

# Computational Intelligence 2022/23

Second Exam (Solution)\* — April 19th, 2023

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**

<b>First Name:</b>																					
<b>Last Name:</b>																					
<b>Matriculation Number:</b>																					
<table border="1"><tr><td>Topic 1</td><td>max. 10 pts.</td><td>pts.</td></tr><tr><td>Topic 2</td><td>max. 22 pts.</td><td>pts.</td></tr><tr><td>Topic 3</td><td>max. 20 pts.</td><td>pts.</td></tr><tr><td>Topic 4</td><td>max. 18 pts.</td><td>pts.</td></tr><tr><td>Topic 5</td><td>max. 20 pts.</td><td>pts.</td></tr><tr><td colspan="2">Total max. 90 pts.</td><td>pts.</td></tr></table>				Topic 1	max. 10 pts.	pts.	Topic 2	max. 22 pts.	pts.	Topic 3	max. 20 pts.	pts.	Topic 4	max. 18 pts.	pts.	Topic 5	max. 20 pts.	pts.	Total max. 90 pts.		pts.
Topic 1	max. 10 pts.	pts.																			
Topic 2	max. 22 pts.	pts.																			
Topic 3	max. 20 pts.	pts.																			
Topic 4	max. 18 pts.	pts.																			
Topic 5	max. 20 pts.	pts.																			
Total max. 90 pts.		pts.																			

---

\*Corrections to the original handout are marked in red in this file.

## 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) Turing designed the Turing Test to test an artificial intelligence's ability to...

i simulate a Turing machine.	ii fit inside a robot.	iii <u>communicate like a human being.</u>	iv produce the technological singularity.
------------------------------	------------------------	--	---

(b) Let  $equal(x, y) \iff (\forall p : p(x) \iff p(y))$ . This approach to give an equality predicate is called...

i emotional	ii intentional	iii <u>extensional</u>	iv logical
-------------	----------------	------------------------	------------

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

i producing log files	ii <u>programming logic</u>	iii probabilistic logarithms	iv pranking old ogres
-----------------------	-----------------------------	------------------------------	-----------------------

(d) Which of these is not a standard component of any neural network?

i weights	ii <u>concentrations</u>	iii biases	iv activation functions
-----------	--------------------------	------------	-------------------------

(e) The optimization algorithm *simulated annealing* compares its current solution candidate to a randomly generated solution candidate from its neighborhood. The chance of accepting a worse solution candidate...

i is virtually always 100%.	ii increases with computational effort spent on the new candidate.	iii <u>decreases according to a temperature parameter.</u>	iv is computed by an artificial intelligence.
-----------------------------	--	--	---

(f) A programming language that is especially designed to program interaction-based systems and has found usage in large-scale communication applications is called ...

i Münch	ii Augsburg	iii <u>Erlang</u>	iv Neu-U
---------	-------------	-------------------	----------

(g) One of the first breakthroughs in AI image generation around 2015 was achieved by the GAN model. Its key concept involved network training that is...

i	<u>adversarial</u>	ii absolute	iii agnostic	iv astonishing
---	--------------------	-------------	--------------	----------------

(h) In a single-iteration game of Prisoner's Dilemma with actions *cooperate* (C) and *defect* (D), the Nash equilibrium is the joint action...

i (C, C)	ii (C, D)	iii (D, C)	iv <u>(D, D)</u>
----------	-----------	------------	------------------

(i) When we deploy a large amount of simple robots without any centralized control unit, we call these robots...

i <u>a swarm</u>	ii a school	iii a pack	iv an empire
------------------	-------------	------------	--------------

(j) The idea of technological singularity is based on ever accelerating...

i warp speeds	ii <u>scientific</u> <u>progress</u>	iii force abilities	iv agent cloning
---------------	---	---------------------	------------------

## 2 Agents and Goals

22pts

Consider a power grid consisting of 100 power nodes located on a  $10 \times 10$  grid. Per default, each power node is assigned the state **working**. If a cell detects a problem, it sounds an alarm and switches to the state **alarming**. After a problem has not been treated for an indeterminate amount of time, the power node may switch to the state **defect** and thereby stop functioning. The state of the power node at position  $(x, y)$  can be checked by calling a function  $\text{state\_of} : \mathcal{C}^2 \rightarrow \mathcal{M}$  where  $\mathcal{C} = [0; 9] \subset \mathbb{N}$  is the space of grid positions and  $\mathcal{M} = \{\text{working}, \text{alarming}, \text{defect}\}$  is the space of power node states, i.e., states that a single power node can be in.

To keep the grid in good shape, we deploy a small drone called *RepairBot*. It can be present at exactly one power node at the same time. It can observe its current position coordinates  $(x, y) \in \mathcal{C}^2$  as well as the nearest **alarming** power node's coordinates  $(x^\dagger, y^\dagger) \in \mathcal{C}^2$ . With one action (which also means within one time step), the *RepairBot* can move to any of the 8 neighboring power nodes (horizontally, vertically, and diagonally), do nothing, or repair the node it is currently at; the *RepairBot* thus has an action space  $\mathcal{A} = \{\text{move\_northwest}, \text{move\_north}, \dots, \text{move\_southeast}, \text{do\_nothing}, \text{repair}\}$  with  $|\mathcal{A}| = 10$ . The move actions place the *RepairBot* at the corresponding neighboring power node in the next time step, **do\\_nothing** (obviously) does not change the robot's position, and **repair** puts the power node where the robot is at in the power node state **working**, regardless of what its previous state was. If the *RepairBot* attempts to move in a direction where there is no neighboring power node (which happens at the edges of the grid), nothing happens.

Assume that the complete state of the whole system (consisting of the power grid and the *RepairBot*) is given by one instance of a function fulfilling the definition of the **state\\_of** function as given above as well as the position of the *RepairBot*. The whole system then generates a sequence of system states  $\langle s_t \rangle_{0 \leq t \leq T}$  for some fixed maximum episode length  $T \in \mathbb{N}$  and every system state  $s_t \in \mathcal{S}$  with  $\mathcal{S} = (\mathcal{C}^2 \rightarrow \mathcal{M}) \times \mathcal{C}^2$  in accordance with all above specifications.

(i) As stated above, the *RepairBot* can observe  $(x^\dagger, y^\dagger)$ , which is the nearest **alarming** power node. Why is this coordinate data not needed as a separate part of the system state definition? (2pts)

When all positions are given, that data can easily be deducted when the environment generates the observation from the state.

(ii) Give a goal predicate  $\gamma : \langle \mathcal{S} \rangle \rightarrow \mathbb{B}$  so that  $\gamma(\langle \text{state\_of}_t, (x_t, y_t) \rangle_{0 \leq t \leq T})$  holds iff no power node has ever been **defect** and any **alarming** power node is **working** again (at least once) within at most 5 time steps. (12pts)

$$\begin{aligned} \gamma(\langle \text{state\_of}_t, (x_t, y_t) \rangle_{0 \leq t \leq T}) \iff \forall t : \forall X, Y : \\ \text{state\_of}_t((X, Y)) \neq \text{defect} \\ \wedge \text{state\_of}_t((X, Y)) = \text{alarming} \implies \\ \exists t' : t' - t \leq 5 \wedge \text{state\_of}_{t'}((X, Y)) = \text{working} \end{aligned}$$

(iii) Now consider that a policy for *RepairBot* is generated via standard optimization and/or reinforcement learning techniques to maximize the undiscounted accumulated reward given by the reward function  $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  defined via

$$R((\text{state\_of}_t, (x_t, y_t)), a_t, (\text{state\_of}_{t+1}, (x_{t+1}, y_{t+1}))) = \begin{cases} +10 & \text{if } a_t = \text{repair and state\_of}_t((x_t, y_t)) = \text{defect}, \\ +5 & \text{if } a_t = \text{repair and state\_of}_t((x_t, y_t)) = \text{alarming}, \\ -3 & \text{else if } \exists X, Y : \text{state\_of}_t((X, Y)) \in \{\text{defect}, \text{alarming}\}, \\ 0 & \text{otherwise.} \end{cases}$$

Explain why this reward function might easily lead to unwanted behavior, i.e., might facilitate the violation of goal  $\gamma$  as given above, and how this behavior looks like. Add to the definition of  $R$  (i.e., do not alter the given if-cases but feel free to add new ones) so that  $R$  now incentivizes the *RepairAgent* to actually fulfill  $\gamma$ . (8pts)

Since the *RepairAgent* is not punished for waiting around, it will gain a higher reward when it lets nodes go **defect** and only then repairs them. This is likely to allow more time steps to pass, even when the agent *could* have acted quicker. For an improved reward function  $R$ , see above.

### 3 Fuzzy Logic and Optimization 20pts

We still consider a power grid consisting of 100 power nodes located on a  $10 \times 10$  grid. The state of the power node at position  $(x, y)$  can be checked by calling a function `state_of` :  $\mathcal{C}^2 \rightarrow \mathcal{M}$  where  $\mathcal{C} = [0; 9] \subset \mathbb{N}$  and  $\mathcal{M} = \{\text{working}, \text{alarming}, \text{defect}\}$ . A *RepairBot* travels through that grid by performing actions from the action space  $\mathcal{A} = \{\text{move\_northwest}, \text{move\_north}, \dots, \text{move\_southeast}, \text{do\_nothing}, \text{repair}\}$  with  $|\mathcal{A}| = 10$ .

(i) We now want to evaluate how well the power nodes of the power grid are treated. We ask power grid whisperers and they tell us that the happiness of a power grid node at position  $p = (x, y)$  can be computed via

$$\text{happy}(p) = \text{OR}(\text{very}(\text{central}(p)), \text{cared}(p))$$

where

$$\text{central}((x, y)) = \frac{10 - |5 - x| - |5 - y|}{10}$$

only depends on the node's position and

$$\text{cared}((x, y)) = \frac{H(x, y)}{100}$$

can change over time where  $H(x, y)$  is the number of times that the *RepairBot* visited this power node's position during the last 100 time steps, i.e., the number of time steps for which the *RepairBot*'s position was  $(x, y)$ .

Furthermore, OR and NOT are the commonly known fuzzy logic functions and *very* is given via

$$\text{very}(\mu) = \mu^2.$$

Assume that the *RepairBot* follows a policy that makes it stand still at node position  $(4, 2)$  infinitely. Compute  $\text{happy}((4, 2))$  and  $\text{happy}((5, 5))$  including all intermediate results, say which of these two nodes is a global optimum (or say if both of them are), and give a brief argument why no other nodes are global optima. (11pts)

$$\begin{aligned}
central((4, 2)) &= \frac{10 - 1 - 3}{10} = 0.6 \\
very(central((4, 2))) &= 0.6^2 = 0.36 \\
cared((4, 2)) &= \frac{100}{100} = 1 \\
happy((4, 2)) &= \max(0.36, 1) = 1 \\
\\ 
central((5, 5)) &= \frac{10 - 0 - 0}{10} = 1 \\
very(central((5, 5))) &= 1^2 = 1 \\
cared((5, 5)) &= \frac{0}{100} = 0 \\
happy((5, 5)) &= \max(1, 0) = 1
\end{aligned}$$

Both these nodes are global optima and no other node can be, because the *central* part of the OR is clearly centered around (5,5) and the *cared* part is only non-zero for this one node.

(ii) We now want to find a policy for the *RepairBot* via an evolutionary algorithm. To this end, we consider only policies which run for exactly 1000 time steps and are fully defined by a parameter  $\theta \in \mathcal{A}^{1000}$ , i.e., a list of actions with length 1000.

We define that a nice mutation function  $\text{mutate} : \mathcal{A}^{1000} \rightarrow \mathcal{A}^{1000}$  for such an evolutionary algorithm has the following properties:

- It changes exactly one action within the policy. That action can be any other action afterwards.
- The mutated policy is not fully determined by the parent policy but depends on random effects in some way.
- Through iterative application we can produce any arbitrary policy.

Give a nice mutation function **mutate**. (9pts)

Hint: You may use a probabilistic function **random** :  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  by calling **random**( $i, j$ ), which returns a random integer within  $[i; j) \subset \mathbb{N}$  according to a uniform distribution.

$$\begin{aligned} \text{mutate}(\theta) &= \theta' \\ \text{where for all } i \in [0, 1000) : \theta'_i &= \begin{cases} a & \text{if } i = r, \\ \theta_i & \text{otherwise,} \end{cases} \\ \text{where } a &\sim \mathcal{A} \setminus \{\theta_r\} \\ \text{and } r &= \text{random}(0, 1000) \end{aligned}$$



## 4 Game Theory and Coordination

18pts

Consider a power grid in which all nodes can choose between two different strategies: Strategy  $E$  signals **alarming** as early as possible while strategy  $L$  signals **alarming** as late as possible. When all power nodes signal early, the *RepairBot* gets overwhelmed and, when all nodes signal late, the chance of the grid breaking down increases. Thus, when two power nodes need to decide on a strategy of how to attract the *RepairBot*, formally, they play the following two-player normal-form game we call *alarm signaling*:

$$\begin{array}{cc} & E & L \\ E & (2, 2) & (3, 1) \\ L & (1, 3) & (1, 1) \end{array}$$

(i) Show that  $(E, E)$  is the only pure Nash equilibrium (cf. Definitions 6 and 8 in the appendix) for *alarm signaling* by denoting all best responses. (4pts)

The best responses are....

$$\begin{array}{cc} & E & L \\ E & (\underline{2}, \underline{2}) & (\underline{3}, 1) \\ L & (1, \underline{3}) & (1, 1) \end{array}$$

(ii) Show that the strategy  $E$  is an evolutionary stable strategy (cf. Definition 9 in the Appendix) in a population of power nodes. (6pts)

We already know that  $E \otimes E = (E, E)$  is a Nash equilibrium. It also holds that

$$\begin{aligned} \chi^{[1]}(E \otimes L) &> \chi^{[1]}(L \otimes L) \\ 3 &> 1 \end{aligned}$$

so  $E$  is an evolutionary stable strategy.

(iii) When the *RepairBot* finally goes up to an **alarming** power node, both follow a fixed protocol of interaction that is specified for the power node as follows in  $\pi$ -calculus:

$$\begin{aligned} \text{PowerNode}(\text{name}, \text{state}) = & \text{announce}(\text{bot\_capability}). \\ & \overline{\text{introduce}}\langle \text{name} \rangle. \\ & (\text{do\_nothing}.\text{PowerNode}(\text{name}, \text{state}) \\ & + \text{repair}.\text{PowerNode}(\text{name}, \text{bot\_capability})) \end{aligned}$$

Give a  $\pi$ -process *RepairBot* so that the process

$$\text{RepairBot} \mid \text{PowerNode}(1, \text{alarming}) \mid \text{PowerNode}(2, \text{defect})$$

always evaluates to the process

$$\text{RepairBot} \mid \text{PowerNode}(1, \text{working}) \mid \text{PowerNode}(2, \text{working}) \mid \text{Bill}(1) \mid \text{Bill}(2)$$

where *Bill*(*name*) is a named (black-box) process that cannot be evaluated any further for any *name* and can be called freely within your namespace. (8pts)

Hint: It is perfectly fine to repair a **working** *PowerNode* as well.

$$\begin{aligned} \text{RepairBot} = & \overline{\text{announce}}\langle \text{working} \rangle. \\ & \text{introduce}\langle \text{name} \rangle. \\ & \overline{\text{repair}}. \\ & (\text{RepairBot} \mid \text{Bill}(\text{name})) \end{aligned}$$

## 5 Scientific Reading: Empowerment

20pts

Read the following (correct) explanation of the novel idea of *empowerment*, which we did not discuss in the lecture, and answer the questions afterwards.

### The CoIn Team:

Generate a one-page explanation of empowerment in reinforcement learning. Include and explain relevant formulas in LaTeX format. Make the text suitable for computer science graduate students.

### The ChatGPT Assistant:

Empowerment in reinforcement learning refers to the ability of an agent to maximize its influence over the environment. It is an intrinsic motivation that encourages agents to explore and take actions that can lead to long-term benefits, even if they do not result in immediate rewards.

One way to measure empowerment is through information-theoretic concepts, such as the mutual information between the agent's actions and the resulting state of the environment. Mutual information is a measure of the degree of dependence between two variables, and in this case, it can be used to capture how much the agent's actions influence the environment. The formula for mutual information is:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

Where  $X$  and  $Y$  represent two random variables, and  $p(x, y)$ ,  $p(x)$ , and  $p(y)$  represent their joint and marginal probabilities, respectively.

To apply empowerment in reinforcement learning, the agent maximizes the mutual information between its actions and the resulting state of the environment. This can be achieved through various algorithms, such as empowerment-based exploration or empowerment-driven reinforcement learning.

Empowerment has several benefits in reinforcement learning. First, it encourages agents to explore their environment more broadly, which can lead to the discovery of new strategies and solutions. Second, it can help agents to learn more robust and adaptable policies, as they focus on the long-term effects of their actions rather than just maximizing immediate rewards.

In conclusion, empowerment is a valuable concept in reinforcement learning that can help agents to become more effective at achieving their goals. By maximizing the mutual information between its actions and the environment, an agent can become more empowered and navigate complex and changing environments more efficiently.

### The CoIn Team:

Give a very short overview of common problems when using empowerment.

### The ChatGPT Assistant:

One common problem when using empowerment in reinforcement learning is that it can be difficult to estimate the mutual information between the agent's actions and the environment, which is required to compute empowerment. Another challenge is that maximizing empowerment can sometimes lead to suboptimal behavior if the agent becomes overly focused on exploring at the expense of exploiting known high-reward actions. Finally, the computational complexity of empowerment-based algorithms can be high, which may limit their practical applicability in some settings.

(i) Answer the following question partly generated by ChatGPT: Explain the concept of empowerment in reinforcement learning, and give an example of how it can be used to encourage more efficient exploration. (8pts)

**The ChatGPT Assistant:**

Empowerment in reinforcement learning refers to the agent's ability to maximize its influence over the environment, by taking actions that lead to long-term benefits even if they do not result in immediate rewards. One way to measure empowerment is through information-theoretic concepts, such as the mutual information between the agent's actions and the resulting state of the environment. Maximizing empowerment can encourage more efficient exploration, as the agent focuses on actions that have a high potential to influence the environment, rather than just maximizing immediate rewards.

(ii) Discuss one potential challenge in using empowerment as a learning signal for reinforcement learning agents. (4pts)

**The ChatGPT Assistant:**

One potential challenge in using empowerment as a learning signal for reinforcement learning agents is that it can be difficult to estimate the mutual information between the agent's actions and the environment. This can result in high computational complexity, which may limit the practical applicability of empowerment-based algorithms. One possible solution to address this issue is to use approximations or simplified models to estimate mutual information, or to combine empowerment with other learning signals such as reward-based reinforcement learning to balance exploration and exploitation more effectively.

(iii) Again imagine a single *RepairBot* traveling through a grid of power nodes as we considered throughout this exam. In a dynamic environment, where nodes can become **alarming** in a chaotic way, what kind of behavior of the *RepairBot*, which is rewarded for not letting nodes be **defect** for long, would we expect when the *RepairBot* is trained to maximize empowerment as well? If the *RepairBot* could teleport to any arbitrary power node at will, how would that change its empowerment? (8pts)

A *RepairBot* trained for empowerment would prefer to visit locations that make it easy to reach the whole grid, thus maximizing the effect of its actions by repositioning near the center of the grid, for example.

If the *RepairBot* could teleport, its empowerment would be increased as it can reach every position via a single action. Consequently, its positioning within the grid would no longer matter.

## Appendix: Definitions

**Notation.**  $\wp(X)$  denotes the power set of  $X$ .  $\mathbb{E}$  denotes the expected value.  $\#$  denotes vector or sequence concatenation, i.e., given two vectors  $\mathbf{x} = \langle x_1, \dots, x_{|\mathbf{x}|} \rangle$  and  $\mathbf{y} = \langle y_1, \dots, y_{|\mathbf{y}|} \rangle$ ,  $\mathbf{x} \# \mathbf{y} = \langle x_1, \dots, x_{|\mathbf{x}|}, y_1, \dots, y_{|\mathbf{y}|} \rangle$ . A vector  $\langle x_0, \dots, x_{n-1} \rangle$  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.).

**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  $\pi : \mathcal{O} \rightarrow \mathcal{A}$ . 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 set called state space. Let  $\mathcal{T}$  be an arbitrary set called target space and  $\leq$  be a total order on  $\mathcal{T}$ . A total function  $\tau : \mathcal{X} \rightarrow \mathcal{T}$  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  $e$  be a possibly randomized or non-deterministic function so that the optimization run is produced by calling  $e$  repeatedly, i.e.,  $x_{t+1} = e(\langle x_u \rangle_{0 \leq u \leq t}, \tau)$  where  $x_0$  is given externally (i.e.,  $x_0 =_{\text{def}} 42$ ) or chosen randomly (i.e.,  $x_0 \sim \mathcal{X}$ ). 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  $neighbors : \mathcal{X} \rightarrow \wp(\mathcal{X})$  be a function that returns a set of neighbors of a given state  $x_u \in \mathcal{X}$ . Let  $x'_u \sim neighbors(x_u)$  be a neighbor state of  $x_u$  that was drawn at random. Let  $T : \mathbb{N} \rightarrow \mathbb{R}$  be a temperature schedule, i.e., a function that returns a temperature value for each time step. Let  $A : \mathcal{T} \times \mathcal{T} \times \mathbb{R} \rightarrow \mathbb{P}$  be a function that returns an acceptance probability given two target values and a temperature. Commonly, we set

$$A(Q, Q', C) = e^{\frac{-(Q' - Q)}{C}}$$

for  $\mathcal{T} \subseteq \mathbb{R}$ . Let  $r \sim \mathbb{P}$  be a random number drawn from  $\mathbb{P} = [0; 1] \subset \mathbb{R}$ . The process  $\mathcal{D}$  continues via simulated annealing if  $e$  is of the form

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

**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  $\pi : \mathcal{S} \rightarrow \mathcal{A}$ . Let  $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{T}$  be a possibly randomized *cost (reward)* function. A Markov decision process is a tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{T}, P, R)$  where  $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{P}$  is the *transition probability function* often written as  $P(s'|s, a) = \Pr(s_{t+1} = s' \mid s_t = s \wedge a_t = a)$ .

A Markov decision process run for policy  $\pi$  is a tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \tau, \pi, \langle s_t \rangle_{t \in \mathbb{Z}}, \langle a_t \rangle_{t \in \mathbb{Z}})$  where

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

and where  $\tau$  is to be minimized (maximized) and usually has a form similar to

$$\begin{aligned} \text{accumulated cost (reward)} \tau(\pi) &=_{\text{def}} \sum_{t \in \mathbb{Z}} R(s_t, a_t, s_{t+1}) \\ \text{or discounted expected cost (reward)} \tau(\pi) &=_{\text{def}} \mathbb{E} \left[ \sum_{t \in \mathbb{Z}} \gamma^t \cdot R(s_t, a_t, s_{t+1}) \right] \end{aligned}$$

where  $\gamma \in [0; 1] \subset \mathbb{R}$  is called a discount factor.

A decision process generates a (possibly infinite) series of rewards  $\langle r_t \rangle_{t \in \mathbb{Z}}$  with  $r_t = R(s_t, a_t, s_{t+1})$ .

**Definition 4** (multi-agent system). Let  $G = \{G^{[1]}, \dots, G^{[N]}\}$  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  $i = 1, \dots, N$ , respectively. The multi-agent system  $G$  then takes a joint action  $a \in \mathcal{A}$  with  $\mathcal{A} = \mathcal{A}^{[1]} \times \dots \times \mathcal{A}^{[N]}$  after making a joint observation  $o \in \mathcal{O}$  with  $\mathcal{O} = \mathcal{O}^{[1]} \times \dots \times \mathcal{O}^{[N]}$  based on the joint policy  $\pi(o^{[1]}, \dots, o^{[N]}) = (a^{[1]}, \dots, a^{[N]})$  where  $a^{[i]} = \pi^{[i]}(o^{[i]})$  for all  $i$ .

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

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

**Definition 8** (Nash equilibrium). Let  $(G, \mathcal{A}, \mathcal{T}, \chi)$  be a normal-form game played for a single iteration. A joint strategy  $\pi$  is a *Nash equilibrium* iff for all agents  $G^{[i]}$  it holds that  $\pi^{[i]}$  is the best response to  $\pi^{[-i]}$ .

**Definition 9** (evolutionary stable strategy). Let  $(G, \mathcal{A}, \mathcal{T}, \chi)$  be a normal-form game played by two players for a single iteration. A strategy  $\pi^{[i]}$  for agent  $G^{[i]}$  is an *evolutionary stable strategy* iff

- $\pi = \pi^{[i]} \oplus \pi^{[-i]}$  with  $\pi^{[-i]} = \pi^{[i]}$  is a Nash equilibrium and
- for every other strategy  $\pi'^{[i]} = \pi'^{[-i]} \neq \pi^{[i]}$  it holds that

$$\chi^{[i]}((\pi^{[i]} \oplus \pi'^{[-i]})(\cdot)) > \chi^{[i]}((\pi'^{[i]} \oplus \pi'^{[-i]})(\cdot)).$$

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

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

**( $\tau$  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  $\bar{a}\langle x \rangle.P \in \mathbb{L}_\pi$ ,

**(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  $\pi$ -process. If the binding order is clear, we leave out parentheses.

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

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

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

**( $\tau$  transition)**  $\tau.P + P' \succ_{\text{TAU}} P,$

**(parallel execution)**  $P \mid R \succ_{\text{PAR}} Q \mid R$  if it holds that  $P \succ Q,$

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

**(structural congruence)**  $P' \succ_{\text{STRUCT}} Q'$  if it holds that  $P \succ Q$  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.