Computational Intelligence 2022/23

Mock Exam for the Exercise on February 2nd, 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

First Name:	
Last Name:	
Matriculation Number:	

Topic 1	max. 10 pts.	pts.
Topic 2	max. 10 pts.	pts.
Topic 3	max. 10 pts.	pts.
Topic 4	max. 10 pts.	pts.
Topic 5	max. 10 pts.	pts.
Topic 6	max. 10 pts.	pts.
Topic 7	max. 10 pts.	pts.
Topic 8	max. 20 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) According to Alan Turing, the goal of the *Imitation Game*, with a party A and B and a judge C, is for party A to convince ...

i B that $A \approx C$	ii C that $B \approx A$	iii C that $A \approx A$	iv A that $B = B$

(b) Which of the following is *not* commonly in use as a policy *encoding*?

i Python code	ii behavior trees	iii random execution	iv pseudo-code

(c) *Simulated annealing* is an...?

i optimization algorithm	ii game strategy	iii agent policy	iv type of neural network

(d) For *evolutionary algorithms* we usually do *not* associate more of which of the following techniques with *increased variation*?

i mutation	ii crossover	iii selection	iv recombination

(e) As measured by sample efficiency, all optimization algorithms perform the same when averaged over all possible target functions. What is this theorem called?

i always cheap breakfast	ii no free lunch	iii sometimes expensive dinner	iv never available midnight snack

(f) The *Pareto front* denotes the set of samples which we consider as ...

i non-optimal	ii fully randomized	iii dominated	iv non-dominated

(g) In the reading exercise we have seen Prolog used to implement an *expert system*, emulating the decision-making ability of a human (expert). For their inference advantage these systems mainly rely on large bodies of... ?

i goal classes	ii knowledge	iii probability	iv uncertainty

(h) Which of the following properties are *not* describing an evolutionary (robotic) swarm as we have defined it in the lecture?

i relatively simple	ii fully interdependent	iii mass producible	iv mostly identical

(i) The quote '[a] physical agent that performs tasks by manipulating the physical world' (Russel and Norvig (2003)) best describes a ... ?

i robot	ii PhD student	iii neural network	iv swarm collective

(j) Which of the following is *not* commonly considered a swarm behavior task?

i autonomous assembly	ii parallel computation	iii coordinated motion	iv decision making

2 Agents, Environments, Goals **10pts**

Consider a beaver in the final steps of building a dam. It just needs to add one more log in precisely the right angle. If it hits the right angle, the dam is now leak-proof and the water level (of the river flowing from behind the dam) rises. The beaver's behavior is thus given by a policy $\pi : \mathit{Level} \to \mathit{Angles}$ where $\mathit{Level} = [0; \infty) \subset \mathbb{R}$ is the water level in *cm* and $\mathit{Angles} = [0; 180] \times [0; 180] \subset \mathbb{R}^2$ are the horizontal and vertical angles in degrees at which the beaver tries to insert the final log into the dam.

(i) Assume that the environment is updated in discrete time steps and thus produces a sequence of states $\langle s_t \rangle_{1 \leq t \leq T}$ where $T \in \mathbb{N}$ is the episode length of the environment and $s_t \in \mathit{Level} \times \mathit{Angles}$ for all t. The beaver deems its work successful iff the water level keeps rising from some point in time on for at least three consecutive time steps and all future time steps afterwards. Give a goal predicate $\gamma : \langle \mathit{Level} \times \mathit{Angles} \rangle \to \mathbb{B}$ so that $\gamma(\langle s_t \rangle_{1 \leq t \leq T})$ holds iff the beaver deems its work successful. (4pts)

$$\gamma\big(\langle (l_t, \_) \rangle_{1 \leq t \leq T}\big) \iff \exists t : \ \ t \leq T - 3 \ \ \wedge \ \ \forall t' > t : l_{t'} > l_{t'-1}$$

(ii) We still assume that our environment's current state s is of the form $s \in \mathit{Level} \times \mathit{Angles}$. We cannot define a Markov Decision Process (MDP) so that its reward function R is optimal iff the goal predicate γ as given above is fulfilled. Briefly explain why. (2pts)

> **Definition 1** (Markov decision process (MDP), *shortened definition*)**.** 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 $P(s'|s, a) = \Pr(s_{t+1} = s' \,|\, s_t = s \wedge a_t = a)$.

The goal predicate requires a hidden state as the history of water levels is necessary to compute its fulfillment but said history is not given within the state. Thus, it is inherently non-Markovian.

(iii) Now assume that we know that the perfect angles to put the log into the dam are $a^* = (42, 42)$. In order to train the beaver to use exactly these angles, we want to define a reward function $R' : \mathit{Angles} \to \mathbb{R}$ so that the beaver achieves the highest reward iff it uses the angles a^*. For all other angles, we want a smooth reward landscape with a gradient pointing towards a^*. Give a possible definition for R'. (4pts)

$$R'((x, y)) = -(x - 42)^2 - (y - 42)^2$$

3 Optimization and Search 10pts

Assume that a beaver's policy for putting logs into dams is encoded by a parameter vector $\theta \in \Theta = \{A, C, G, T\}^5$ where A, C, G, T are arbitrary fixed symbols. We are given a fitness function $\phi : \Theta \to \mathbb{R}$ so that $\phi(\theta)$ encodes the time it takes a beaver with policy π_θ to put a log into a dam successfully, i.e., lower values of $\phi(\theta)$ are better.

Algorithm 1 (basic evolutionary algorithm)**.** Let $\mathcal{E} = (\mathcal{X}, \mathcal{T}, \tau, E, \langle X_u \rangle_{0 \le u \le t})$ be a population-based optimization process. The process $\mathcal{E}$ continues via an evolutionary algorithm if E has the form

$$E(\langle X_u \rangle_{0 \le u \le t}, \tau) = X_{t+1} = \mathit{selection}\big(X_t \cup \mathit{variation}(X_t)\big)$$

where *selection* and *variation* are possibly randomized or non-deterministic functions so that $|\mathit{selection}(X)| \le |X|$ and $|\mathit{variation}(X)| \ge |X|$ and $\big|\mathit{selection}\big(X_t \cup \mathit{variation}(X_t)\big)\big| = |X|$.

(i) We initialize a population $X \subseteq \Theta$ with population size $|X| = 10$. A nice *variation* function should

- construct 5 new individuals based on random candidates from the original population X,
- not always generate completely new individuals, but have them based on the given population X, and
- be able to reach every point in the search space through iterated application.

Give a complete definition for a nice *variation* function. (6pts)

$$\begin{aligned}
\mathit{variation}(\emptyset) &= \emptyset \\
\mathit{variation}(X) &= \{\mathit{mutate}(x)\} \cup \mathit{variation}(X \setminus \{x, x'\}) \\
&\quad \text{where } X \neq \emptyset \\
&\quad \text{and } x, x' \sim X \\
\text{with } \mathit{mutate}(x) &= (x'_0, x'_1, x'_2, x'_3, x'_4) \\
&\quad \text{where } x'_i = \begin{cases} g \sim \{A, C, G, T\} & \text{if } i = j \\ x_i & \text{otherwise} \end{cases} \\
&\quad \text{and } j \sim \{0, ..., 4\}
\end{aligned}$$

(ii) Consider the following two definitions of a *selection* function:

$$\mathit{selection}_1(X) = \begin{cases} \mathit{selection}_1\big(X \setminus \arg\max_{x \in X} \phi(x)\big) & \text{if } |X| > 10, \\ X & \text{otherwise,} \end{cases}$$

$$\mathit{selection}_2(X) = \begin{cases} \mathit{selection}_2\big(X \setminus x\big) & \text{for } x \sim X \text{ if } |X| > 10, \\ X & \text{otherwise.} \end{cases}$$

Briefly explain why both functions fulfill the definition for a *selection* function in Algorithm 1. Which of these two functions exerts the stronger selection pressure? Which of these two functions is more usable in the context of optimization? State your argument. (4pts)

Both functions reduce the given population (recursively) until $|X| = 10$ is reached. $\mathit{selection}_1$ exerts the stronger selection pressure as $\mathit{selection}_2$ exerts no directed pressure at all. Thus, only $\mathit{selection}_1$ is usable for optimization.

4 Reinforcement Learning 10pts

Recall the Vacuum World from the lecture. It consist of two rooms A, B, which can be dirty or clean. An agent is positioned in exactly one of these rooms and can execute one of two actions: switch the room (action *switch*) or vacuum the room (action *vacuum*) with the obvious result. If the agent chooses to vacuum, it receives a reward of $+10$ if the room it is in was dirty or -1 if it was not dirty. Other actions inflict no reward nor cost. The initial state of the Vacuum World is given as the agent being positioned in room A, room A being clean, and room B being dirty.

(i) Recall Definition 1 from the lecture (which is printed in this exam under task 2(ii)) and provide suitable definitions for

- the state space $\mathcal{S}$,
- the action space $\mathcal{A}$,
- the target space $\mathcal{T}$, and
- the reward function R.

You do not need to give the state transition probability function P. However, note that state transition is deterministic in this case. (5pts)

$$\mathcal{S} = \{A, B\} \times \{\mathit{dirty}, \mathit{clean}\} \times \{\mathit{dirty}, \mathit{clean}\},$$
$$\mathcal{A} = \{\mathit{switch}, \mathit{vacuum}\},$$
$$\mathcal{T} = \mathbb{R},$$
$$R(s,a) = \begin{cases} +10 & \text{if } s \in \{(A, \mathit{dirty}, \_), (B, \_, \mathit{dirty})\} \text{ and } a = \mathit{vacuum}, \\ -1 & \text{if } s \in \{(A, \mathit{clean}, \_), (B, \_, \mathit{clean})\} \text{ and } a = \mathit{vacuum}, \\ 0 & \text{otherwise} \end{cases}$$

(ii) Consider the fixed policy π which executes the actions *vacuum*, *switch*, *vacuum*, *vacuum* in that order and then stops execution. Execute π and for each generated state s during the execution (including the initial state $s = s_0$) compute its value $V^\pi(s)$ given a fully deterministic state transition function P and $\gamma = 1$. To this end, recall the Bellman equation as given in the lecture:

Theorem 1 (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 π being executed starting from state s is given via π'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 π^* is given via

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

(5pts)

$$\begin{aligned} s_0 &= \{A, \mathit{clean}, \mathit{dirty}\} \\ s_1 &= \{A, \mathit{clean}, \mathit{dirty}\} \\ s_2 &= \{B, \mathit{clean}, \mathit{dirty}\} \\ s_3 &= \{B, \mathit{clean}, \mathit{clean}\} \\ s_4 &= \{B, \mathit{clean}, \mathit{clean}\} \\ V^\pi(s_4) &= 0 \\ V^\pi(s_3) &= -1 \\ V^\pi(s_2) &= 9 \\ V^\pi(s_1) &= 9 \\ V^\pi(s_0) &= 8 \end{aligned}$$

5 Fuzzy Logic **10pts**

A home buyer has to decide between the houses A and B. Thus, he makes his decision considering the criteria *price* and *location*. The houses are similar with respect to their location — which the buyer is familiar with — and statements regarding their pricing are already assigned with 'truth values' (see table below). For the price, however, the buyer has to consult a survey, here pictured in Figure 1.

	Statement	Truth value
ψ_1	Price of house A (400k EUR) is good	?
ψ_2	Location of house A is good	0.6
ψ_3	Price of house B (600k EUR) is good	?
ψ_4	Location of house B is good	0.7

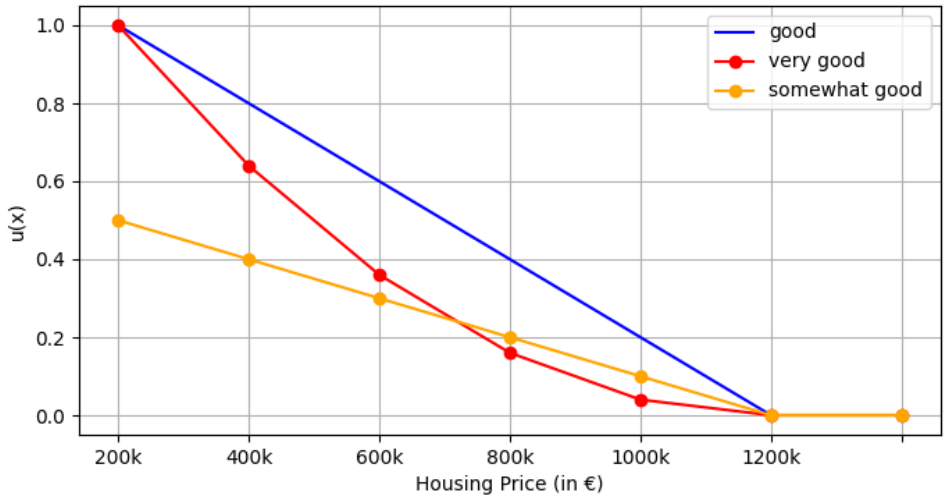

Figure 1: Housing Prices — Fuzzy Membership Function [Solution]

(i) Now consider that our buyer gets more picky and now wants a *very* nice house. We have seen linguistic modifiers defined in the lecture before; we again define the operator *very* as a modification of the truth value membership function as the following:

$$very(a) = a^2$$

In the Figure 1, add the graph for this modification. Your function should be defined at

least for the points

$$x_1 = 400k,$$
$$x_2 = 600k,$$
$$x_3 = 1200k.$$

(2pts)

Cf. Figure 1 solution.

$$very\ \mu(x_1)^2 = 0.8^2 = 0.64$$
$$very\ \mu(x_2)^2 = 0.6^2 = 0.39$$
$$very\ \mu(x_3)^2 = 0^2 = 0$$

(ii) In case the buyer changes his mind, define a linguistic modifier *somewhat* with the only constraint

$$\mu(x) > somewhat(\mu(x))$$

Also add the graph for the membership function of *somewhat* in the Figure 1, again, defined for (at least) the points x_1, x_2, x_3 (2pts).

E.g.

$$somewhat\ (a) = a/2$$

$$somewhat\ \mu(x_1)/2 = 0.8/2 = 0.4$$
$$somewhat\ \mu(x_2)/2 = 0.6/2 = 0.3$$
$$somewhat\ \mu(x_3)/2 = 0/2 = 0$$

(iii) Finally, compute the truth value for the buyer considering the proposition '*The price of house x is very good* OR *the location of house x is somewhat good*'. In favor of which house does the buyer decide? (6pts)

$very(good\ price\ A\ 400k) = 0.8^2 = 0.64$
$somewhat(good\ location\ A) = 0.6/2 = 0.3$
$very(good\ price\ A) \vee somewhat(good\ location\ A) = \max(0.64, 0.3) = 0.64$

$very(good\ price\ B\ 600k) = 0.6^2 = 0.36$
$somewhat(good\ location\ B) = 0.7/2 = 0.35$
$very(good\ price\ B) \vee somewhat(good\ location\ B) = \max(0.36, 0.35) = 0.36$

$\rightarrow A > B$

6 Game Theory **10pts**

In the lecture we have defined the common notion of a normal-form game like this:

> **Definition 2** (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*.

Now let us consider the following new definition:

> **Definition 3** (mock exam game)**.** A *mock exam game* is a two-player normal-form game written as pair of $m \times n$ matrices (A, B), where m is the number of rows and n the number of columns. If agent $G^{[1]}$ (the row player) plays row i and $G^{[2]}$ (the column player) plays column j, then $\chi^{[1]} = a_{ij}$ and $\chi^{[2]} = b_{ij}$, where a_{ij}, b_{ij} are the corresponding entries of A and B respectively.

(i) Give one example of a zero-sum, two-player, two-action normal-form game. (1pts)

$$\begin{array}{c} \\ \text{L} \\ \text{R} \end{array} \begin{array}{c} \begin{array}{cc} \text{L} & \text{R} \end{array} \\ \begin{pmatrix} 0,0 & 1,-1 \\ -1,1 & 0,0 \end{pmatrix} \end{array}$$

(ii) Give one example of a zero-sum, two-player, two-action mock exam game. (2pts)

For the row player $G^{[1]}$:

$$A := \begin{array}{c} \\ \text{L} \\ \text{R} \end{array} \begin{array}{c} \begin{array}{cc} \text{L} & \text{R} \end{array} \\ \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \end{array}$$

For the column player $G^{[2]}$:

$$B := \begin{array}{c} \\ \text{L} \\ \text{R} \end{array} \begin{array}{c} \begin{array}{cc} \text{L} & \text{R} \end{array} \\ \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \end{array}$$

(iii) What are the two key differences between the definitions? (2pts)

- Two-players are explicitly assigned their roles as row- and column-player.
- The payoff matrices of these agents are written separately.

(iv) Remember the definition of the key concepts of Pareto-optimality from the lecture:

Definition 4 (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.

Can symmetrical mock exam games of the form

$$A := \begin{array}{c} \\ \text{X} \\ \text{Y} \end{array} \begin{array}{c} \begin{array}{cc} \text{X} & \text{Y} \end{array} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \end{array}, \qquad B := \begin{array}{c} \\ \text{X} \\ \text{Y} \end{array} \begin{array}{c} \begin{array}{cc} \text{X} & \text{Y} \end{array} \\ \begin{pmatrix} a & c \\ b & d \end{pmatrix} \end{array}$$

contain strategies that are also *Pareto-optimal*? If so, give an example for a symmetrical mock exam game with a payoff (defined by $a, b, c, d \in \mathbb{R}$) so that it contains at least one *Pareto-optimal* strategy. If not, show why. (5pts)

Yes, they can. We have already seen such an example in the Hawk–Dove Game, with

$$\begin{array}{c} \\ \text{Hawk} \\ \text{Dove} \end{array} \begin{array}{c} \begin{array}{cc} \text{Hawk} & \text{Dove} \end{array} \\ \begin{pmatrix} 0,0 & \underline{3},\underline{1} \\ \underline{1},\underline{3} & 2,2 \end{pmatrix} \end{array}$$

where (Hawk,Dove), (Dove,Hawk), and (Dove,Dove) are symmetrical *Pareto-Optimal* strategies (two of which also happen to be both Nash equilibria).

As exam game matrices:

$$A := \begin{array}{c} \\ \text{X} \\ \text{Y} \end{array} \begin{array}{c} \begin{array}{cc} \text{X} & \text{Y} \end{array} \\ \begin{pmatrix} 0 & 3 \\ 1 & 2 \end{pmatrix} \end{array} \qquad B := \begin{array}{c} \\ \text{X} \\ \text{Y} \end{array} \begin{array}{c} \begin{array}{cc} \text{X} & \text{Y} \end{array} \\ \begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix} \end{array}$$

Note: The simplest solution is probably something like

$$a = 100, b = c = d = 0$$

with a Pareto-optimum at (X, X).

7 The π-Calculus **10pts**

Recall the basic rules of π-calculus from the lecture:

Definition 5 (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 6 (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 $\rightarrowtail\colon \mathbb{L}_\pi \to \mathbb{L}_\pi$ 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 $\equiv\colon \mathbb{L}_\pi \times \mathbb{L}_\pi \to \mathbb{B}$ is the predicate for structural congruence.

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

$$\overline{a}\langle 42\rangle.b(x).\overline{c}\langle x\rangle.0 \mid a(y).\overline{b}\langle y\rangle.0 \mid c(z).S(z)$$

$$\begin{aligned}
&\rightarrowtail b(x).\overline{c}\langle x\rangle.0 \mid \overline{b}\langle 42\rangle.0 \mid c(z).S(z)\\
&\rightarrowtail \overline{c}\langle 42\rangle.0 \mid 0 \mid c(z).S(z)\\
&\rightarrowtail 0 \mid 0 \mid S(42)\\
&\rightarrowtail S(42)
\end{aligned}$$

$$\overline{a}.\overline{c}.\overline{b}.0 \mid (\overline{a}.A + \overline{c}.C) \mid !(a.A + b.B)$$

$$\begin{aligned}
&\rightarrowtail \overline{c}.\overline{b}.0 \mid (\overline{a}.A + \overline{c}.C) \mid A \mid !(a.A + b.B)\\
&\rightarrowtail \overline{c}.\overline{b}.0 \mid A \mid A \mid A \mid !(a.A + b.B)
\end{aligned}$$

$$a(x).b(y).S(x,y) + (a(x).A(x) \mid \overline{a}\langle Y\rangle.0)$$

$$\begin{aligned}
&\rightarrowtail a(x).b(y).S(x,y) + (A(Y) \mid 0)\\
&\rightarrowtail a(x).b(y).S(x,y) + A(Y)
\end{aligned}$$

(ii) Consider the following definition for π-processes to model the auction for buying a house:

$$\begin{aligned} \mathit{Bidder}(\mathit{current}) = {} & \mathit{give\_house}.\overline{\mathit{be\_happy}}.0 \\ & + \overline{\mathit{ask\_more}}\langle \mathit{current} \rangle.(\\ & \quad \mathit{approved}(\mathit{new}).\overline{\mathit{bid}}\langle \mathit{new} \rangle.\mathit{Bidder}(\mathit{new}) \\ & \quad + \mathit{denied}.0 \\ &) \\ \mathit{Seller} = {} & \mathit{bid}(\text{“low”}).\mathit{Seller} \\ & + \mathit{bid}(\text{“high”}).\overline{\mathit{give\_house}}.0 \end{aligned}$$

Give a definition for the π-process *Bank* so that

$$\mathit{Bidder}(\text{“low”}) \mid \mathit{Seller} \mid \mathit{Bank}$$

evaluates to the process 0. Give a short explanation of the behavior of your *Bank* in English or German (4pts).

Note: The behavior of *Bank* does not need to be plausible in a real-world scenario but merely needs to fulfill the requirements by the above definition and the rules of the π-calculus.

Possibly one of the simplest banks receives an ask for more, always approves “high” and then gives a target for the buyer’s happiness before vanishing.

$$\mathit{Bank} = \mathit{ask\_more}(\_).\overline{\mathit{approved}}(\text{“high”}).\mathit{be\_happy}.0$$

8 Scientific Reading: The Spi-Calculus 20pts

Read the adapted paper below[1] and answer the following questions.
Note: This excerpt is self-contained.

Abstract
We introduce the *spi-calculus*, an extension of the pi-calculus designed for the description and analysis of cryptographic protocols. We show how to use the spi calculus, particularly for studying authentication protocols. The pi-calculus (without extension) suffices for some abstract protocols; the spi calculus enables us to consider cryptographic issues in more detail. We represent protocols as processes in the spi-calculus and state their security properties in terms of coarse-grained notions of protocol equivalence. [...]

2.1 Basics

The pi-calculus is a small but extremely expressive programming language. It is an important result of the search for a calculus that could serve as a foundation for concurrent computation, in the same way in which the lambda calculus is a foundation for sequential computation. Pi-calculus programs are systems of independent, parallel processes that synchronize via message-passing handshakes on named channels. The channels a process knows about determine the communication possibilities of the process. Channels may be restricted, so that only certain processes may communicate on them. [...]

What sets the pi-calculus apart from earlier calculi is that the scope of a restriction—the program text in which a channel may be used—may change during computation. When a process sends a restricted channel as a message to a process outside the scope of the restriction, the scope is said to *extrude*, that is, it enlarges to embrace the process receiving the channel. Processes in the pi-calculus are mobile in the sense that their communication possibilities may change over time; they may learn the names of new channels via scope extrusion. Thus, a channel is a transferable capability for communication. A central technical idea of this paper is to use the restriction operator and scope extrusion from the pi-calculus as a formal model of the possession and communication of secrets, such as cryptographic keys. These features of the pi-calculus are essential in our descriptions of security protocols.

2.2 Outline of the Pi-Calculus

[...] We assume an infinite set of names, to be used for communication channels, and an infinite set of variables. We let m, n, p, q, and r range over *names*, and let z, y, and z range over *variables*. The set of *terms* is defined by the grammar:

$L, M, N ::=$	terms
n	name
(M, N)	pair
$\mathbf{0}$	zero
$suc(M)$	successor
x	variable

In the standard pi-calculus, names are the only terms. For convenience we have added constructs for pairing and numbers, (M, N), 0, and suc(M), and have also distinguished variables from names. The set of *processes* is defined by the grammar:

$P, Q, R ::=$	processes
$\bar{M}\langle N\rangle.P$	output
$M(x).P$	input
$P \mid Q$	composition
$(\nu n)\ P$	restriction
!P	replication
$[M\ is\ N]\ P$	match
$\mathbf{0}$	nil
$let\ (x, y) = M\ in\ P$	pair splitting
$case\ M\ of\ 0 : P\ suc(x) : Q$	integer case

[1] from: Abadi and Gordon. A Calculus for Cryptographic Protocols – The Spi Calculus. ACM, 1997. https://dl.acm.org/doi/pdf/10.1145/266420.266432

In $(\nu n)\ P$, the name n is bound in P. In $M(x).P$, the variable x is bound in P. In *let* $(x, y) = M$ *in* P, the variables x and y are bound in P. In *case* M *of* $0 : P$ *suc*$(x) : Q$, the variable x is bound in the second branch, Q. We write $P[M/x]$ for the outcome of replacing each free occurrence of x in process P with the term M, and identify processes up to renaming of bound variables and names. We adopt the abbreviation $\bar{M}\langle N\rangle.P$ for $\bar{M}\langle N\rangle.P.\mathbf{0}$. [. . .]

Since we added pairs and integers, we have two new process forms:

- A pair splitting process *let* $(x, y) = M$ *in* P behaves as $P[N/x][L/y]$ if term M is the pair (N, L), and otherwise it is stuck.
- An integer case process *case* M *of* $0 : P$ *suc*$(x) : Q$ behaves as P if term M is 0, as $Q[N/x]$ if M is *suc*(N), and otherwise is stuck.

We write $P \simeq Q$ to mean that the behaviors of the processes P and Q are indistinguishable. [. . .]

2.3 A first example

Our first example is extremely basic. In this example, there are two principals A and B that share a channel, c_{AB}; only A and B can send data or listen on this channel. The protocol is simply that A uses c_{AB} for sending a single message M to B. In informal notation, we may write this protocol as follows:

$$\text{Message 1} \quad A \to B : M \quad \text{on } c_{AB}$$

A first pi-calculus description of this protocol is:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle M\rangle \\ B &\triangleq c_{AB}(x).\mathbf{0} \\ \mathit{Inst}(M) &\triangleq (\nu c_{AB}).(A(M) \mid B) \end{aligned}$$

The processes $A(M)$ and B describe the two principals, and $\mathit{Inst}(M)$ describes (one instance of) the whole protocol. The channel c_{AB} is restricted; intuitively, this achieves the effect that only A and B have access to c_{AB}.

In these definitions, $A(M)$ and $\mathit{Inst}(M)$ are processes parameterised by M. More formally, we view A and Inst as functions that map terms to processes, called abstractions, and treat the M's on the left of $\triangleq$ as bound parameters. Abstractions can of course be instantiated (applied); for example, the instantiation $A(0)$ yields $c_{AB}\langle 0\rangle$. The standard rules of substitution govern application, forbidding parameter captures; for example, expanding $\mathit{Inst}(c_{AB})$ would require a renaming of the bound occurrence of c_{AB} in the definition of Inst.
The first pi-calculus description of the protocol may seem a little futile because, according to it, B does nothing with its input. A more useful and general description says that B runs a process F with its input. We revise our definitions as follows:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle M\rangle \\ B &\triangleq c_{AB}(x).F(x) \\ \mathit{Inst}(M) &\triangleq (\nu c_{AB})\ (A(M) \mid B) \end{aligned}$$

Informally, $F(x)$ is simply the result of applying F to x. More formally, F is an abstraction, and $F(x)$ is an instantiation of the abstraction. We adopt the convention that the bound parameters of the protocol (in this case, M, c_{AB}, and x) cannot occur free in F. This protocol has two important properties:

- Authenticity (or integrity): B always applies F to the message M that A sends; an attacker cannot cause B to apply F to some other message.
- Secrecy: The message M cannot be read in transit from A to B: if F does not reveal M, then the whole protocol does not reveal M.

The secrecy property can be stated in terms of equivalences: if $F(M) \simeq F(M')$, for any M, M', then $\mathit{Inst}(M) \simeq \mathit{Inst}(M')$. This means that if $F(M)$ is indistinguishable from $F(M')$, then the protocol with message M is indistinguishable from the protocol with message M'.

[. . .]

(i) In the pi-calculus definition, as described above, the authors include some additional distinctions to the canonical grammar. Name *two* of these additions and briefly explain their intended purpose. (5pts)

Any two of the following examples:

- added constructs for pairings and numbers like (M,N) and 0
- added a successor function that works with numbers
- added matches, pair-splitting and integer case to the process notation
- (obviously) introduced the spi-calculus extension

(ii) The described spi-calculus is built *upon* the basics of the pi-calculus. In particular: (a) What 'feature' of the pi-calculus does the spi-calculus extension rely on? (b) What two properties are gained as such? Briefly describe their purpose. (10 pts)

(a) The 'restricted channel' property $(\nu n)\ P$ for a process P (to model cryptographically secure channel communication).

(b 1) Authenticity: The secure F process always receives the same input parameter as the bound variable of the c_{AB} channel, making sure that the message always gets 'passed on' to the following (secure) process.
and
(b 2) Secrecy: Due to the priority and exclusivity of the c_{AB} channel and the authenticity property, if message M does not get 'leaked' (revealed/sent) from process F, then M is always secret to F.

(iii) The authors clarify the definitions of the spi-calculus running example from their first description in pi-calculus notation. What is the change they made? Give an argument why the protocol makes more sense in the second iteration. (5pts)

Changed from $c_{AB}(x).\mathbf{0}$ to $c_{AB}(x).F(x)$ to signify that the information x received over secure channel c_{AB} then actually gets used in / bound to some process F and only F (authenticity / integrity property).