Computational Intelligence 2023/24

Exam — February 12th, 2024

Please mind the following:

- Fill in your **personal information** in the fields below.
- You may use an unmarked dictionary during the exam. **No** additional tools (like calculators, e.g.) might be used.
- Fill in all your answers **directly into this exam sheet**. You may use the backside of the individual sheets or ask for additional paper. In any case, make sure to mark **clearly** which question you are answering. Do not use pens with green or red color or pencils for writing your answers. You may give your answers in both **German and English**.
- Points annotated for the individual tasks only serve as a preliminary guideline.
- At the end of the exam hand in your **whole exam sheet**. Please make sure you do not undo the binder clip!
- To forfeit your exam ("entwerten"), please cross out clearly this **cover page** as well as **all pages** of the exam sheet. This way the exam will not be evaluated and will not be counted as an exam attempt.
- Please place your LMU-Card or student ID as well as photo identification ("Lichtbildausweis") clearly visible on the table next to you. Note that we need to check your data with the data you enter on this cover sheet.

Time Limit: 90 minutes

First Name:	
Last Name:	
Matriculation Number:	

Topic 1	max. 10 pts.	pts.
Topic 2	max. 18 pts.	pts.
Topic 3	max. 10 pts.	pts.
Topic 4	max. 14 pts.	pts.
Topic 5	max. 10 pts.	pts.
Topic 6	max. 10 pts.	pts.
Topic 7	max. 18 pts.	pts.
Total max. 90 pts.		pts.

1 General Knowledge / Single Choice **10pts**

For each of the following questions select **one** correct answer ('1 of n'). Every correct answer is awarded one point. Multiple answers or incorrect answers will be marked with zero points.

(a) If you train a large language model to imitate how you text with people and then ask your fellow students to discern if they are writing with you or with the large language model, you are effectively running an instance of...

i the Turing Test	ii Prolog	iii the Singularity	iv FunSearch

(b) In accordance with the main focus of the programming language Prolog, its name is derived from ...

i producing log files	ii programming logic	iii probability logarithms	iv pranking old ogres

(c) Given a fixed graph structure, a neural network is fully defined by three categories of entities. Which is not one of them?

i weights	ii biases	iii curvatures	iv activation functions

(d) In a standard evolutionary algorithm, *variation* operators are usually employed to increase which property of the optimization?

i selection	ii exploration	iii exploitation	iv production

(e) Compared to the game of Go, which was only considered artificially 'mastered' as of 2016, the game of Chess was solvable much earlier. The software that was able to beat then world champion Garri Kasparow in the late 1990s was called ...

i Deep Blue	ii AlphaGo	iii DeepMind	iv ChatGPT

(f) A multi-agent system made up of many similar agents that have a shared interest but little communication and only locally executed algorithms is called ...

i market	ii evolutionary stable	iii goal class	iv swarm

(g) One of the main difficulties for finding effective strategies in the game *Lemonade Stand* (sometimes also called *Glühwein Stand*) lies in the opponent strategies' ...

i action space	ii non-stationarity	iii beverage quality	iv policy model

(h) In a multi-iteration tournament of the game *Prisoner's Dilemma* where you play against each opponent for an unknown number of games, you are allowed to adopt any of these four strategies. Which one of these is expected to be the most successful over the course of such a tournament?

i always *Cooperate*	ii always *Defect*	iii give up	iv tit-for-tat

(i) The *Hawk–Dove* game was modeled as a possible explanation for the ecological balance of ...

i aggression between peers	ii differences in size	iii animal intelligence	iv CO2 emissions

(j) The outlook of machine learning expert Richard Sutton on the eventual superiority of using search-based algorithms instead of human-experience-based knowledge was summarized in his famous 2019 publication titled ...

i "The Bitter Lesson"	ii "The Bitter Lemon"	iii "The Barren Line"	iv "The Better LISP"

2 Agents and Goals **18pts**

Scenario 1. We consider a multi-agent system made up of $n = 20$ cows and a mowing robot called *MowBot*. All these 21 agents live on an infinite meadow which we model as an infinite 2d plane of positions $l \in \mathbb{R}^2$. The agent $G^{[0]}$ is the *MowBot* and all other agents $G^{[1]}, ..., G^{[20]}$ are cows.
The evolution of the multi-agent system is encoded as a series of states $\langle s_t \rangle_t$ of unspecified length for states $s_t \in \mathcal{S}$ at time t, where $\mathcal{S} = \mathbb{N} \times (\mathbb{R}^2)^{21}$ is the (in total 43-dimensional) state space, as well as series of joint observations $\langle o_t \rangle_t = \langle (o_t^{[0]}, ..., o_t^{[20]}) \rangle_t$ made by the agents and joint actions $\langle a_t \rangle_t = \langle (a_t^{[0]}, ..., a_t^{[20]}) \rangle_t$ executed by the agents.
Our system is fully observable. Thus, each agent $G^{[i]}$ makes observations

$$o_t^{[i]} = s_t = (t, l_t^{[0]}, ..., l_t^{[20]}) \in \mathcal{S}$$

at each time step t where $l_t^{[i]} \in \mathbb{R}^2$ is the position of the agent $G^{[i]}$ at time $t \in \mathbb{N}$. Furthermore, any agent $G^{[i]}$ uses the identical action space $\mathcal{A}^{[i]} = \{$`do_nothing`, `go_northeast`, `go_east`, `go_southeast`, `go_south`, `go_southwest`, `go_west`, `go_northwest`, `go_north`, `eat/mow`$\}$ with $|\mathcal{A}^{[i]}| = 10$ where `do_nothing` has the agent not do anything, all actions starting with `go_` cause the agent to move for a certain length in the respective direction, and `eat/mow` causes any cow agent to eat the grass and any *MowBot* agent to mow the grass.

You can use a 2d geometric distance function $dist(l, l') = \sqrt{(x - x')^2 + (y - y')^2}$ where $l = (x, y) \in \mathbb{R}^2$ and $l' = (x', y') \in \mathbb{R}^2$.

(i) Give a goal predicate $\gamma_1 : \langle \mathcal{S} \rangle \times \langle \mathcal{A} \rangle \to \mathbb{B}$ so that γ_1 holds iff the *MowBot* has never (at full time steps) been closer than a distance of 1 to any of the cows. (4pts)

$$\gamma_1\Big(\langle (t, l_t^{[0]}, ..., l_t^{[20]}) \rangle_t, \langle (a_t^{[0]}, ..., a_t^{[20]}) \rangle_t\Big) \iff$$

(ii) Let $(\mathcal{S}, \mathcal{A}, \mathbb{R}, P, R)$ be a Markov decision process (MDP, cf. Definition 3 in the appendix) with state space $\mathcal{S}$ from Scenario 1, joint action space $\mathcal{A}$ from Scenario 1, target space $\mathcal{T} = \mathbb{R}$, and a given transition probability function P. Give — from our *MowBot*'s perspective — a reward function R that both incentivizes to (a) fulfill γ_1 and (b) execute `mow/eat` as often as possible. (8pts)

(iii) Now, in any run at time step $t = 26$ all cows suddenly change their policy to a specific new (from then on again fixed) policy.

Can R still be learned as the reward function of an MDP? Briefly state your reasoning. (3pts)

(iv) Let $\gamma_2 : \langle \mathcal{S} \rangle \times \langle \mathcal{A} \rangle \to \mathbb{B}$ be goal predicate so that γ_2 holds iff the *MowBot* has never executed the action `eat/mow` within a radius of 1 from any point where any cow has ever executed `mow/eat`. Assume that the cows' policies are fixed.

Can the *MowBot* learn to fulfill γ_2 effectively for an MDP with the given state space $\mathcal{S}$ for some reward function R'? Briefly state your reasoning. (3pts)

3 Fuzzy Logic **10pts**

To evaluate the impact of the *MowBot* in Scenario 1 on the lives of our cows, we performed a field study and measured a cow's happiness at various positions $(x, y) \in \mathbb{R}^2$ on the infinite meadow. For our experiments, we firmly placed the *MowBot* at position $(3, -1)$ and never let it change that position.

From that study, we found that a cow's happiness can be described by the following formula in fuzzy logic:

$$\textit{happy}((x,y)) = \texttt{AND}\big(\textit{green}((x,y)), \texttt{NOT}\big(\textit{mowbot\_nearby}((x,y))\big)\big)$$

$$\text{where } \textit{green}((x,y)) = \begin{cases} 1.0 & \text{if } |x| < 3 \text{ and } |y| < 3, \\ 0.8 & \text{otherwise,} \end{cases}$$

$$\text{and } \textit{mowbot\_nearby}((x,y)) = 0.5 \cdot \max\big\{2 - \textit{dist}((x,y),(3,-1)),\ 0\big\}.$$

(i) What is the happiness of a cow at position $(3, 0)$? Show your computation steps. (4pts)

(ii) We now consider that cows can walk one step in any direction by adding one of the eight vectors $\{(+1,+1),(+1,0),(+1,-1),(0,+1),(0,-1),(-1,+1),(-1,0),(-1,-1)\}$ to their position. What is the <u>most drastic increase</u> in happiness a cow can achieve with one step? What is one possible corresponding start position and one possible corresponding end position of such a drastic step? (6pts)

4 Optimization **14pts**

Scenario 2. The policy $\pi : \mathcal{S} \to \mathcal{A}^{[\_]}$ of a *math cow* is given by a real number $0 \leq u < 1$, i.e.,

$$\pi((t, \_, ..., \_)) = map(\lfloor 10^t \cdot u \mod 10 \rfloor)$$

where state space $\mathcal{S}$ and action space $\mathcal{A}$ are given as in Scenario 1, $\lfloor w \rfloor$ for $w \in \mathbb{R}$ is the nearest integer from w rounded down, $w \mod 10$ for positive real numbers $w \in \mathbb{R}$ is given recursively via

$$w \mod 10 = \begin{cases} w & \text{if } w < 10, \\ w - 10 \mod 10 & \text{otherwise,} \end{cases}$$

and $map : \{0, ..., 9\} \to \mathcal{A}^{[\_]}$ is given exhaustively via

$$\begin{aligned} map(0) &= \texttt{do\_nothing}, \\ map(1) &= \texttt{go\_northeast}, \\ &... \\ map(8) &= \texttt{go\_north}, \\ map(9) &= \texttt{eat/mow}. \end{aligned}$$

In short, the *math cow* policy encoding described in Scenario 2 uses a single real number u to encode a (potentially very long or even infinitely long) behavior by interpreting every single decimal digit of u as a subsequent action to be taken by a corresponding policy. We now give two examples if you need them:

- If a *math cow*'s policy π is given via the number $u = 0.199$, then that cow's action at time point $t = 1$ is given via

$$\begin{aligned} \pi((1, \_, ..., \_)) &= map(\lfloor 10^1 \cdot 0.199 \mod 10 \rfloor) \\ &= map(\lfloor 1.99 \mod 10 \rfloor) \\ &= map(\lfloor 1.99 \rfloor) \\ &= map(1) \\ &= \texttt{go\_northeast}. \end{aligned}$$

  Analogously, $\pi((2, \_, ..., \_)) = \texttt{eat/mow}$, then $\pi((3, \_, ..., \_)) = \texttt{eat/mow}$, and then $\pi((t, \_, ..., \_)) = \texttt{do\_nothing}$ for all $t \geq 4$.

- If a *math cow*'s policy π is given via the number $u = \frac{8}{9} = 0.\overline{8} = 0.888...$, then that cow's action at time point $t = 1$ is given via

$$\begin{aligned} \pi((1, \_, ..., \_)) &= map(\lfloor 10^1 \cdot 0.\overline{8} \mod 10 \rfloor) && (1) \\ &= \texttt{go\_north}. && (2) \end{aligned}$$

  This action is then also chosen for any other time point $t \geq 1$.

We now want to optimize one cow's policy for any given target function τ via *simoolated annealing*, a special form of simulated annealing (cf. Algorithm 1 in the appendix) with a specific acceptance probability function

$$A(Q, Q', K) = \frac{1}{K}$$

for temperature schedule

$$\kappa(t) = t.$$

(i) What is the effect of this definition for A and κ in *simoolated annealing* compared to more typical or more general variants of simulated annealing? How does this effect vary with the quality of the newly generated solution candidate? (4pts)

(ii) A friendly *neighbors* function, through which we generate from a policy π a new policy $\pi' \sim \mathit{neighbors}(\pi)$, has the following properties:

1. On average, substantial parts of the information in π remain within π'.
2. π' always differs in at least one action within the first 1000 actions from π.
3. *neighbors* is symmetric in the sense that generating π' from π has the same chance as generating π from π' would have.
4. For a given π, the value of π' is non-deterministic.

Give a friendly *neighbors* function. Give a brief explanation why your function fulfills each of the four properties enumerated above. (10pts)

Hint: You can use policies π, π' and their corresponding real numbers u, u' interchangeably, without any need to type cast from π to u or vice versa.

Note: You can write $x \sim X$ to sample a (uniformly) random element x from a finite set X. You can also write $x = \texttt{random}()$ to sample a random real number x with $x \in [0; 1] \subset \mathbb{R}$.

5 The π-Calculus **10pts**

(i) Evaluate the following π-process step by step until it cannot be reduced any further. (6pts)

$!(\overline{panic}.beep.\mathbf{0}+\overline{beep}.\text{MowBot}+panic.beep.\text{Cow}+\overline{brr}.moo.\text{Cow}) \mid \overline{moo}.\text{Cow} \mid \overline{panic}.moo.\mathbf{0}$

(ii) Evaluate the following π-process step by step until it cannot be reduced any further. (4pts)

$!(\overline{beep}.\text{MowBot}) \mid \overline{beep}.beep.(panic.\overline{moo}.beep.\mathbf{0} + beep.panic.\text{MowBot}) \mid beep.\mathbf{0} \mid \overline{beep}.\mathbf{0}$

6 Game Theory **10pts**

Since cows (and their farmer) can be terribly dangerous animals, the *MowBot* has to think twice how it should behave around the meadow, depending on which behavior it encounters. Within the two-player normal-form game for our true-to-life scenario, we can observe these dynamics unfold in the following payoff matrix for the *Cowmunication Game*:

	Cool Cow	Shy Cow	Default Cow	Brave Cow	Farmer
'moo?'	$2,2$	$3,0$	$4,0$	$4,2$	$2,1$
'brrrr'	$4,1$	$3,1$	$3,0$	$4,4$	$2,4$
stop	$1,0$	$0,1$	$3,4$	$4,1$	$2,4$
ignore	$3,0$	$0,3$	$1,3$	$0,0$	$4,4$

(i) Compute <u>all</u> pure Nash equilibria (cf. Definitions 6 and 9 in the appendix) for the *Cowmunication Game* by denoting all best responses. (6pts)

(ii) Give <u>all</u> strategies on the Pareto front (cf. Definition 7 in the appendix) for the *Cowmunication Game*. Give your reasoning! (4pts)

7 Scientific Reading: Model-Based Reinforcement Learning

18pts

After some tweaking, a large language model generated the following (correct) explanation of the established technique *model-based reinforcement learning*, which we did not discuss in the lecture. Read this explanation and answer the questions afterwards!

In the context of model-based reinforcement learning (MBRL), a "model" refers to an abstract representation or approximation of the environment in which the agent operates. This model captures the dynamics of the environment, including how states transition to other states and the rewards associated with these transitions, based on actions taken by the agent.

Model-based reinforcement learning is a sophisticated approach in the field of artificial intelligence, where the system explicitly constructs this model of the environment. Unlike model-free methods, which learn solely from trial-and-error interactions with the environment, MBRL leverages its internal model to simulate outcomes, enabling it to anticipate future states and make more informed decisions. The internal model can be either learned from interactions with the real environment or provided *a priori*.

In MBRL, the model interacts with other components of the learning process in a pivotal way. It serves as a predictive engine that forecasts the next state and potential rewards for given actions. The agent uses this model for planning by simulating actions in a "mental rehearsal" before executing them in the real environment. This ability to predict and evaluate potential future scenarios allows the agent to optimize its policy with fewer real-world interactions, which is particularly valuable in scenarios where experimentation is costly or dangerous.

The core components of an MBRL system include the model of the environment, the policy that dictates the agent's behavior, and the value function, which estimates the expected return from states or state-action pairs. The environment model is central, providing a bridge between the policy and value function by predicting the outcomes of actions. Learning can occur in two key areas: improving the environment model to better predict future states and rewards, and optimizing the policy and value function based on predictions from the model.

The distinction between model-based RL and simpler value or Q-value approaches (model-free methods) lies in their strategies for improving the reward. Model-free methods learn a direct mapping from states or state-action pairs to values or Q-values, respectively, relying heavily on extensive interaction with the environment to iteratively improve their estimates. They excel in environments where the model is hard to learn, but they often require a large amount of experience to converge to an optimal policy. On the other hand, MBRL seeks to understand the dynamics of the environment through its model, which allows it to anticipate and plan for future states. This foresight enables MBRL to potentially achieve higher rewards with fewer interactions by using its model to explore outcomes of actions without physically executing them, making it more efficient in environments where building a reasonably accurate model is feasible.

(i) In Scenario 1, a cow agent $G^{[7]}$ might want to mirror all actions performed by the much cooler cow agent $G^{[6]}$ except that $G^{[7]}$ wants to execute `eat/mow` whenever $G^{[6]}$ executes `do_nothing` and vice versa. Give a policy $\pi^{[7]}$ for $G^{[7]}$ that uses a *model* $\mu^{[6]}$ of $G^{[6]}$'s behavior. Explain how the usage of a *model* matters in your example. (12pts)

(ii) In Scenario 2, imagine an "irrational math cow" agent whose policy is given by the irrational number $u = \frac{\pi}{10} = 0.314...$, i.e., π (the policy) is given by $\frac{\pi}{10}$ (the number) . Recall that, for any irrational number, there is no repeating pattern in its decimal digits. Will model-based reinforcement learning be helpful for the other cows to model the behavior of the "irrational math cow" agent? Explain your reasoning. (6pts)

Appendix: Definitions

Notation. $\mathbb{B}$ is the set of truth values or Booleans, i.e., $\mathbb{B} = \{0, 1\}$. $\mathbb{N}$ is the set of natural numbers starting from zero, i.e., $\mathbb{N} = \{0, 1, 2, ...\}$ so that it holds that $\mathbb{B} \subset \mathbb{N} \subset \mathbb{Z} \subset \mathbb{R} \subset \mathbb{C}$. $\mathbb{P}$ denotes the space of probabilities and $\mathbb{F}$ denotes the space of fuzzy values with $\mathbb{P} = \mathbb{F} = [0; 1] \subset \mathbb{R}$ being only discerned for semantic but not mathematical reasons. $\wp(X)$ denotes the power set of X. $\mathbb{E}$ denotes the expected value. $+\!\!+$ denotes vector or sequence concatenation, i.e., given two vectors $\mathbf{x} = \langle x_1, ..., x_{|\mathbf{x}|} \rangle$ and $\mathbf{y} = \langle y_1, ..., y_{|\mathbf{y}|} \rangle$, $\mathbf{x} +\!\!+ \mathbf{y} = \langle x_1, ..., x_{|\mathbf{x}|}, y_1, ..., y_{|\mathbf{y}|} \rangle$. A vector $\langle x_0, ..., x_{n-1} \rangle$ with length $n \in \mathbb{N}$ can also be written as $\langle x_i \rangle_{0 \leq i \leq n-1}$ for a new iteration variable i. _ denotes unspecified function arguments ($f(\_) = 0$ is the constant function that always returns zero, e.g.) or other variables that are not used. For any finite set $X = \{x_0, ..., x_n\}$, $|X| = n$ denotes the number of elements in X. For infinite sets, $|X| = \infty$.

Definition 1 (agent)**.** Let $\mathcal{A}$ be a set of actions. Let $\mathcal{O}$ be a set of observations. An agent A can be given via a policy function $\pi : \mathcal{O} \to \mathcal{A}$. Given a time series of observations $\langle o_t \rangle_{t \in \mathcal{Z}}$ for some time space $\mathcal{Z}$ the agent can thus generate a time series of actions $\langle a_t \rangle_{t \in \mathcal{Z}}$ by applying $a_t = \pi(o_t)$.

Definition 2 (optimization)**.** Let $\mathcal{X}$ be an arbitrary state space. Let $\mathcal{T}$ be an arbitrary set called target space and let $\leq$ be a total order on $\mathcal{T}$. A total function $\tau : \mathcal{X} \to \mathcal{T}$ is called target function. Optimization (minimization/maximization) is the procedure of searching for an $x \in \mathcal{X}$ so that $\tau(x)$ is optimal (minimal/maximal). Unless stated otherwise, we assume minimization.
An optimization run of length $g+1$ is a sequence of states $\langle x_t \rangle_{0 \leq t \leq g}$ with $x_t \in \mathcal{X}$ for all t.

Let $e : \langle \mathcal{X} \rangle \times (\mathcal{X} \to \mathcal{T}) \to \mathcal{X}$ be a possibly randomized or non-deterministic function so that the optimization run $\langle x_t \rangle_{0 \leq t \leq g}$ is produced by calling e repeatedly, i.e., $x_{t+1} = e(\langle x_u \rangle_{0 \leq u \leq t}, \tau)$ for all t, $1 \leq t \leq g$, where x_0 is given externally (e.g., $x_0 =_{def} 42$) or chosen randomly (e.g., $x_0 \sim \mathcal{X}$). An optimization process is a tuple $(\mathcal{X}, \mathcal{T}, \tau, e, \langle x_t \rangle_{0 \leq t \leq g})$.

Algorithm 1 (simulated annealing)**.** Let $\mathcal{D} = (\mathcal{X}, \mathcal{T}, \tau, e, \langle x_u \rangle_{0 \leq u \leq t})$ be an optimization process. Let *neighbors* : $\mathcal{X} \to \wp(\mathcal{X})$ be a function that returns a set of neighbors of a given state $x \in \mathcal{X}$. Let $\kappa : \mathbb{N} \to \mathbb{R}$ be a temperature schedule, i.e., a function that returns a temperature value for each time step. Let $A : \mathcal{T} \times \mathcal{T} \times \mathbb{R} \to \mathbb{P}$ with $\mathbb{P} = [0; 1] \subset \mathbb{R}$ be a function that returns an acceptance probability given two target values and a temperature. Commonly, we use

$$A(Q, Q', K) = \mathtt{e}^{\frac{-(Q'-Q)}{K}} \text{ with Euler's number } \mathtt{e}$$

for $\mathcal{T} \subseteq \mathbb{R}$. The process $\mathcal{D}$ continues via simulated annealing if e is of the form

$$e(\langle x_u \rangle_{0 \leq u \leq t}, \tau) = x_{t+1} = \begin{cases} x'_t & \text{if } \tau(x'_t) \leq \tau(x_t) \text{ or } r \leq A\big(\tau(x_t), \tau(x'_t), \kappa(t)\big), \\ x_t & \text{otherwise,} \end{cases}$$

where $x'_t \sim \mathit{neighbors}(x_t)$ and $r \sim \mathbb{P}$ are drawn at random for each call to e.

Definition 3 (Markov decision process (MDP))**.** Let $\mathcal{A}$ be a set of actions. Let $\mathcal{S}$ be a set of states. Let A be an agent given via a policy function $\pi : \mathcal{S} \to \mathcal{A}$. Let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{T}$ be a possibly randomized *cost* (*reward*) function. A Markov decision process is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, P, R)$ where $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{P}$ is the *transition probability function* often written as $P(s'|s, a) = \Pr(s_{t+1} = s' \,|\, s_t = s \wedge a_t = a)$.
A Markov decision process run for policy π is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \tau, \pi, \langle s_t \rangle_{t \in \mathcal{Z}}, \langle a_t \rangle_{t \in \mathcal{Z}})$ where

$$s_{t+1} \sim P(s_{t+1}|s_t, a_t)$$

and where τ is to be minimized (maximized) and usually has a form similar to

$$\textit{accumulated cost (reward)}\ \tau(\pi) =_{\mathit{def}} \sum_{t \in \mathcal{Z}} R(s_t, a_t, s_{t+1})$$
$$\text{or } \textit{discounted expected cost (reward)}\ \tau(\pi) =_{\mathit{def}} \mathbb{E}\Big[\sum_{t \in \mathcal{Z}} \gamma^t \cdot R(s_t, a_t, s_{t+1})\Big]$$

where $\gamma \in [0; 1] \subset \mathbb{R}$ is called a discount factor.
A decision process generates a (possibly infinite) series of rewards $\langle r_t \rangle_{t \in \mathcal{Z}}$ with $r_t = R(s_t, a_t, s_{t+1})$.

Definition 4 (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.

Definition 5 (normal-form game)**.** Let $G = \{G^{[1]}, ..., G^{[N]}\}$ be a set of $|G| = N$ agents. Let $\mathcal{A} = \mathcal{A}^{[1]} \times ... \times \mathcal{A}^{[N]}$ be the space of joint actions where $\mathcal{A}^{[i]}$ is the set of actions available to agent $G^{[i]}$ for all i. Let $\chi : \mathcal{A} \to \mathcal{T}$ be a utility function for the joint action space $\mathcal{A}$ and the joint target space $\mathcal{T} = \mathcal{T}^{[1]} \times ... \times \mathcal{T}^{[N]}$ where $\mathcal{T}^{[i]}$ is the target space of agent $G^{[i]}$ for all i. Unless stated otherwise, the utility χ is to be maximized. From χ we can derive a set of single-agent utility functions $\chi^{[i]} : \mathcal{A} \to \mathcal{T}^{[i]}$ for all i. A tuple $(G, \mathcal{A}, \mathcal{T}, \chi)$ is called a *normal-form game*.

Definition 6 (strategy)**.** Let $(G, \mathcal{A}, \mathcal{T}, \chi)$ be a normal-form game. In a single iteration of the game, an agent $G^{[i]}$'s behavior is given by a (possibly randomized) policy $\pi^{[i]} : () \to \mathcal{A}^{[i]}$. Then, $\pi^{[i]}$ is also called $G^{[i]}$'s *strategy*. If multiple iterations of the game are played, an agent $G^{[i]}$'s behavior can be given by a (possibly randomized) policy $\pi^{[i]} : \mathcal{A}^n \to \mathcal{A}^{[i]}$ where n is a number of previous iterations. The agent's strategy is then conditioned on a list of previous joint actions.
An agent $G^{[i]}$ whose actions are given via a policy $\pi^{[i]}$ of the form $\pi^{[i]}(\_) = a^{[i]}$ for some action $a^{[i]} \in \mathcal{A}^{[i]}$ is playing a pure strategy.
An agent $G^{[i]}$ whose actions are given via a policy $\pi^{[i]}$ of the form $\pi^{[i]}(\_) \sim A^{[i]}$ according to some distribution over $A^{[i]} \subseteq \mathcal{A}^{[i]}$ is playing a mixed strategy.
If a mixed strategy is based on a uniform distribution over actions $A^{[i]} \subseteq \mathcal{A}^{[i]}$, we write $\pi^{[i]} = A^{[i]}$ as a shorthand.

Definition 7 (Pareto front for strategies)**.** Let $(G, \mathcal{A}, \mathcal{T}, \chi)$ be a normal-form game. A joint strategy $\pi(\_) = (\pi^{[1]}(\_), ..., \pi^{[|G|]}(\_))$ Pareto-dominates another joint strategy $\pi'(\_) = (\pi'^{[1]}(\_), ..., \pi'^{[|G|]}(\_))$ iff for all agents $G^{[i]}$ it holds that

$$\chi^{[i]}(\pi(\_)) \geq \chi^{[i]}(\pi'(\_))$$

and there exists some agent $G^{[j]}$ so that

$$\chi^{[j]}(\pi(\_)) > \chi^{[j]}(\pi'(\_)).$$

A joint strategy π is Pareto-optimal iff there is no other strategy π' so that π' Pareto-dominates π.
The set of all Pareto-optimal strategies is called the Pareto front.

Definition 8 (best response)**.** Let $(G, \mathcal{A}, \mathcal{T}, \chi)$ be a normal-form game. Let $\pi^{[-i]}$ be a joint strategy of agents $G^{[1]}, ..., G^{[i-1]}, G^{[i+1]}, ..., G^{[N]}$, i.e., all agents except $G^{[i]}$. Let $\pi^{[i]} \oplus \pi^{[-i]}$ be a joint strategy of all agents then. Given a strategy $\pi^{[-i]}$ for all agents except $G^{[i]}$, $G^{[i]}$'s *best response* is the strategy $\pi^{*[i]}$ so that for all strategies $\pi'^{[i]}$ it holds that

$$\chi^{[i]}((\pi^{*[i]} \oplus \pi^{[-i]})(\_)) \geq \chi^{[i]}((\pi'^{[i]} \oplus \pi^{[-i]})(\_)).$$

Definition 9 (Nash equilibrium)**.** Let $(G, \mathcal{A}, \mathcal{T}, \chi)$ be a normal-form game played for a single iteration. A joint strategy π is a *Nash equilibrium* iff for all agents $G^{[i]}$ it holds that $\pi^{[i]}$ is the best response to $\pi^{[-i]}$.

Definition 10 (π-process)**.** Let N be a set of names ($N = \{$"a", "b", "c", ...$\}$, e.g.). $\mathbb{L}_\pi$ is the set of valid processes in the π-calculus given inductively via:

(null process) $0 \in \mathbb{L}_\pi$,

(τ prefix) if $P \in \mathbb{L}_\pi$, then $\tau.P \in \mathbb{L}_\pi$,

(receiving prefix) if $a, x \in N$ and $P \in \mathbb{L}_\pi$, then $a(x).P \in \mathbb{L}_\pi$,

(sending prefix) if $a, x \in N$ and $P \in \mathbb{L}_\pi$, then $\overline{a}\langle x\rangle.P \in \mathbb{L}_\pi$,

(choice) if $P, Q \in \mathbb{L}_\pi$, then $(P + Q) \in \mathbb{L}_\pi$,

(concurrency) if $P, Q \in \mathbb{L}_\pi$, then $(P \mid Q) \in \mathbb{L}_\pi$,

(scoping) if $x \in N$, $P \in \mathbb{L}_\pi$, then $(\nu x)\ P \in \mathbb{L}_\pi$,

(replication) if $P \in \mathbb{L}_\pi$, then $!P \in \mathbb{L}_\pi$.

Any element $P \in \mathbb{L}_\pi$ is called a π-process. If the binding order is clear, we leave out parentheses.
The free names of a π-process $P \in \mathbb{L}_\pi$, written $\mathfrak{F}(P)$ with $\mathfrak{F} : \mathbb{L}_\pi \to \wp(N)$ are given inductively via:

- $\mathfrak{F}(0) = \emptyset$,
- $\mathfrak{F}(\tau.P) = \mathfrak{F}(P)$,
- $\mathfrak{F}(a(x).P) = \{a\} \cup (\mathfrak{F}(P) \setminus \{x\})$,
- $\mathfrak{F}(\overline{a}\langle x\rangle.P) = \{a, x\} \cup \mathfrak{F}(P)$,
- $\mathfrak{F}(P + Q) = \mathfrak{F}(P) \cup \mathfrak{F}(Q)$,
- $\mathfrak{F}(P \mid Q) = \mathfrak{F}(P) \cup \mathfrak{F}(Q)$,
- $\mathfrak{F}((\nu x)\ P) = \mathfrak{F}(P) \setminus \{x\}$,
- $\mathfrak{F}(!P) = \mathfrak{F}(P)$,

for any $a, x \in N$ and any $P, Q \in \mathbb{L}_\pi$.

Definition 11 (π-congruence)**.** Two π-processes $P, Q \in \mathbb{L}_\pi$ are structurally congruent, written $P \equiv Q$, if they fulfill the predicate $\equiv\colon \mathbb{L}_\pi \times \mathbb{L}_\pi \to \mathbb{B}$. We define inductively:

(α-conversion) $P \equiv Q$ if both only differ by the choice of bound names,

(choice rules) $P + Q \equiv Q + P$, and $(P + Q) + R \equiv P + (Q + R)$, and $P \equiv P + P$

(concurrency rules) $P \mid Q \equiv Q \mid P$, and $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$, and $P \equiv P \mid 0$,

(scoping rules) $(\nu x)\ (\nu y)\ P \equiv (\nu y)\ (\nu x)\ P$, and $(\nu x)\ (P \mid Q) \equiv P \mid ((\nu x)\ Q)$ if $x \notin \mathfrak{F}(P)$, and $(\nu x)\ 0 \equiv 0$,

(replication rules) $!P \equiv P \mid !P$,

for any names $x, y \in N$ and processes $P, Q, R \in \mathbb{L}_\pi$.

Definition 12 (π-substitution)**.** For a π-process P we write $P[y := z]$ for the π-process where every free occurrence of name y is replaced by name z. Formally, we define:

- $0[y := z] = 0$,
- $(\tau.P)[y := z] = \tau.(P[y := z])$,
- $(a(x).P)[y := z] = z(x).P$ for $a = y$ and $x = y$,
- $(a(x).P)[y := z] = a(x).P$ for $a \neq y$ and $x = y$,
- $(a(x).P)[y := z] = z(x).(P[y := z])$ for $a = y$ and $x \neq y$,
- $(a(x).P)[y := z] = a(x).(P[y := z])$ for $a \neq y$ and $x \neq y$,
- $(\overline{a}\langle x\rangle.P)[y := z] = \overline{z}\langle z\rangle.(P[y := z])$ for $a = y$ and $x = y$,
- $(\overline{a}\langle x\rangle.P)[y := z] = \overline{a}\langle z\rangle.(P[y := z])$ for $a \neq y$ and $x = y$,
- $(\overline{a}\langle x\rangle.P)[y := z] = \overline{z}\langle x\rangle.(P[y := z])$ for $a = y$ and $x \neq y$,
- $(\overline{a}\langle x\rangle.P)[y := z] = \overline{a}\langle x\rangle.(P[y := z])$ for $a \neq y$ and $x \neq y$,
- $(P + Q)[y := z] = (P[y := z]) + (Q[y := z])$,
- $(P \mid Q)[y := z] = (P[y := z]) \mid (Q[y := z])$,
- $((\nu x)\ P)[y := z] = ((\nu x)\ P)$ for $x = y$,
- $((\nu x)\ P)[y := z] = (\nu x)\ (P[y := z])$ for $x \neq y$,
- $(!P)[y := z] = !(P[y := z])$,

for any names $a, x, y, z \in N$ and processes $P, Q \in \mathbb{L}_\pi$.

Definition 13 (π-evaluation)**.** An evaluation of a π-process P is a sequence of π-processes $P_0 \rightarrowtail ... \rightarrowtail P_n$ where $P_0 = P$ and P_{i+1} is generated from P_i via the application of an evaluation rule $\rightarrowtail: \mathbb{L}_\pi \rightarrow \mathbb{L}_\pi$ to any sub-term of P_i. We define the following evaluation rules:

(reaction) $(a(x).P + P') \mid (\overline{a}\langle y\rangle.Q + Q') \rightarrowtail_{\text{REACT}} (P[x := y]) \mid Q$,

(τ transition) $\tau.P + P' \rightarrowtail_{\text{TAU}} P$,

(parallel execution) $P \mid R \rightarrowtail_{\text{PAR}} Q \mid R$ if it holds that $P \rightarrowtail Q$,

(restricted execution) $(\nu x)\ P \rightarrowtail_{\text{RES}} (\nu x)\ Q \mid R$ if it holds that $P \rightarrowtail Q$,

(structural congruence) $P' \rightarrowtail_{\text{STRUCT}} Q'$ if it holds that $P \rightarrowtail Q$ and $P \equiv P'$ and $Q \equiv Q'$,

for any names $a, x, y \in N$ and processes $P, P', Q, Q' \in \mathbb{L}_\pi$ where $\equiv: \mathbb{L}_\pi \times \mathbb{L}_\pi \rightarrow \mathbb{B}$ is the predicate for structural congruence.