Modelling Multi-Level Learning in Multi-Agent-Systems with Stochastic Nets-within-Nets

Michael Köhler-Bußmeier[1,†], Lorenzo Capra[2] and Heiko Rölke[3]

[1]Hamburg University of Applied Sciences, Berliner Tor 7, D-20099 Hamburg, Germany
[2]Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, Milano, Italy
[3]University of Applied Science of the Grisons, Pulvermühlestrasse 57, CH-7000 Chur

Abstract

In multi-agent systems (MAS), we have micro elements, i.e., the agents, and macro elements, i.e., the organizational network with roles and behavior rules. Learning occurs at both levels, and the learning processes are intertwined. We like to analyze interaction and learning in MAS. We develop a formal framework rooted in game theory to study the micro-macro learning processes.

While our ground formalism is suitable for analytical purposes, it is not well adjusted to engineer MAS. Therefore, in a second step, we translate micro and macro models into specialized probabilistic Petri nets to bridge the gap between theory and real applications. Our model is inspired by the common MAPE pattern for feedback loops (monitor, analyze, plan, and execute).

Finally, we integrate the multi-level model into one single model. This model is based on the nets-within-nets approach of probabilistic Hornets. Since our ground formalism is probabilistic in nature, it is essential that our Hornet provides probabilistic features. This allows for a seamless translation of the ground formalism to enable formal analysis.

Keywords

nets-within-nets, mape feedback loop, multi-agent systems, micro-macro link, multi-level learning, self-adaptive systems

1. Modelling Learning for Multi-Agent-Interaction

In this paper, we aim to study coordinated interactions in multi-agent systems (MAS) from the perspective of organization-centered multi-agent systems (OCMAS) [1]. Broadly speaking, the problem agents face in MAS is that the local reasoning of agents usually leads to *suboptimal* results due to limitations (restricted knowledge, bounded resources, reasoning complexity, etc. – usually summarized as bounded rationality). Organizations are structures that should restrict and pre-structure agent interaction in such a clever way that the risk of sub-optimality is reduced to an acceptable amount. In this context, learning is a *multi-level* concept: We have *macro-level* learning (where the organization has to choose a 'good' team), *meso-level* learning (where the team has to negotiate a good team strategy), and *micro-level* learning (where the agents choose individual actions in accordance with the team strategy). Moreover, these processes do not occur independently; instead, they are intertwined. This is known as the *micro-macro link* in multi-agent theory [2]. These tight connections of overlapping multi-level processes are challenging for the understanding of alternative approaches and parameterization of learning algorithms.

The objective of the paper is to develop an integrated framework that allows specifying the interaction of multi-level learning and evaluating the impact of different instantiations of the learning mechanisms. Additionally, we would like our formalism to be 'close' to typical development models and diagrams for MAS, like AUML [3].

PNSE'26, International Workshop on Petri Nets and Software Engineering, 2026
*Corresponding author.
michael.koehler-bussmeier@haw-hamburg.de (M. Köhler-Bußmeier); capra@di.unimi.it (L. Capra); heiko.roelke@fhgr.ch (H. Rölke)
https://www.haw-hamburg.de/michael-koehler-bussmeier (M. Köhler-Bußmeier); https://capra.di.unimi.it/ (L. Capra); https://www.fhgr.ch/personen/person/roelke/ (H. Rölke)
0000-0002-3074-4145 (M. Köhler-Bußmeier); 0000-0002-1029-1169 (L. Capra); 0000-0002-9141-0886 (H. Rölke)

The paper has the following structure: Section 2 introduces a probabilistic formalization, strongly grounded in game-theory, of the micro-, meso-, and macro-levels of interaction and learning in MAS. Here, we also define our running example that should exemplarily describe the problems arising in the context of multi-level learning. Here, we will explain that while the micro-level is quite well established and understood (cf. [4]), this does not apply to the meso- or macro-level. Recent approaches include, e.g., coalition theory [5] or Multi-Agent Reinforcement Learning [6], but they have not received the same attention as the micro-level. Even less work is available that studies the emerging effects of the multi-level interdependencies (cf. [2]).

Our work aims to study multi-level learning processes in a uniform way. However, our foundation is highly inspired by game theory and is therefore rather abstract. Henceforth, the gap between the formal model and the specification of MAS applications is rather broad. Therefore, as a second step, we propose Petri nets in Section 3 to bridge the gap. First, we express each level independently by a specialized Petri net model. These models use high-level features like colors but also include probabilistic extensions. We will propose a Petr-net-based approach to model the micro, meso, and macro levels. At the micro-level, we propose agent interaction protocols; at the meso-level, we define team negotiating nets; and at the macro-level, we define feedback-loop nets to express organizational learning. Especially, we aim to provide a general framework in which to express different kinds of feedback loops with various update functions, etc., to test their impact on the learning process.

In Section 4, we will integrate the three different net formalisms. Since learning at a higher level comes with the modification of a lower level net model, we propose to use a Petri net formalism that is capable of self-modification. There are several formalisms proposed that can modify Petri nets, e.g., [7, 8, 9, 10], but the early idea of self-modification of Petri nets can be traced back to the seventies [11]. In this paper, we advocate for nets-within-nets [7] in their algebraic and probabilistic extensions [12], since these two modeling elements allow us to define the core ingredient: a process algebraic XOR choice, extended with probabilities for choices and a logging of recent choices made, together with an update function that reacts to a feedback value. We will demonstrate how micro, meso, and macro-elements are embedded into the nets-within-nets formalism of Hornets using the running example from Section 3. We conclude with an outlook.

2. Our Scenario: Multi-Level OCMAS Learning

In the following, we describe the interplay of the different levels in our multi-level learning scenario within a game theoretical setting. In the following Section 3, we close the gap between the theoretical model and practical application by defining Petri net models.

2.1. Micro-Level: Multi-Agent Interaction with Rewards

Our approach is situated in game theory. Here, we add a utility function u (also called a payoff or reward) to the interaction of agents. Basic example: Alice has a decision between two options: a_1 and a_2; Bob has a decision between two options: b_1 and b_2. We have to add a payoff to these actions.

Let us use the well-known *prisoners' dilemma* from game theory. The two agents, Alice and Bob, are accused of some crime and they have the opportunity to confess (a_1 and b_1) of to deny (a_2 and b_2). In this scenario the agents' payoff is the punishment avoided. So, denying is a kind of cooperative behaviour.

	b_1	b_2
a_1	$(3,3)$	$(0,5)$
a_2	$(5,0)$	$(1,1)$

Since there is no incriminating evidence, it is best for them to choose (a_1, b_1), i.e., when both agents deny; in this case, they receive only a small punishment for some other minor crimes. This choice, called 'cooperative', leads to the highest payoff of $(3,3)$. (Cooperation from the criminals' point of view.) However, if exactly one of them is 'uncooperative' and confesses: (a_1, b_2) or (a_2, b_1), this traitor goes along without punishment , and only the other one is severely punished. If both confess: (a_2, b_2) both are punished, but the punishment is milder and equally distributed.

It is common practice to describe the agents' strategies using probabilistic distributions.[1] When considering strategies as probabilistic distributions, we consider the *expected* payoff $E[U]$ of the game, given the strategies.

Let $\mathbb{P}(X)$ be the set of all probability distributions over a set X. A **(local) strategy** for Alice is a probability distribution over the action set $A = \{a_1, a_2\}$. A strategy for Alice might be, e.g., the distribution:

$$s_{\text{Alice}} = [a_1 : p, \quad a_2 : (1-p)]$$

meaning that Alice chooses a_1 with a fixed probability of p. Analogously, Bob might have the strategy:

$$s_{\text{Bob}} = [b_1 : q, \quad b_2 : (1-q)]$$

Since actions are chosen independently, the outcome of the game is obtained by the *product rule*; i.e., (a_1, b_1) is chosen with a probability of $p \cdot q$. This independence describes the *non-cooperative* nature of the setting.

The prisoners' dilemma is well-known in (non-cooperative) game-theory [5] since all the fundamental reasoning concepts, including Nash-equilibria, lead to the *worst* outcome (a_2, b_2) with a payoff of only $(1, 1)$. This problem stems from the fact that agents act locally, and from a local point of view, Alice is better off when choosing a_2 (or switching from a_1 to a_2)–independently of what Bob does. Assuming that Alice and Bob both apply this kind of reasoning, we arrive at the Nash-equilibrium (a_2, b_2) with a payoff of $(1, 1)$. From a more global point of view, they both obtain a total payoff (also called the social welfare of the game) of $1 + 1 = 2$, which is rather low compared to the maximum of $3 + 3 = 6$. The ratio of the payoff in the Nash-equilibrium to the maximum expresses the price for Alice and Bob of not having the possibility to coordinate their actions. It is called the **price of anarchy** (PoA). The price-of-anarchy quantifies the need for coordination in a game, i.e., some kind of multi-agent interaction. Therefore, the agents can use the PoA as a signal that stimulates a coordination process.

The problem is that if we go beyond non-cooperative game theory, agents decisions become more complex. To describe cooperation in *cooperative game theory*, we allow a joint probability; i.e., a probability distribution for *joint* actions. For the prisoners' dilemma, we might have a strategy that emphasizes cooperation (a_1, b_1), e.g., we generate a *coalition* with a **joined strategy**:

$$s_{\text{joined}} = [(a_1, b_1) : 0.75,\ (a_1, b_2) : 0.1,\ (a_2, b_1) : 0.1,\ (a_2, b_2) : 0.05]$$

From a complexity point of view, a strategy $s \in \mathbb{P}(A_a) \times \mathbb{P}(A_b)$ (non-cooperative game theory) is linear in size, while $s' \in \mathbb{P}(A_a \times A_b)$ (cooperative game theory) grows quadratically. The exponential blow-up in joined actions discourages the use of cooperation and motivates interest in non-cooperative game theory.

2.2. Meso-Level: Joined Action Learning in MAS

A more complex game, i.e., a complex multi-agent interaction, could be easily modeled as a Petri net, like a multi-party workflow [13]. For our purposes, we will restrict ourselves to workflows having a very simple block-structure: They are constructed from atomic actions a, concurrent execution $(x \| y)$ (AND), sequences $(x; y)$, and exclusive choice $(x + y)$ (XOR). We like to extend the choice operator with probabilities [14, 15, 16, 17, 18]. We will use the following interaction as our running example.

Example Let us extend the example where Alice and Bob now have two decisions each. Alice has two **decision points** a and c, where $A_a = \{a_1, a_2\}$ and $A_c = \{c_1, c_2\}$. Bob has two decision points b and d, where $A_b = \{b_1, b_2\}$ and $A_d = \{d_1, d_2\}$. This structure is illustrated in Fig. 1. Note, that any strategy, either local or joined, turns this Petri net into a net with probabilistic choices.

[1] Probabilities are an abstraction from any 'real' decision logic. In real MAS applications, the agents also have to choose concrete actions, which usually depend on a more detailed state of the agent that we also abstract away from. Probabilities are also useful whenever agents do not want to reveal their logic in detail.

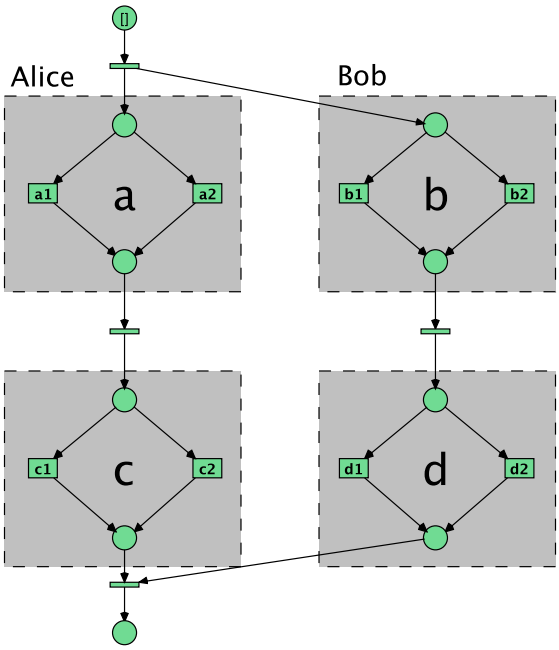

Figure 1: An Agent Interaction Protocol (AIP) for Alice (left) and Bob (right) with four decisions, which has the block-structure $(a;c)\|(b;d)$

In classical non-cooperative game theory there is an asymmetry among these choices as Alice would remember her first action when taking the second. Therefore, she has the action set $A_{\text{Alice}} = A_a \times A_c$ and chooses a strategy from $\mathbb{P}(A_a \times A_c)$. Analogously, Bob chooses from $\mathbb{P}(A_b \times A_d)$. However there is no 'horizontal' relationship. The interaction is described by a strategy s from:

$$\underbrace{\mathbb{P}(A_a \times A_c)}_{\text{Alice}} \times \underbrace{\mathbb{P}(A_b \times A_d)}_{\text{Bob}}$$

Whenever, agents do not remember their prior actions they choose their distributions independently. So, Alice chooses a strategy from $\mathbb{P}(A_a) \times \mathbb{P}(A_c)$ – instead of $\mathbb{P}(A_a \times A_c)$. In this case the game is described by the much simpler strategy space, where no decision is correlated with any other:

$$S_\perp := \mathbb{P}(A_a) \times \mathbb{P}(A_c) \times \mathbb{P}(A_b) \times \mathbb{P}(A_d)$$

On the opposite we could imagine a completely coordinated game. So Alice and Bob do no longer choose their actions independently: instead they will coordinate them.[2] Then the strategy space is based on the complete Cartesian product:

$$S_\top := \mathbb{P}(A_a \times A_c \times A_b \times A_d)$$

In general we will consider a space somewhere in between these two extremes.

Let us assume temporarily – and only for illustrational reasons – that the structure of the utility function is known. (However, we do not want to rely on this assumption in the following since usually utilities are given in implicit form for our MAS.) Assume that the payoff u depends on the first two decisions a and b combined with the decisions c and d. The whole payoff is then

$$u(a,b,c,d) = u_1(a,b) + u_2(c,d). \tag{1}$$

In this case there is no incentive for Alice to correlate her decisions a and c; and similarly for Bob with b and d. Instead both have to coordinate their first actions a and b and then their second actions c and

[2] In a real MAS this coordination would be specified by a protocol on its own. Here it is simply expressed in the structure of the distribution.

d. So, we are interested in the strategy space that correlates "horizontally" (inter-agent) rather than "vertically" (intra agent):

$$S_1 := \mathbb{P}(A_a \times A_b) \times \mathbb{P}(A_c \times A_d) \tag{2}$$

One example for such a **team strategy** $s \in S_1$ is (with $0 \leqslant p, q \leqslant 1$):

$$s_{p,q} \quad = \quad \Big(\begin{array}{llll} [(a_1, b_1) : p, & (a_1, b_2) : 0, & (a_2, b_1) : 0, & (a_2, b_2) : (1-p)], \\ [(c_1, d_1) : q, & (c_1, d_2) : 0, & (c_2, d_1) : 0, & (c_2, d_2) : (1-q)] \Big) \end{array} \tag{3}$$

Here, the organization defines the probability p to coordinate on (a_1, b_1) and $(1-p)$ for (a_2, b_2), while the mixed actions (a_1, b_2) and (a_2, b_1) have a probability of 0. (In the following we will omit elements that have a probability of 0.) Analogously, probability q relates the decision points c and d.

We assume that each strategy is chosen with a certain probability, i.e., we are considering distributions $\tau \in \mathbb{P}(S)$ over S, which are called a **team structure** in the following.

Example[cntd.] For the fixed space S_1 from above, a team structure τ assigns probabilities to joined team strategies. For example:

$$\tau_1 = [s_{0.9,0.9} : 0.45,\ s_{0.9,0.1} : 0.25,\ s_{0.1,0.9} : 0.10,\ s_{0.1,0.1} : 0.20] \tag{4}$$

Here, the team prescribes either a high (here: 90%) probability to coordinate on (a_1, b_1) or to coordinate on (a_2, b_2). Analogously for the decision points c and d. An example with a smaller support is $\tau_2 = [s_{0.9,0.9} : 0.85,\ s_{0.1,0.1} : 0.15]$.

Intuitively, sampling a joined team strategy s from a team structure τ could be considered a **negotiation** process among the team members.[3]

If we assume a fixed strategy space S, then *meso-level learning* means modifying the probabilities $\tau(s)$ for the strategies s. There are different approaches to this meso-level learning in MAS, which directly evolve from game theory, such as Nash-Q-Learning [5], or those that generalize Reinforcement Learning [4], such as MARL [6].

2.3. Macro-Level: Learning Organisational Structures in OCMAS

It is common to augment the multi-agent system with an organizational structure that contains the formal aspects of the multi-agent system, including the position network, roles, norms, etc. This approach is known as *Organisation-Centric MAS* (OCMAS) [1]. Up to now, we have assumed a given strategy space S. Now, we are interested in the selection process of an appropriate space, which is occurring at the organizational macro-level. An organization must balance between two extremes: A huge space S has the benefit of allowing for highly joint actions, which usually increases the expected payoff. The prisoners' dilemma is the classical example in game theory, demonstrating that agents pay a high price for not having the ability to coordinate their actions. On the other hand, learning becomes harder in larger spaces since the number of joined actions (i.e., a Cartesian product) grows exponentially with the number of joined decisions.

> Problem: We have to learn an appropriate strategy space S, but the set of possible spaces $\mathcal{S}$ cannot be evaluated beforehand – even in cases the environment, modeled by the utility u, is static.

Therefore, we need a learning mechanism to identify an acceptable space. Generating a space essentially means clustering the decision points. This has much in common with **coalition theory** of cooperative game-theory [5], with the main difference being that a coalition is a group of agents, while we are clustering decisions within a single agent as well as among different agents.

[3] In this work, we assume a fixed team τ for a given space S. In future work, we will sample $\tau = \tau^{(1)}$ s from a distribution $\tau^{(2)} \in \mathbb{P}^2(S) = \mathbb{P}(\mathbb{P}(S))$. This sampling is extended until a fixed distribution $\tau^{(k)}$. Then, learning will also modify these distributions $\tau^{(i)}$.

We are using the same approach again and assume that all spaces S from the set of all spaces $\mathcal{S}$ are available in principle, but they are chosen with a probability; i.e., we are considering distributions from $\mathbb{P}(\mathcal{S})$. Of course, organizations could modify the current distribution, which is considered organizational learning.

Learning which S is a promising candidate (and should have a high probability) occurs at the organizational side of the systems; it is independent of the individual learning of the agents. Intuitively, this learning could be considered a **team formation** process.

Example[cntd.] For the utility function $u(a,b,c,d) = u_1(a,b) + u_2(c,d)$, the space $S_1 := \mathbb{P}(A_a \times A_b) \times \mathbb{P}(A_c \times A_d)$ seems to be a promising (i.e., high probability) candidate, as it relates those actions that are also related by the utility u. However, S_1 may already be too complex, in the sense that we certainly have the opportunity to find the optimal strategy in this space, but we may waste so much time searching for it that we will never be able to cover the exploration expenses.[4] In this case, we may have to trade a speed-up in learning for a loss of utility and reduce the complexity of the strategy space. This might be the case, e.g., for $u_1(a,b) \ll u_2(c,d)$. In this case

$$S_2 := \mathbb{P}(A_a) \times \mathbb{P}(A_b) \times \mathbb{P}(A_c \times A_d) \tag{5}$$

It might be a good compromise as the exploration is cheaper, even if the expected outcome is less than the optimal one. An example of a team strategy $s \in S_2$ is ($0 \leqslant p, p', q \leqslant 1$):

$$s_{p,p',q} := \big([a_1{:}\,p, a_2 : (1-p)], [b_1{:}\,p', b_2{:}\,(1-p')], [(c_1,d_1){:}\,q, (c_2,d_2){:}\,(1-q)]\big) \tag{6}$$

Analogously, whenever the environment changes such that we add a third addend $u_3(a,d)$ to u, making Alice's first action correlated with Bob's second one:

$$u'(a,b,c,d) = u_1(a,b) + u_2(c,d) + u_3(a,d) \tag{7}$$

In this case, all decisions are correlated, and only an in-depth analysis can show where a good compromise between exploration and exploitation lies.

So, on the one hand, we have to split the set of decision points into smaller subsets. On the other hand, we observe from the example that we have to regroup them later due to environmental change. To retain flexibility, we introduce an inner structure in the partitioning, i.e., we introduce a *splitting process* with early and late splits. From a formal point of view, this means that our strategy space is not "flat" like in:

$$S = \mathbb{P}(A_a \times A_b) \times \mathbb{P}(A_c) \times \mathbb{P}(A_d)$$

Instead, it may have some inner (i.e., recursive) structure deriving from the generation process. Assume that we are confident that decision d is independent from the rest, so we first split into $A_a \times A_b \times A_c$ and A_d. Then we have several options, and one of them is to separate c, and we split $A_a \times A_b \times A_c$ into $A_a \times A_b$ and A_c at the second stage. So, we have to think about the inner team structure, which is a tree, to describe a splitting process for the set of decision points (we abbreviate sets like $\{a,b,c,d\}$ as $abcd$):

$$abcd\big[abc\big[ab[], c[]\big], d[]\big]$$

The leaves of this tree define the clustering, but the inner nodes describe splitting decisions. Roughly speaking, nodes of lesser depth describe clustering decisions about which we are more certain. This recursive structure is needed for another reason, too. In our example, we considered a multi-agent system (MAS) with only one kind of interaction; however, in general, a MAS has more than one kind, and there is some relationship between them, e.g., due to shared resources.[5]

[4] This conflict *between exploration and exploitation* in learning is specific to *online* learning since the agents learn while they are acting – opposed to off-line learning, where training phases are separated from acting. In an online learning scenario, which is conceptually based on some kind of feedback-loop, we no longer have clearly separated phases; instead, learning and acting are intertwined.

[5] In theory, one could integrate all interactions into a single one, but this has the obvious disadvantage of destroying separation

3. Petri Net Models for OCMAS Learning

In OCMAS, agents act within given structures. In our model, we call this structure a **team**. One major task is to construct the strategy space S and a team $\tau \in \mathbb{P}(S)$, which serves as the coordination context for the agents. This process is called **team-formation** in SONAR [19]. From the current team τ, the agents sample a team strategy s. This process is called **team-negotiation**.

In the following, we aim to bridge the gap between our formal setting and the practical use of our OCMAS framework SONAR by introducing net models for each level of the learning process. Here, we consider each level independently. The following Section 4 will integrate them. The Petri net models will still be probabilistic (allowing for a closer connection to our foundational model from the previous Section), but it is quite straightforward to replace the probabilistic aspects with interaction protocols.

3.1. Probabilistic Strategy Spaces

Team-formation generates a strategy space, where each space has a certain probability. As already explained, a space has a recursive structure over subsets of action sets. So, we have to think about the inner team structure, which is a special tree. (We will specify the generation process of trees by Petri nets in Sect. 3.2.) The nodes are subsets of decisions $C \subseteq DCP = \{d_1, \dots, d_n\}$; the root node is $C_0 = DCP$, and the children of a node C form a partition of C. The leaves $\mathcal{C} = \{C_1, \dots, C_k\}$ of such a partitioning tree are a **partition** of the set of decision points DCP. Each leaf $C \in \mathcal{C}$ is a set of decisions that we would like to correlate, thus generating the correlated action set:

$$A_C := \bigtimes_{d \in C} A_d \tag{8}$$

The leaves $\mathcal{C} = \{C_1, \dots, C_k\}$ generate a **strategy space** $S_{\mathcal{C}}$:

$$S_{\{C_1,\dots,C_k\}} := \mathbb{P}(A_{C_1}) \times \cdots \times \mathbb{P}(A_{C_k}) \tag{9}$$

The actions from A_C are chosen by a strategy $Pr_C \in \mathbb{P}(A_C)$. The clustering described by the Cartesian product $S_{\mathcal{C}}$ has the consequence that we obtain the probability for the decision points $\vec{d} = (d_1, \dots, d_n)$ – called the **team strategy** – by the product rule of probability:

$$s_{\mathcal{C}}(\vec{d}) := Pr_{\mathcal{C}}(\vec{d}) = \prod_{C \in \mathcal{C}} Pr_C\big(\pi_C(\vec{d})\big) \tag{10}$$

Here $\pi_C(\vec{d})$ denotes the projection onto the sub-tuple corresponding to C.

Example Given the example tree $abcd\big[abc\big[ac[], b[]\big], d[]\big]$ we obtain the partition $\mathcal{C} = \{C_1, C_2, C_3\}$ of DCP into $C_1 = \{a, c\}$, $C_2 = \{b\}$, and $C_3 = \{d\}$, and the space

$$S_{\mathcal{C}} = \mathbb{P}(A_a \times A_c) \times \mathbb{P}(A_b) \times \mathbb{P}(A_d),$$

which introduces the product form:

$$s_{\mathcal{C}}(\vec{d}) = Pr_{\mathcal{C}}(a, b, c, d) = Pr_{C_1}(a, c) \cdot Pr_{C_2}(b) \cdot Pr_{C_3}(d)$$

Note that the team strategy is not completely determined since we are still free in the choice of the strategies $Pr_{C_1}(a, c)$, $Pr_{C_2}(b)$, and $Pr_{C_3}(d)$. So, only the product form is introduced.

---

and modularity. Assuming that the system consists of many interactions, we obtain a hierarchy of processes: For each interaction, the agents coordinate independently in teams. However, since the coordination processes involve this learning process, it again represents some kind of uncorrelated action, and we obtain a second-order price of anarchy: the PoA for the coordination process. This 2^{nd}-order PoA stimulates some 2^{nd}-order coordination, which generates a third-order PoA, and so on.

3.2. Partition Nets

In the following, we specify these trees using Petri nets, which are essentially trees with two kinds of nodes: And-nodes and Or-nodes. Or-nodes define alternatives. The idea is that the net describes all possible trees, and the token game describes a team formation process. These and-or graphs can easily be denoted as PNs.

Assume a given AIP (Agent Interaction Protocol, like that shown in Fig. 1) with the set DCP of decision points. In our model, the places are subsets of DCP. Each transition $t \in \mathcal{T}_1$ describes a partition of C into the sets $C_1, \ldots, C_n$. At each place, we have the possibility of stopping the partitioning of the decision set. To express this, each place $C \in P$ is connected to an additional stop-transition $(C, \varnothing)$ from the set $\mathcal{T}_2$:

$$\begin{array}{lcl} \mathcal{P} & = & \{C \mid C \subseteq DCP\} \\ \mathcal{T} & = & \mathcal{T}_1 \cup \mathcal{T}_2 \\ \mathcal{T}_1 & = & \{(C, \{C_1, \ldots, C_n\}) \mid \ C_1, \ldots, C_n, n > 1 \text{ are a partition of } C \in \mathcal{P}\} \\ \mathcal{T}_2 & = & \{(C, \varnothing) \mid C \in \mathcal{P}\} \\ \mathcal{F} & = & \{(x,t),(t,y) \mid t = (x,Y) \in \mathcal{T}, y \in Y\} \end{array} \tag{11}$$

Each subset $T_1 \subseteq \mathcal{T}_1$ induces the net $N = (P, T, \mathcal{F} \cap (P \cup T)^2)$ where:

$$\begin{array}{lcl} P & = & \{DCP\} \cup \{C, C_1, \ldots, C_n \mid (C, \{C_1, \ldots, C_n\}) \in T_1\} \\ T & = & T_1 \cup \{(C, \varnothing) \mid C \in P\} \end{array}$$

Definition 1. *Assume a given AIP with a set DCP of decision points. N is called **Partition Net** (for DCP) whenever it is induced by a set T and all places are F^*-reachable from the place $p_0 = DCP$.*

For partition nets, transitions have exactly one place in the preset; i.e., we have BPP nets [20]. This is a subclass of free-choice nets, and all conflicts are structural. Since partitions form a natural order on transitions, each partition net is acyclic and finite.

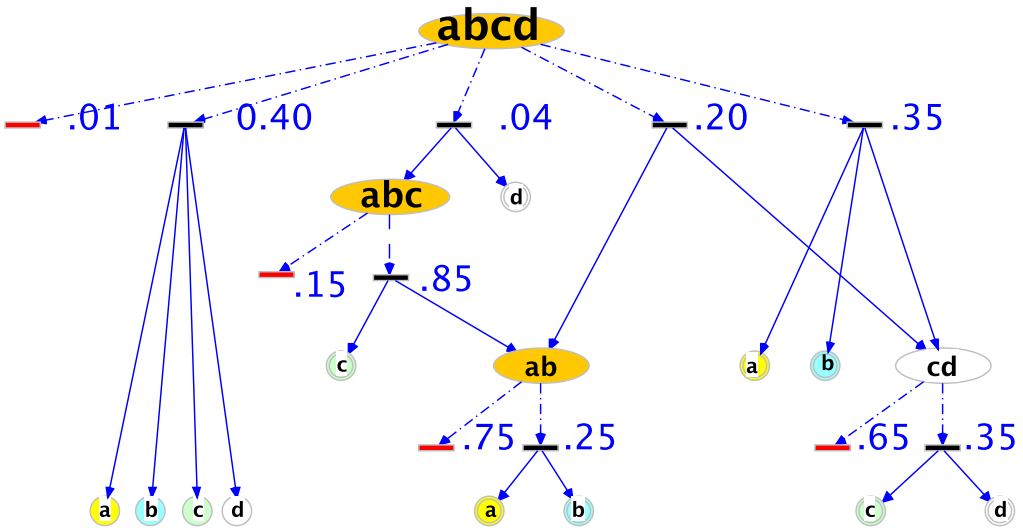

Figure 2: A Probabilistic Team Net

An example is given in Fig. 2. (Here, we duplicated the graphical representation of some places to avoid too many crossing arcs. We also omit stop transitions at singleton sets.) Ignore the probabilistic annotations at this point. Transition names are omitted in the graphical representations as they are implicitly given by the pre- and postsets.

3.3. Team Nets: Formation of Teams

In general, team-formation is not a one-shot process, but an iterative one. The token game is able to capture this process. In a process the 'stop' transitions that fire indicate the 'leaves' of the tree. Let $N = (P, T, F)$ be a Partition Net. The canonical initial marking m_0 marks only the place DCP with one token: $m_0(p) = \mathbf{1}[p = DCP]$. We like to assign probabilities to alternative partitions (or stopping) at each subset C. The function $\lambda : T \to \mathbb{R}^+$ assign weights to transitions. We define the probability of selecting transition t at the place C to be proportional to its weight compared to all other transitions in marking m:

$$Prob(t) := \frac{\lambda(t)}{\sum_{t' \in (\bullet t)^\bullet} \lambda(t')} \tag{12}$$

Especially, the probability of stopping the splitting of C is proportional to the weight of the stop transition $(C, \varnothing)$. As a consequence, whenever the net does not contain any split transition then the stop transition has probability 100%.

We annotated the augmented partition net of Fig. 2 with probabilities to obtain a **team net**. Here we show only transitions with non-zero probability. A process of the net generates a random sample, i.e. one strategy space. The net itself introduces a stochastic distribution on strategy spaces. For the team net given in Fig. 2 the team $abcd\big[abc[], d\big]$ occurs with probability $0.04 \cdot 0.15$, the team $abcd\big[abc[ab[], c], d\big]$ occurs with probability $0.04 \cdot 0.85 \cdot 0.75$, etc.

Note, that we allow situations where we have sub-branches that occur with a probability of zero. We do not want to prune the net since the probabilities are updated and may become reachable during the execution.

3.4. Negotiation Nets: Negotiating a Joined Strategy

Once the team formation process has generated a strategy space, like S_1, we have the distribution over team strategies. In a simulation we would then sample a concrete team strategy $s \in S_1$. Every strategy space S corresponds to a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of the set of decision points of DCP. The partition generates the team action probability $Pr_{\mathcal{C}}(\vec{d})$ for the decisions $\vec{d}$ using (10). The negotiation process assigns a concrete function Pr_C to each C, and we obtain team strategies like $s_{p,q}$ from (3).

Analogous to the splitting net, we allow consideration of weighted alternatives for each $Pr_C(\pi_C(\vec{d}))$ and use a feed-back mechanism on weights to learn which one maximizes the expected game payoff. This net is called the **negotiation net**. (Since it is technically the same idea as we have for the team net we omit an example here.)

3.5. Agent Interaction Protocols

An agent interaction protocol (AIP) is a block-structured workflow net. The block structure is obtained from atomic actions a, concurrent execution $(x \| y)$ (AND), sequences$(x; y)$, and exclusive choice $(x + y)$ (XOR). Each atomic action is executed by agents. Therefore, we assign each action to a role. In our MAS Sonar [19] we instantiate these AIP and assign an agent to each role.

The block-structure of the AIP from Fig. 1 is $(a; c) \| (b; d)$, with the choices $a = (a_1 + a_2)$, $b = (b_1 + b_2)$, $c = (c_1 + c_2)$, and $d = (d_1 + d_2)$. The set of choices is $DCP = \{a, b, c, d\}$. Each choice is equipped with a probability. For workflow nets, especially in the context of process mining, probability is an obvious extension since event logs naturally provide information about frequencies [14]. Since these choices are marking dependent, we call this block an **adapt-or** block [15]. As explained above, the probability is obtained from the team-formation and negotiation process, which generates a clustering $\mathcal{C}$, and from this, we obtain $Pr_{\mathcal{C}}(a, b, c, d)$. Probabilistic choices generate a probabilistic Petri net in the usual way, i.e., we have probabilities on runs and extend the reachability graph into a *discrete time Markov chain* (DTMC). Since we have a Markov chain with a payoff, our setting has much in common with Markov reward models [21].

3.6. OCMAS-Learning in Feedback Loops

In our theoretical setting, learning is modeled as a transformation of a probability distribution. In more practical scenarios, these updates are part of a feedback loop pattern. Maybe the most prominent one is known as the MAPE-K loop (monitor-analyze-plan-execute with knowledge) [22, 23]. The software engineering perspective based on feedback loops is increasingly combined with machine learning [24].

Here, we make the feedback-loop an explicit part of our multi-level learning process. All branches of choices – in the AIP, but also in the team formation net and in the negotiation net – are assigned weights. How these weights are translated into probabilities is defined by a function $w2p_{dp}$ (weights to probabilities) for each decision point dp. These functions are a parameter of our model. The probability of selecting the i-th option at the decision point is $Pr[DP = i] = w2p_{dp}(i, w_1, \ldots, w_k)$.

We have several possibilities to explore how these weights translate into probabilities, among others:

- Probabilities are directly proportional to weights: $Pr[DP = i] = w_i / \sum_j w_j$.
- Application specific definition: In our splitting scenario, it is, e.g., useful to have a dependency on the decision point dp itself, more precisely, on its cardinality, since a larger set describes more effort in the negotiation process. Therefore, we would like to dampen the probability of choosing the 'stop' option at a splitting choice. In this case, we enforce that large sets are split into smaller subsets, resulting in lower negotiation costs.
- ϵ-greedy selection: With probability $1 - \epsilon$, we take the action with a probability $w2p_{dp}(i, w_1, \ldots, w_k)$, but with probability ϵ, we take any option with equal probability.

In reinforcement learning [4], ϵ-greedy is a commonly used technique since it exploits the best options most of the time (assuming that the current weights approximate the so-called Q-value of the choices). However, since we cannot be sure about this, we allow ourselves, with probability ϵ, to explore the options. The parameter ϵ balances the trade-off between exploration and exploitation. In fact, the strategy is a sequence of choices; first, we choose between exploration and exploitation, and second, we choose one option i, where the distribution depends on the first stage. Note that the selection is directly proportional to w_i, which is common in probabilistic automata or Petri nets. This is the special case where $\epsilon = 0$ (no exploration).

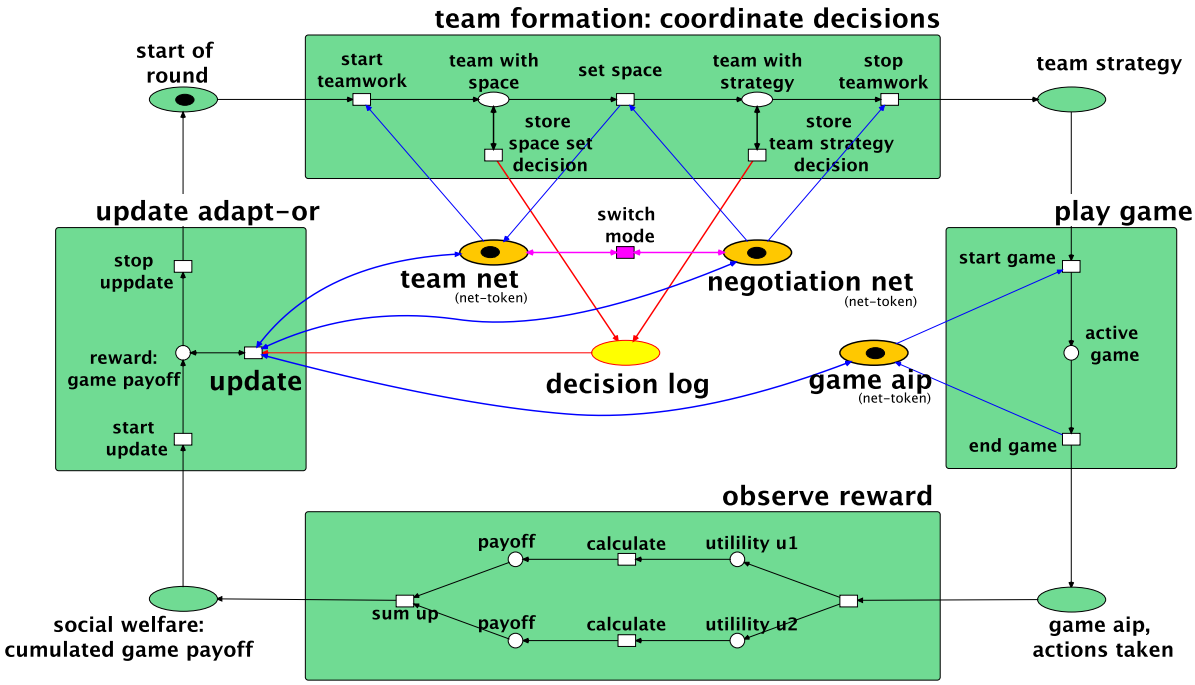

Figure 3: The system net models the feedback-loop. Note, that the tokens on the places *team net* (macro), *negotiation net* (meso), and *game aip* (micro) are in fact net-tokens with an inner structure and marking.

Figure 3 shows the general net structure of the feedback-loop. (The implemented high-level net variant is discussed in the following section.) The loop tracks the decisions made during team-formation and negotiation. Whenever the environment sends us a **reward signal** r (which happens after the execution of the AIP in our setting), we update the weights of a decision point dp according to logged decisions using a function $updateWeights_{dp}(\vec{w}, i, r)$. Whenever the weight $w_i(t)$ and the reward r for the choice i are given on the scale (i.e., they are normalized), then *temporal difference (TD) learning* [4] is the 'classical' option where w_i is updated proportionally to the difference $(r - w_i(t))$:

$$w_i(t+1) = w_i(t) + \alpha \cdot (r - w_i(t)) \quad \text{for } 0 < \alpha < 1 \tag{13}$$

Here, we use the cumulated reward, i.e., the sum of all individual payoffs, as a reward signal since the cumulated reward (also known as social welfare) indicates the price-of-anarchy. Any reasonable update rule would punish decisions with low social welfare relative to those with high social welfare.

Note that especially the parameters $w2p$ and $updateWeights$ are part of the HORNET marking and could be modified. As a special feature, we allow replacing the function $w2p$, which models a kind of adaptation via mode switching [25].

4. Modelling OCMAS Learning with Nets-within-Nets

To analyse our of our multi-level learning model we integrate the different levels into one single *probabilistic HORNET model* [12]. The nesting of nets-as-tokens is very useful for expressing the learning hierarchy: macro-meso-micro.

Let us return to our example illustrated in Fig. 1. where Alice and Bob play a game with four decision points a, b, c and d. The HORNET model consists of a system-net that models the feedback-loop which collects all decisions made and update weights according to the reward signal. The feedback loop carries Petri nets as tokens. We have a net-token for the AIP, one for the team formation, and another one for the negotiation. We assume that we have a (weighted by γ_1 and γ_2) payoff depending on 'horizontal' choices:

$$u'(a, b, c, d) = \gamma_1 \cdot u_1(a, b) + \gamma_2 \cdot u_2(c, d)$$

At the beginning of our simulation, the second stage has a much higher impact; i.e., we set $\gamma_1 = 0.1$ and $\gamma_2 = 0.9$. Therefore, we assume that a clustering $\mathcal{C}_1 = \{C_{11}, C_{12}, C_{13}\}$ of $C_{11} = \{a\}$, $C_{12} = \{b\}$, and $C_{13} = \{c, d\}$ is a good candidate; i.e., we do not correlate a and b to safe exploration costs.

After a given time st (switching time), both horizontal choices will have equal impact, i.e., $\gamma_1 = \gamma_2 = 0.5$. In this case, we can assume that we may introduce a correlation of a and b and should give the clustering $\mathcal{C}_2 = \{C_{21}, C_{22}\}$ into $C_{21} = \{a, b\}$ and $C_{22} = \{c, d\}$ a try. So, we assume that $Pr_t(\mathcal{C}_1)$ will decrease for $t > st$ and therefore $Pr_t(\mathcal{C}_2) = 1 - Pr_t(\mathcal{C}_1)$ increases.

For our example, we have chosen the utility functions u_1 and u_2 in such a way that they reward a coordinated choice; i.e., we play a coordination game. Here, we have also chosen the well-known **battle-of-sexes** scenario of game theory where, again, Alice and Bob choose between two actions, labeled a and b.

They gain a positive reward if their choices align; otherwise, they do not receive any. However, Alice prefers the action a and Bob b. Assuming the reward for the preferred choice is three times more than the alternative, we obtain the following payoff matrix:

	a	b
a	$(3, 1)$	$(0, 0)$
b	$(0, 0)$	$(1, 3)$

For the team formation with respect to $\mathcal{C}_1$, we consider the strategies $s_{p,p'q}$ from (6) for $p, p', q \in \{0.1, 0.9\}$ and sample them with equal probability. for $\mathcal{C}_2$, we choose $s_{p,q}$ from (3), where $p, q \in \{0.1, 0.9\}$. These team strategies prescribe either a high (here: 90%) probability of coordinating on Alice's desired outcome (c_1, d_1) or on Bob's desired outcome (c_2, d_2). For $\mathcal{C}_1$, decisions a and b are unrelated, but correlated for $\mathcal{C}_2$.

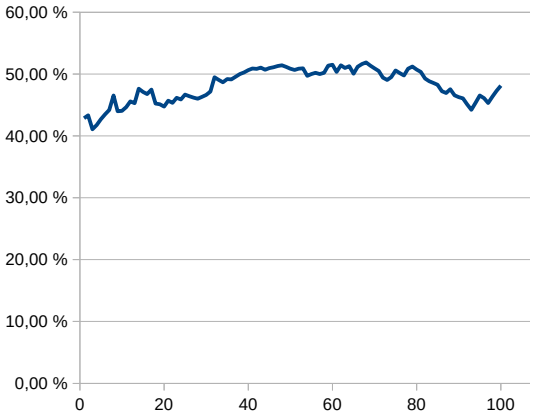

Figure 4: Dynamics of the Generation Probability $Pr_t(S_{\mathcal{C}_1})$ of the Space $S_{\mathcal{C}_1}$ over Time t

We implemented the HORNET model using our tool RENEW [26].[6] In our preliminary analysis, we carried out discrete-event simulation runs on this model until time $t = 100$, where each time step is a complete round in the feedback-loop. Switching from $\gamma_1 = 0.1$ and $\gamma_2 = 0.9$ to $\gamma_1 = \gamma_2 = 0.5$ occurs in the middle of the simulation at time $st = 50$.

We expect the following behavior: Whenever we have a correlation of decision points a and b by clustering, we obtain the maximal payoff of $1 + 3 = 4$ with a very high probability (here: 90%); analogously for c and d. If we do not have such a correlation, then all choices are equally likely, and coordination occurs only in $\approx 50\%$ of the cases, which results in an expected payoff of ≈ 2. In the first phase, the coordination of c and d matters most. In the second phase, the coordination of a and b, as well as of c and d, matters. The weighted payoff drops from approximately 4 to $0.5 \cdot 2 + 0.5 \cdot 4 = 3$. The initial choice of the clustering probability is set proportional to these expected payoffs; we start with a selection probability for the splitting $Pr_0(\mathcal{C}_1) = 3/(3 + 4) = 0.43...$

The simulation dynamics of $Pr_t(\mathcal{C}_1)$ is shown in Figure 4. One can see that in the first phase of the simulation (i.e., before the switching) the probability $Pr_t(\mathcal{C}_1)$ increases, as expected. But, also as expected, after the switching at time st, the correlation of a and b becomes more important, and thus $Pr_t(\mathcal{C}_1)$ starts to decrease. Obviously, to obtain reliable results we would need more runs.

Note, however, in this paper we are not interested in concrete results and would like to demonstrate the general approach only.

5. Conclusion

In this paper, we studied the multi-level learning of OCMAS: We have micro-level learning of the agents (*Adapt the probabilities of a good strategy!*), meso-level learning of the teams (*Choose a good team strategy from a given strategy space!*), and macro-level learning (*Choose a good strategy space!*). As a challenge, we have that these processes do not occur independently. Instead, they are intertwined: Learning means finding a good strategy s within a good strategy space S that is dynamic, too.

To bridge the gap between our formal model and the design of MAS applications, we introduced Petri net models for each level. Then, we integrated these different net models within the formalism of HORNETS. The benefit of this net model is twofold: Firstly, these models are still probabilistic, but it is quite clear how to replace the probabilistic sampling of decisions with an interaction protocol among the agents. Therefore, these Petri nets help to narrow the conceptual gap between the abstract model and an application specification. Secondly, we are able to reason about the impact of different update regimes, feedback functions, etc., with respect to the learning process to allow for good system design. Especially regarding the last aspect, we will rely on our MAUDE representation of probabilistic HORNETS [27, 28], and we will further exploit the symbolic analysis features of MAUDE.

[6] Sources and logs are available online: github.com/koehler-bussmeier/mapeloop.

Generally, we are interested in the impact of our meta-parameters; i.e., we vary, e.g., the learning rate α used in TD-learning and observe how the learning quality changes. Naturally, the example above relies on simulation to demonstrate the method. Nonetheless, it reveals an interesting pattern. Consequently, as part of our ongoing research, we intend to conduct a more comprehensive quantitative analysis of the underlying discrete-time Markov chain (DTMC) induced by the MAUDE-based encoding of the example as an algebraic Petri net.

In our approach, we assume a given reward model and aim to learn appropriate coordination among the many decisions. In future work, we will consider this the other way around: We observe the behavior of the system, deduce the current mode of cooperation between the agents, and calculate the underlying reward function from that. This could lead to a new aspect in process mining that is linked to the mining of nets-within-nets [29].

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] V. Dignum, J. Padget, Multiagent organizations, in: G. Weiss (Ed.), Multiagent Systems, 2nd ed., Intelligent Robotics; Autonomous Agents Series, MIT Press, 2013, pp. 51–98.

[2] M. Schillo, K. Fischer, C. Klein, The micro-macro link in DAI and sociology, in: S. Moss, P. Davidsson (Eds.), Second International Workshop on Multi-Agent Based Simulation, volume 1979 of *Lecture Notes in Computer Science*, Springer-Verlag, 2000, pp. 133–148.

[3] B. Bauer, J. Odell, H. v. D. Parunak, Extending UML for Agents, in: Proceeding of Agent-Oriented Information Systems Workshop, 2000, pp. 3 – 17.

[4] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 2018.

[5] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, New York, 2008.

[6] S. V. Albrecht, F. Christianos, L. Schäfer, Multi-Agent Reinforcement Learning: Foundations and Modern Approaches, MIT Press, 2024. URL: https://www.marl-book.com.

[7] R. Valk, Object Petri nets: Using the nets-within-nets paradigm, in: J. Desel, W. Reisig, G. Rozenberg (Eds.), Advanced Course on Petri Nets 2003, volume 3098 of *Lecture Notes in Computer Science*, Springer-Verlag, 2003, pp. 819–848.

[8] K. Hoffmann, H. Ehrig, T. Mossakowski, High-level nets with nets and rules as tokens, in: Application and Theory of Petri Nets and Other Models of Concurrency, volume 3536 of *Lecture Notes in Computer Science*, Springer-Verlag, 2005, pp. 268 – 288.

[9] M. Köhler-Bußmeier, A survey on decidability results for elementary object systems, Fundamenta Informaticae 130 (2014) 99–123. doi:10.5555/2608435.2608440.

[10] J. Padberg, L. Kahloul, Overview of reconfigurable Petri nets, in: R. Heckel, G. Taentzer (Eds.), Graph Transformation, Specifications, and Nets, volume 10800, Springer-Verlag, 2018, pp. 201–222.

[11] R. Valk, Self-modifying nets, a natural extension of Petri nets., in: Ausiello, G., Böhm, C. (Eds.), Automata, Languages and Programming, volume 62 of *Lecture Notes in Computer Science*, Springer-Verlag, 1978, pp. 464–476.

[12] M. Köhler-Bußmeier, L. Capra, Analysing probabilistic Hornets, in: E. Amparore, L. Mikulski (Eds.), Application and Theory of Petri Nets and Concurrency (PETRI NETS 2025), volume 15714, 2025, pp. 287–309. doi:10.1007/978-3-031-94634-9_14.

[13] W. M. P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, K. Wolf, Multiparty Contracts: Agreeing and Implementing Interorganizational Processes, The Computer Journal 53 (2010) 90–106.

[14] S. J. J. Leemans, F. M. Maggi, M. Montali, Reasoning on labelled Petri nets and their dynamics in a stochastic setting, in: C. Di Ciccio, R. Dijkman, A. del Río Ortega, S. Rinderle-Ma (Eds.), Business

Process Management, Springer International Publishing, Cham, 2022, pp. 324–342. doi:0.1007/978-3-031-16103-2_22.

[15] M. Köhler-Bußmeier, H. Rölke, Petri-nets@run.time: Handling uncertainty during run-time adaptation using digital twins, in: Petri Nets and Software Engineering 2023, PNSE'23, volume 3430, CEUR, 2023, pp. 34–52. URL: http://ceur-ws.org/Vol-3430/.

[16] M. Köhler-Bußmeier, L. Capra, Modelling and simulation of adaptive multi-agent systems with stochastic nets-within-nets, in: Proceedings of the 16th International Joint Conference on Computational Intelligence, Scitepress, 2024, pp. 313–320. doi:10.5220/0013019700003837.

[17] M. Köhler-Bußmeier, L. Capra, Simulating self-modifying multi-agent systems with probabilistic nets-within-nets, in: K. Arai (Ed.), Intelligent Systems and Applications, Proceedings of the 2025 Intelligent Systems Conference (IntelliSys), Springer Nature Switzerland, Cham, 2026, pp. 147–165.

[18] M. Köhler-Bußmeier, H. Rölke, Petri-nets@run.time: Using Petri nets as run-time models within self-adaptation loops, Transactions on Petri Nets and Other Models of Concurrency XIX (2026). (to appear).

[19] M. Köhler-Bußmeier, J. Sudeikat, Studying the micro-macro-dynamics in MAPE-like adaption processes, in: Intelligent Distributed Computing XVI. IDC 2023, volume 1138 of *Studies in Computational Intelligence*, Springer-Verlag, 2024. doi:10.1007/978-3-031-60023-4_24.

[20] S. Christensen, Y. Hirshfeld, F. Moller, Bisimulation equivalence is decidable for basic parallel processes, in: E. Best (Ed.), CONCUR'93, volume 715 of *Lecture Notes in Computer Science*, Springer, 1993, pp. 143–157.

[21] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st ed., John Wiley & Sons, Inc., USA, 1994.

[22] B. H. C. e. a. Cheng, Software engineering for self-adaptive systems: A research roadmap, in: B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee (Eds.), Software Engineering for Self-Adaptive Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 1–26.

[23] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, C. Becker, A survey on engineering approaches for self-adaptive systems, Pervasive Mob. Comput. 17 (2015) 184–206. doi:10.1016/j.pmcj.2014.09.009.

[24] O. Gheibi, D. Weyns, F. Quin, Applying machine learning in self-adaptive systems: A systematic literature review, ACM Trans. Auton. Adapt. Syst. 15 (2021) 1–37.

[25] J. Wellershaus, M. Köhler-Bußmeier, J. Sudeikat, On switching organisational modes in industrial agent systems: Designing self-organised adjustments of localized autonomy, in: Workshop on Petri Nets for Adaptive Systemes, PNAS'26, 2026.

[26] O. Kummer, F. Wienberg, M. Duvigneau, J. Schumacher, M. Köhler, D. Moldt, H. Rölke, R. Valk, An extensible editor and simulation engine for Petri nets: Renew, in: J. Cortadella, W. Reisig (Eds.), International Conference on Application and Theory of Petri Nets 2004, volume 3099 of *Lecture Notes in Computer Science*, Springer-Verlag, 2004, pp. 484 – 493.

[27] L. Capra, M. Gribaudo, M. Iacono, M. Köhler-Bußmeier, Using rewriting systems for performance analysis, in: C. Colarusso, etãl. (Eds.), QualITA 2025: The Fourth Conference on System and Service Quality, volume 4080, 2025. URL: https://ceur-ws.org/Vol-4080/.

[28] L. Capra, M. Köhler-Bußmeier, Integral implementation of higher-order algebraic Petri nets in maude, in: B. Chatterjee, K. Kothapalli, N. Mittal, A. M. Natarajan, D. Singh (Eds.), Distributed Computing and Intelligent Technology - 22nd International Conference, ICDCIT26, volume 16420, Springer Nature Switzerland, Cham, 2026, pp. 137–153. doi:https://doi.org/10.1007/978-3-032-16632-6.

[29] H. Rölke, Towards mining nets-within-nets, in: Workshop Proceedings of PNSE, ATAED, and PeNGE at PETRI NETS 2025, volume 3998, CEUR, 2025, pp. 81–96. URL: https://ceur-ws.org/Vol-3998/.