(s, \pi_{\overline{\mathcal{N}}}(s)\big) + \gamma \cdot V_{\pi_{\overline{\mathcal{N}}}}\big(T(s, \pi_{\overline{\mathcal{N}}}(s))\big).$$

  Note that for $\gamma < 1$ we can abort this recursive computation once the effect of the further recursive part is sufficiently small. Note that we may also have a fixed recursion depth or that $T(s^\dagger, \_)$ might not be defined for all $s^\dagger \in \mathcal{S}$, which are then called terminal states and also cause the recursion to end.
  Let $\mathbb{S} = \{\mathbf{s}_i \, : \, i = 1, ..., N\} \subseteq S$ be a set of training states. If $\tau$ is of the form

  $$\tau(\overline{\mathcal{N}}) = -\frac{1}{N} \cdot \sum_{i=1}^{N} V_{\pi_{\overline{\mathcal{N}}}}(\mathbf{s}_i)$$

  or a similar form, the process of training $\mathcal{N}$ is called value-based reinforcement learning.

# Reinforcement Learning

# Variations of Value Functions

| name | network | policy |
|---|---|---|
| policy-based | $\mathcal{N} : \mathcal{S} \to \mathcal{A}$ | $\pi_{\overline{\mathcal{N}}}(s) = \mathcal{N}(s)$ |
| value-based $(V)$ | $\mathcal{N} : \mathcal{S} \to \mathbb{R}$ | $\pi_{\overline{\mathcal{N}}}(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim P(s' \mid s,a)} [\mathcal{N}(s')]$ |
| value-based $(Q)$ | $\mathcal{N} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ | $\pi_{\overline{\mathcal{N}}}(s) = \arg\max_{a \in \mathcal{A}} \mathcal{N}(s, a)$ |

...

## On-Policy vs. Off-Policy Learning

## The Goal Class Hierarchy

Goal Class 5: State Values

Goal Class 4: Rewards and Costs

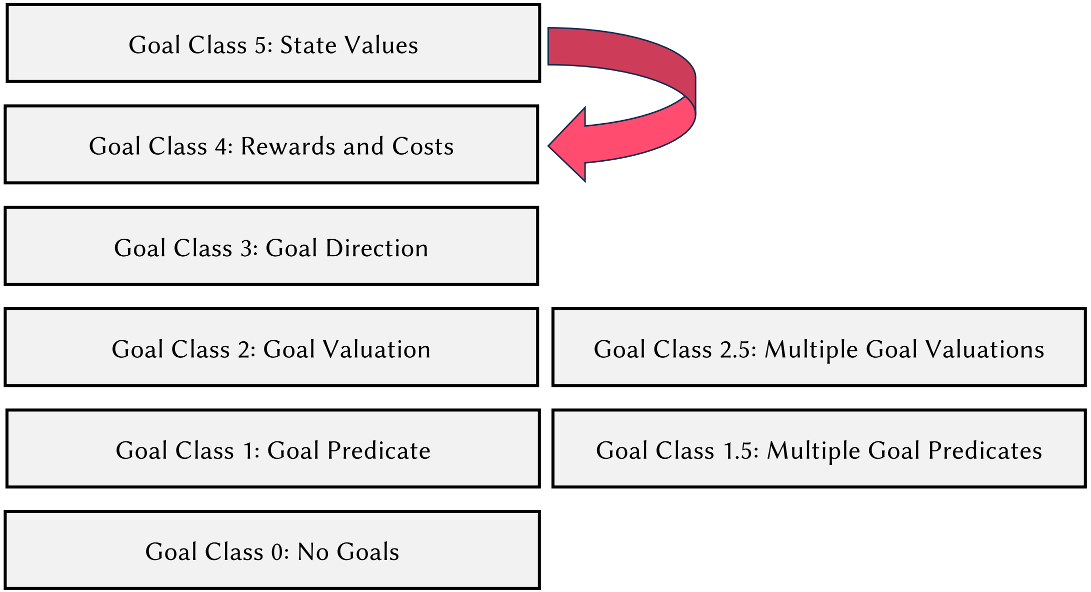
Goal Class 3: Goal Direction

Goal Class 2: Goal Valuation | Goal Class 2.5: Multiple Goal Valuations

Goal Class 1: Goal Predicate | Goal Class 1.5: Multiple Goal Predicates

Goal Class 0: No Goals

## The Goal Class Hierarchy — taking a step back


Goal Class 5: State Values
Goal Class 4: Rewards and Costs
Goal Class 3: Goal Direction
Goal Class 2: Goal Valuation
Goal Class 2.5: Multiple Goal Valuations
Goal Class 1: Goal Predicate
Goal Class 1.5: Multiple Goal Predicates
Goal Class 0: No Goals

# Partial Observability

## POMDPs

# The Goal Class Hierarchy

| | |
|---|---|
| Goal Class 5: State Values | |
| Goal Class 4: Rewards and Costs | |
| Goal Class 3: Goal Direction | |
| Goal Class 2: Goal Valuation | Goal Class 2.5: Multiple Goal Valuations |
| Goal Class 1: Goal Predicate | Goal Class 1.5: Multiple Goal Predicates |
| Goal Class 0: No Goals | |

https://stablediffusionweb.com

# Multi-Agent Applications

https://stablediffusionweb.com/prompts

https://aaai.org/Conferences/AAAI-19/invited-speakers/

**Speaker:** Cynthia Breazeal

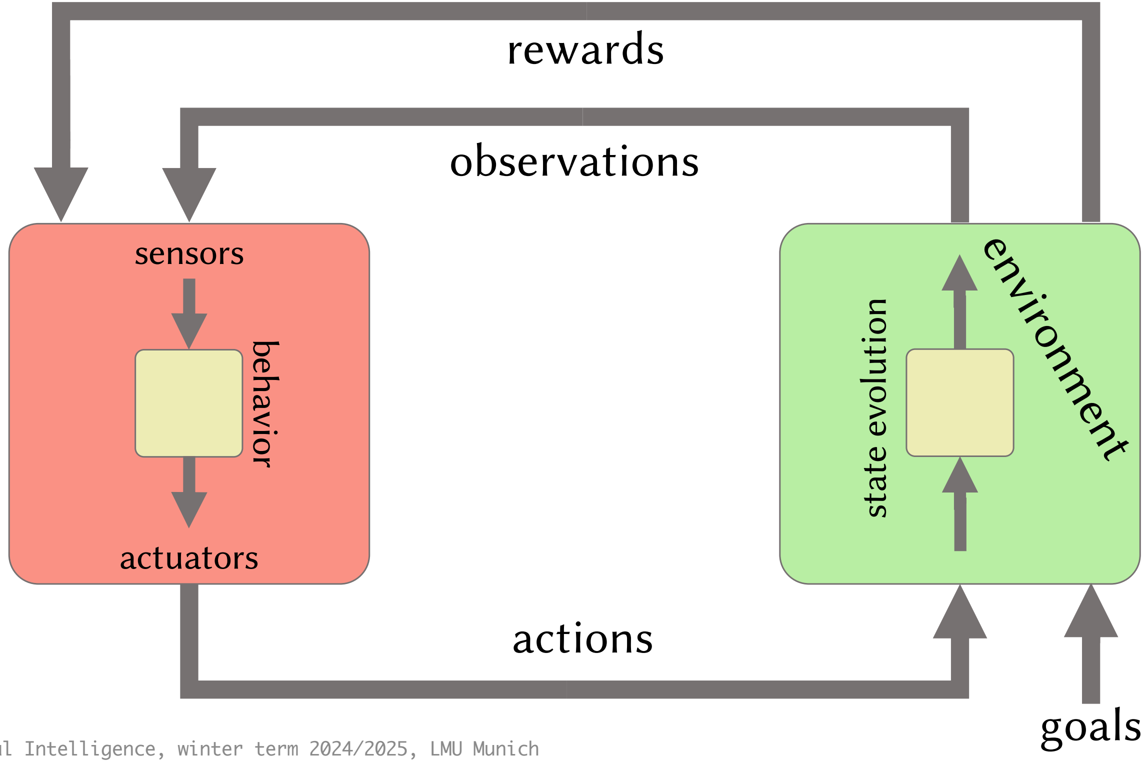
**Title:** „Living and Flourishing with AI"

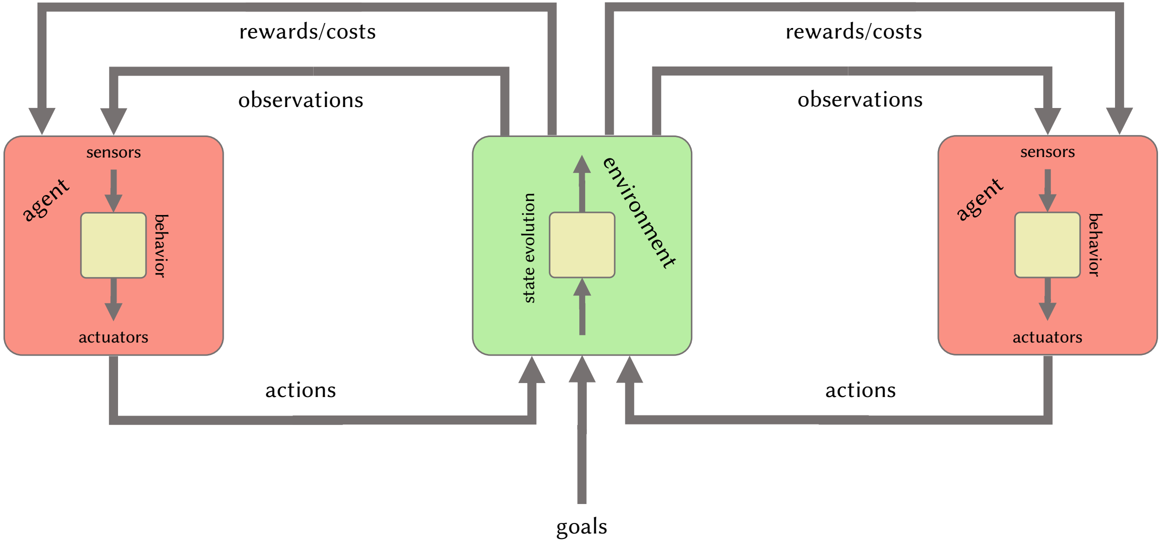
https://aws.amazon.com/de/solutions/case-studies/amazon-robotics-case-study/

| | single-agent programming | multi-agent programming |
|---|---|---|
| single-agent goals | single agent system (or ignorant agents?) | coordination (game theory) |
| multi-agent goals | emergent behavior (swarms etc) | agent groups, societies, institutions |

# Multi-Agent Systems

Why?


rewards
observations
sensors
behavior
actuators
state evolution
environment
actions
goals


rewards/costs
observations
sensors
agent
behavior
actuators
actions
state evolution
environment
rewards/costs
observations
sensors
agent
behavior
actuators
actions
goals

**Definition 12** (multi-agent system)**.** Let $G = \{G^{[1]}, ..., G^{[N]}\}$ be a set of $|G| = N$ agents with observation spaces $\mathcal{O}^{[i]}$ and action spaces $\mathcal{A}^{[i]}$ controlled by policies $\pi^{[i]}$ for all $i = 1, ..., N$, respectively. The multi-agent system $G$ then takes a joint action $a \in \mathcal{A}$ with $\mathcal{A} = \mathcal{A}^{[1]} \times ... \times \mathcal{A}^{[N]}$ after making a joint observation $o \in \mathcal{O}$ with $\mathcal{O} = \mathcal{O}^{[1]} \times ... \times \mathcal{O}^{[N]}$ based on the joint policy $\pi(o^{[1]}, ..., o^{[N]}) = (a^{[1]}, ..., a^{[N]})$ where $a^{[i]} = \pi^{[i]}(o^{[i]})$ for all $i$.

## Multi-Agent Systems as a Paradigm for Distributed Programming

# Multi-Agent Systems as a Paradigm for Distributed Programming

$$\begin{aligned} \mathit{GreetingRobot}(\_) &= \mathit{good\_morning}.\overline{\mathit{morning}}.\mathit{GreetingRobot}(\text{``happy''}) \\ &\quad + \mathit{morning}.\mathit{GreetingRobot}(\text{``happy''}) \end{aligned}$$

$$\mathit{HappyMultiAgentSystem} = \mathit{GreetingRobot}(\text{``sad''}) \;\mid\; \overline{\mathit{good\_morning}}.\mathit{GreetingRobot}(\text{``angry''})$$

# Swarm Intelligence

local programming => emergent behavior