Towards Process Mining for Nets within Nets

Heiko Rölke[1]

[1]University of Applied Science of the Grisons, Pulvermühlestrasse 57, CH-7000 Chur

Abstract

In this paper we investigate process mining for nets-within-nets. Nets-within-nets, also known as nets-as-tokens, describe formalisms, where the tokens of a Petri net are nets again. Therefore, a net is located as a token inside a place and this place acts a local context for this net-token. Regarding process mining this leads to scenarios where we like to mine nested structures. Nested structures arises naturally in many applications, i.e., robots are acting within the context a factory or workflows are processed within an organisation.

Unfortunately, existing process mining algorithms generate flat models, which do not express nesting or execution context. Since no mining algorithm especially dedicated to nets-within-nets exists, we try to adapt existing technology as far as possible, as a first step.

In this contribution we compare two approaches to design a process mining techniques for nets-within-nets built upon existing algorithms: The first, so-called **compositional approach**, splits the log file into parts and mines Petri nets independently for each part. The nets-within-nets models is then composed of these separate Petri nets. The second, **decompositional approach**, is just the other way around. It mines one single flat Petri net model from the whole log and then decomposes the net into sub-nets that constitute the parts of the desired nets-within-nets model.

Keywords

Nets-within-Nets, Process Mining, Self-Adaptive Systems, Mobile Agents

1. Mining Nets-within-Nets

Process mining [1] is – beyond any doubt – a success story, especially in the combination with Petri nets as a target formalism. It demonstrates the necessity for a good theoretical foundation when solving realistic problems.

In our research on adaptive systems we have experienced the fact that the concept of a *context* is extremely useful. However, context is a concept that is very hard to capture with flat net models, like p/t nets. In process algebra this gave rise to mobility calculi like the Ambient Calculus [2] or the π-calculus [3]. For process mining the target model is usually not process algebra, but a Petri net model. Therefore, we like to step from flat models towards nested model for process mining.

In this paper, we try to extend process mining onto nets-within-nets [4], i.e. Petri nets where the tokens of are nets again. This concept directly expresses the notion of context as the firing of a net-token's transition happens within a place of the surrounding system net. For example, the system might describe some physical entity, like a factory with robots, and the net-tokens workpieces being process within this factory. Moving net-tokens around expresses mobility of workpieces in a very natural way [5]. Another example is an adaptive system executing some process plan and adaptation is allowed to modify the process plan at run-time, e.g. by following the well-known MAPE-K pattern [6] (monitor-analyse-plan-execute with knowledge). For this scenario the system net describes the MAPE-K loop and the process plan is a net-token being "moved" within the loop (cf. our model in [7]).

Our setting is closely connected to the recent approaches of *object-centric process mining* [8] with the specialty that the in our setting we consider *active*(!) objects, i.e. objects with its own process thread, like assumed e.g. for actor programming [9].

To reach our aim, we try to re-use existing process mining technology [1] to mine nets-within-nets. In our setting we assume that our event log is generated by a very simple nets-within-nets formalism,

PNSE'25, International Workshop on Petri Nets and Software Engineering, 2025
heiko.roelke@fhgr.ch (H. Rölke)
https://www.fhgr.ch/personen/person/roelke/ (H. Rölke)
0000-0002-9141-0886 (H. Rölke)

i.e., by an *elementary object net system* (Eos) [10]. An Eos consists of a system net (which is a p/t net) that contains object nets as tokens (which are called net-tokens). The transitions of these two levels are related by synchronisation.

A central question that arises is how the mining process reflects the compositional nesting structure of nets-within-nets. Here, we study two obvious candidates, which we call **compositional** and **decompositional**. For the compositional approach we filter the log file L into several fragments L_i belonging to the system's parts (like factory and robots), apply processing mining to each L_i independently, and finally *compose* the mined nets into an Eos OS. For the decompositonal approach we start with the complete log L and apply mining techniques to obtain a Petri net N. In the basic case the net N generated is a p/t net. We already know from our work on Eos in [10] that – under certain conditions – the behaviour of an Eos OS can be approximated by a p/t net $\textsc{Rn}(OS)$, called the *reference net*. The major task is then to *decompose* the reference net into its constituting parts, i.e., the system net and the object nets.

At this very early step of our investigation we will investigate whether one of the two approaches seems to be more promising than the other.

The paper has the following structure: Section 2 discusses other approaches to re-construct hierarchical processes. After that, we start with the foundations of our work. Section 3 introduces the formalism of Elementary Object Net Systems (Eos). Section 4 explains in more detail the two mining approaches we identified here: the compositional and the decompositional method, which are based on existing algorithms. We evaluate our approach in Section 5, where we study some examples. The work ends with a conclusion.

2. Related Work

Mining hierarchical and nested process models has attracted increasing interest in the process mining community, as many real-world processes exhibit inherent hierarchical structure. Several approaches have been proposed to extend flat process discovery techniques to support hierarchy.

Xixi Lu et al. introduced Hierarchical Process Trees and techniques to discover them from event logs [11]. Their approach recursively detects nested behavior and constructs a tree-based process model with hierarchical operators. Similarly, Sander Leemans et al. [12] introduced multi-level logs and generalised hierarchical models to discover simpler models from existing logs. Other work has addressed modular and multi-level process discovery. Begicheva et al. [13] present an algorithm for discovering hierarchical process models represented as two-level workflow nets. The algorithm is based on predefined event clustering.

In the domain of Petri nets, several techniques exist for process discovery, but most generate flat models. While hierarchical Petri nets and especially object nets provide natural means to represent nesting, discovery techniques for these extended net types are still in their infancy. Our work builds upon the developments mentioned above by addressing the challenge of discovering nets-within-nets, a more expressive hierarchical formalism where nets act as tokens in other nets.

Compared to existing hierarchical discovery approaches, our contribution specifically targets mining execution context and nested token dynamics — as required by nets-within-nets — rather than purely structural hierarchy.

3. Elementary Object Net Systems (Eos)

In this Section we recall the definition of Eos as presented in the survey [10]. Let $MS(D)$ denote the set of multisets over a set D. A *p/t net* is a tuple $N = (P, T, \mathbf{pre}, \mathbf{post})$, such that P is a set of places, T is a set of transitions, with $P \cap T = \emptyset$, and $\mathbf{pre}, \mathbf{post} : T \to MS(P)$ are the pre- and post-condition functions. An elementary object system (Eos) is composed of a system net, which is a p/t net $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post})$ and a set of object nets $\mathcal{N} = \{N_1, \ldots, N_n\}$, which are p/t nets given as $N = (P_N, T_N, \mathbf{pre}_N, \mathbf{post}_N)$, where $N \in \mathcal{N}$. In extension we assume that all sets of nodes (places

and transitions) are pairwise disjoint. Moreover we assume $\widehat{N} \notin \mathcal{N}$ and the existence of the object net $\bullet \in \mathcal{N}$, which has no places and no transitions and is used to model anonymous, so called black tokens.

The system net places are typed by $d : \widehat{P} \to \mathcal{N}$ with the meaning, that a place $\widehat{p} \in \widehat{P}$ of the system net with $d(\widehat{p}) = N$ may contain only net-tokens of the object net type N.[1] No place of the system net is mapped to the system net itself since $\widehat{N} \notin \mathcal{N}$.

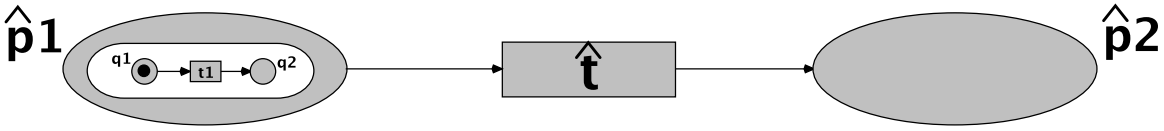

Figure 1: An Elementary Object Net System

Nested Markings Since the tokens of an Eos are instances of object nets, a *marking* of an Eos is a *nested* multiset. A marking of an Eos OS is denoted $\mu = \sum_{k=1}^{|\mu|}(\widehat{p}_k, M_k)$, where $\widehat{p}_k$ is a place of the system net and M_k is the marking of a net-token with type $d(\widehat{p}_k)$. To emphasise the nesting, markings are also denoted as $\mu = \sum_{k=1}^{|\mu|} \widehat{p}_k[M_k]$. For example, the marking of Fig. 1 is $\mu = \widehat{p}_1[q_1]$. The set of all markings which are syntactically consistent with the typing d is denoted $\mathcal{M}$, where $d^{-1}(N) \subseteq \widehat{P}$ is the set of system net places of the type N:

$$\mathcal{M} := MS\left(\bigcup_{N\in\mathcal{N}} \left(d^{-1}(N) \times MS(P_N)\right)\right) \tag{1}$$

We define the partial order $\sqsubseteq$ on nested multisets by setting $\mu_1 \sqsubseteq \mu_2$ iff $\exists \mu : \mu_2 = \mu_1 + \mu$.

Events Analogously to markings, which are nested multisets μ, the events of an Eos are also nested. An Eos allows three different kinds of events – as illustrated by the Eos in Fig. 1.

1. System-autonomous: The system net transition t fires autonomously which moves the net-token from p_1 to p_2 without changing its marking.
2. Object-autonomous: The object net fires transition t_1, which "moves" the black token from q_1 to q_2. The object net itself remains at its location p_1.
3. Synchronisation: The system net transition t fires synchronously with t_1 in the object net. Whenever synchronisation is necessary, autonomous actions are forbidden.

The set of events is denoted Θ. Events are formalised as a pair $\widehat{\tau}[\vartheta]$, where $\widehat{\tau}$ is either the transition that fires in the system net or a special "idle" transition $id_{\widehat{p}}$ (cf. below); and ϑ is a function such that $\vartheta(N)$ is the multiset of transitions, which have to fire synchronously with $\widehat{\tau}$, (i.e. for each object net $N \in \mathcal{N}$ we have $\vartheta(N) \in MS(T_N)$).[2]

In general $\widehat{\tau}[\vartheta]$ describes a synchronisation, but autonomous events are special subcases: Obviously, a system-autonomous events is the special case, where $\vartheta = \mathbf{0}$ with $\mathbf{0}(N) = \mathbf{0}$ for all object nets N. To describe an object-autonomous event we assume the set of *idle transitions* $\{id_{\widehat{p}} \mid \widehat{p} \in \widehat{P}\}$, where $id_{\widehat{p}}$ formalises object-autonomous firing on the place $\widehat{p}$:

[1] In some sense, net-tokens are object nets with its own marking. However, net-tokens should not be considered as *instances* of an object net (as in object-oriented programming), since net-tokens do not have an identity. This is reflected by the fact that the firing rule joins and distributes the net-tokens' markings.

Instead, all net-tokes of an object net act as a *collective* entity, like a group. This group can be considered as an object with identy – an object with its state distributed over the net-tokens. For in in-depth discussion of this semantics cf. [4].

[2] In the graphical representation the events are generated by transition inscriptions. For each object net $N \in \mathcal{N}$ a system net transition $\widehat{t}$ is labelled with a multiset of channels $\widehat{l}(\widehat{t})(N) = ch_1 + \cdots + ch_n$, depicted as $\langle N{:}ch_1, N{:}ch_2, \ldots\rangle$. Similarily, an object net transition t may be labelled with a channel $l_N(t) = ch$ – depicted as $\langle{:}ch\rangle$ whenver there is such a label. We obtain an event $\widehat{t}[\vartheta]$ by setting $\vartheta(N) := t_1 + \cdots + t_n$ to be any transition multiset such that the labels match: $l_N(t_1) + \cdots + l_N(t_n) = \widehat{l}(\widehat{t})(N)$.

1. Each idle transition $id_{\widehat{p}}$ has $\widehat{p}$ as a side condition: $\mathbf{pre}(id_{\widehat{p}}) = \mathbf{post}(id_{\widehat{p}}) := \widehat{p}$.
2. Each idle transition $id_{\widehat{p}}$ synchronises only with transitions from $N = d(\widehat{p})$:

$$\forall \widehat{\tau}[\vartheta] \in \Theta : \ \widehat{\tau} = id_{\widehat{p}} \quad \Longrightarrow \quad \forall N \in \mathcal{N} : \ \ (\vartheta(N) \neq \mathbf{0} \iff N = d(\widehat{p}))$$

Definition 1. *An elementary object system (Eos) is a tuple* $OS = (\widehat{N}, \mathcal{N}, d, \Theta, \mu_0)$, *where:*

1. $\widehat{N}$ *is a p/t net, called the* system net.
2. $\mathcal{N}$ *is a finite set of disjoint p/t nets, called* object nets.
3. $d : \widehat{P} \to \mathcal{N}$ *is the* typing *of the system net places.*
4. Θ *is the set of* events.
5. $\mu_0 \in \mathcal{M}$ *is the initial marking.*

Example Figure 2 shows an Eos with the system net $\widehat{N}$ and the object nets $\mathcal{N} = \{N_1, N_2\}$. The system has four net-tokens: two on place $\widehat{p}_1$ and one on $\widehat{p}_2$ and $\widehat{p}_3$ each. The net-tokens on $\widehat{p}_1$ and $\widehat{p}_2$ share the same net structure, but have independent markings. (Ignore the net-tokens above $\widehat{t}$ and on the places $\widehat{p}_4$, $\widehat{p}_5$ and $\widehat{p}_5$ at this moment, as they illustrate the firing of $\widehat{t}$; they are not part of the marking.)

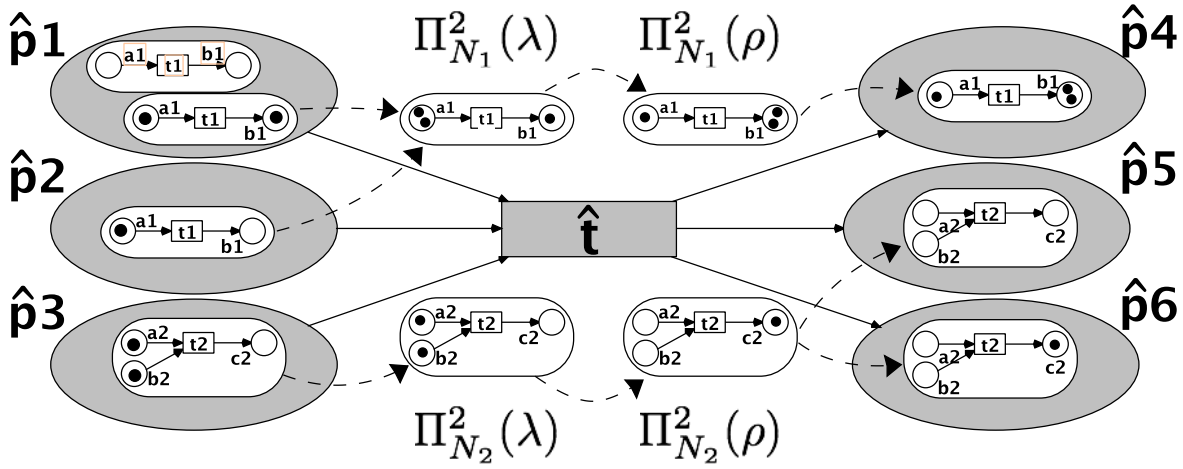

Figure 2: An Elementary Object Net System – illustrating the Firing of $\widehat{t}[N_1 \mapsto t_1, N_2 \mapsto t_2]$

The system net is $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post})$, where $\widehat{P} = \{\widehat{p}_1, \ldots, \widehat{p}_6\}$ and $\widehat{T} = \{\widehat{t}\}$.
We have two object nets $N_i = (P_i, T_i, \mathbf{pre}_i, \mathbf{post}_i), i = 1, 2$ with $P_1 = \{a_1, b_1\}$, $T_1 = \{t_1\}$, $P_2 = \{a_2, b_2, c_2\}$, and $T_2 = \{t_2\}$.
The typing is $d(\widehat{p}_1) = d(\widehat{p}_2) = d(\widehat{p}_4) = N_1$ and $d(\widehat{p}_3) = d(\widehat{p}_5) = d(\widehat{p}_6) = N_2$.
The labelling (not shown in the Figure) generates one event such that $\widehat{t}$ fires synchronously with t_1 and t_2, i.e., we have $\Theta = \{\widehat{t}[N_1 \mapsto t_1, N_2 \mapsto t_2]\}$.
The initial marking has two net-tokens on $\widehat{p}_1$, one on $\widehat{p}_2$, and one on $\widehat{p}_3$:

$$\mu = \widehat{p}_1[a_1 + b_1] + \widehat{p}_1[\mathbf{0}] + \widehat{p}_2[a_1] + \widehat{p}_3[a_2 + b_2]$$

Note that the structure is the same for the three net-tokens on $\widehat{p}_1$ and $\widehat{p}_2$ but the net-tokens' markings are different.

Firing Rule The projection Π^1 on the first component abstracts from the substructure of all net-tokens for a marking of an Eos:

$$\Pi^1 \left(\sum\nolimits_{k=1}^{n} \widehat{p}_k[M_k] \right) := \sum\nolimits_{k=1}^{n} \widehat{p}_k \tag{2}$$

The projection Π^2_N on the second component is the sum of all net-token markings M_k of the type $N \in \mathcal{N}$, ignoring their local distribution within the system net:

$$\Pi^2_N \left(\sum_{k=1}^{n} \widehat{p}_k[M_k] \right) := \sum_{k=1}^{n} \mathbf{1}_N(\widehat{p}_k) \cdot M_k \tag{3}$$

where the indicator function $\mathbf{1}_N : \widehat{P} \to \{0, 1\}$ is $\mathbf{1}_N(\widehat{p}) = 1$ iff $d(\widehat{p}) = N$. Note that $\Pi^2_N(\mu)$ results in a marking of the object net N.

A system event $\widehat{\tau}[\vartheta]$ removes net-tokens together with their individual internal markings. Firing the event replaces a nested multiset $\lambda \in \mathcal{M}$ that is part of the current marking μ, i.e. $\lambda \sqsubseteq \mu$, by the nested multiset ρ. Therefore the successor marking is $\mu' := (\mu - \lambda) + \rho$. The enabling condition is expressed by the *enabling predicate* ϕ_{OS} (or just ϕ whenever OS is clear from the context):

$$\begin{aligned} \phi(\widehat{\tau}[\vartheta], \lambda, \rho) \quad \iff & \Pi^1(\lambda) = \mathbf{pre}(\widehat{\tau}) \wedge \Pi^1(\rho) = \mathbf{post}(\widehat{\tau}) \wedge \\ & \forall N \in \mathcal{N} : \Pi^2_N(\lambda) \geq \mathbf{pre}_N(\vartheta(N)) \wedge \\ & \forall N \in \mathcal{N} : \Pi^2_N(\rho) = \Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N)) \end{aligned} \tag{4}$$

With $\widehat{M} = \Pi^1(\lambda)$ and $\widehat{M'} = \Pi^1(\rho)$ as well as $M_N = \Pi^2_N(\lambda)$ and $M'_N = \Pi^2_N(\rho)$ for all $N \in \mathcal{N}$ the predicate ϕ has the following meaning:

1. The first conjunct expresses that the system net multiset $\widehat{M}$ corresponds to the pre-condition of the system net transition $\widehat{\tau}$, i.e. $\widehat{M} = \mathbf{pre}(\widehat{\tau})$.
2. In turn, a multiset $\widehat{M'}$ is produced, that corresponds to the post-set of $\widehat{\tau}$.
3. A multi-set $\vartheta(N)$ of object net transitions is enabled if the sum M_N of the net-token markings (of type N) enable it, i.e. $M_N \geq \mathbf{pre}_N(\vartheta(N))$.
4. The firing of $\widehat{\tau}[\vartheta]$ must also obey the *object marking distribution condition*: $M'_N = M_N - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N))$, where $\mathbf{post}_N(\vartheta(N)) - \mathbf{pre}_N(\vartheta(N))$ is the effect of the object net's transitions on the net-tokens.

Note that conditions 1. and 2. assure that only net-tokens relevant for the firing are included in λ and ρ. Conditions 3. and 4. allow for additional tokens in the net-tokens.

For system-autonomous events $\widehat{t}[\mathbf{0}]$ the enabling predicate ϕ can be simplified further. We have $\mathbf{pre}_N(\mathbf{0}(N)) = \mathbf{post}_N(\mathbf{0}(N)) = \mathbf{0}$. This ensures $\Pi^2_N(\lambda) = \Pi^2_N(\rho)$, i.e. the sum of markings in the copies of a net-token is preserved w.r.t. each type N. This condition ensures the existence of linear invariance properties

Analogously, for an object-autonomous event we have an idle-transition $\widehat{\tau} = id_{\widehat{p}}$ for the system net and the first and the second conjunct is: $\Pi^1(\lambda) = \mathbf{pre}(id_{\widehat{p}}) = \widehat{p} = \mathbf{post}(id_{\widehat{p}}) = \Pi^1(\rho)$. So, there is an addend $\lambda = \widehat{p}[M]$ in μ with $d(\widehat{p}) = N$ and M enables $\vartheta(N)$.

Definition 2 (Firing Rule). *Let OS be an Eos and $\mu, \mu' \in \mathcal{M}$ markings. The event $\widehat{\tau}[\vartheta]$ is enabled in μ for the mode $(\lambda, \rho) \in \mathcal{M}^2$ iff $\lambda \sqsubseteq \mu \wedge \phi(\widehat{\tau}[\vartheta], \lambda, \rho)$ holds.*

An event $\widehat{\tau}[\vartheta]$ that is enabled in μ for the mode (λ, ρ) can fire: $\mu \xrightarrow[OS]{\widehat{\tau}[\vartheta](\lambda,\rho)} \mu'$. The resulting successor marking is defined as $\mu' = \mu - \lambda + \rho$.

Note that the firing rule makes no a-priori assumptions about how to distribute the object net markings onto the generated net-tokens. Therefore we need the mode (λ, ρ) to formulate the firing of $\widehat{\tau}[\vartheta]$ in a functional way.

As for p/t nets the firing rule has several nice properties [10]: Firing enjoys monotonicity, it is symmetric when inverting arc directions, and when projecting a firing sequence of an Eos onto the system net we obtain a valid firing sequence of the system net considered as a p/t net.

Example Consider the Eos of Figure 2 again. The current marking μ of the Eos enables $\widehat{t}[N_1 \mapsto t_1, N_2 \mapsto t_2]$ in the mode (λ, ρ), where

$$\begin{aligned} \mu &= \widehat{p}_1[\mathbf{0}] + \widehat{p}_1[a_1 + b_1] + \widehat{p}_2[a_1] + \widehat{p}_3[a_2 + b_2] = \widehat{p}_1[\mathbf{0}] + \lambda \\ \lambda &= \widehat{p}_1[a_1 + b_1] + \widehat{p}_2[a_1] + \widehat{p}_3[a_2 + b_2] \\ \rho &= \widehat{p}_4[a_1 + b_1 + b_1] + \widehat{p}_5[\mathbf{0}] + \widehat{p}_6[c_2] \end{aligned}$$

The net-token markings are added by the projections Π^2_N resulting in the markings $\Pi^2_N(\lambda)$. The sub-synchronisation generates $\Pi^2_N(\rho)$. (The results are shown above and below the transition $\widehat{t}$.) After the synchronisation we obtain the successor marking μ' with net-tokens on $\widehat{p}_4$, $\widehat{p}_5$, and $\widehat{p}_6$ as shown in Figure 2:

$$\begin{aligned}\mu' &= (\mu - \lambda) + \rho = \widehat{p}_1[\mathbf{0}] + \rho \\ &= \widehat{p}_1[\mathbf{0}] + \widehat{p}_4[a_1 + b_1 + b_1] + \widehat{p}_5[\mathbf{0}] + \widehat{p}_6[c_2]\end{aligned}$$

Note, that we have only presented one mode (λ, ρ) of the event and that other modes are possible, too.

4. Mining Approaches for Eos

The behaviour of an Eos differs from the system-net (and from the object nets) when considered in isolation due to the synchronisation; the effect is very similar to modularised p/t nets [14]. To express this relation of surrounding system net and net-tokens we have defined in [10] the so-called *reference net*, which is the p/t net obtained by putting system net and the object nets aside (by a set theoretical union of places) and use all synchronisations as transistions.

Since the places of all nets in $\mathcal{N}$ are disjoint by definition, the projections $(\Pi^1(\mu), (\Pi^2_N(\mu))_{N\in\mathcal{N}})$ can be identified with the multiset:

$$\textsc{Rn}(\mu) := \Pi^1(\mu) + \sum_{N\in\mathcal{N}} \Pi^2_N(\mu)$$

The Reference Net For each Eos there is an obvious construction of a p/t net, called the *reference net*, which is constructed by taking as the set of places the disjoint union of all places and the set of events as transitions.

Definition 3. *Let $OS = (\widehat{N}, \mathcal{N}, d, \Theta, \mu_0)$ be an Eos. The* reference net *$\textsc{Rn}(OS)$ is defined as the p/t net:*

$$\textsc{Rn}(OS) = \left(\left(\widehat{P} \cup \bigcup\nolimits_{N\in\mathcal{N}} P_N\right), \Theta, \mathbf{pre}^{\textsc{Rn}}, \mathbf{post}^{\textsc{Rn}}, \textsc{Rn}(\mu_0)\right)$$

where $\mathbf{pre}^{\textsc{Rn}}$ (and analogously: $\mathbf{post}^{\textsc{Rn}}$) is defined by:

$$\mathbf{pre}^{\textsc{Rn}}(\widehat{\tau}[\vartheta]) = \mathbf{pre}(\widehat{\tau}) + \sum\nolimits_{N\in\mathcal{N}} \mathbf{pre}_N(\vartheta(N))$$

The net is called *reference net* because it behaves as if each object net would have been accessed via pointers and not like a value: A black token on a system net place $\widehat{p}$ is interpreted as a pointer to the object $N = d(\widehat{p})$, where each object net has exactly one instance but several pointers referring to it.

Theorem 1 ([10], Thm. 6.1). *Every event $\widehat{\tau}[\vartheta]$ that is activated in OS for (λ, ρ) is so in $\textsc{Rn}(OS)$:*

$$\mu \xrightarrow[OS]{\widehat{\tau}[\vartheta](\lambda,\rho)} \mu' \quad\Longrightarrow\quad \textsc{Rn}(\mu) \xrightarrow[\textsc{Rn}(OS)]{\widehat{\tau}[\vartheta]} \textsc{Rn}(\mu')$$

Please note that while the reference net is sufficient to recover the structure it is not behaviorally equivalent to the original Eos as every enabled firing sequence of the Eosis enabled in the reference net, too, but not vice versa – the α-centauri example serves as a counter example for the latter [10]. The main reason is that the system net places of an Eos act as a *context* for the object-nets. So, the reference net is a kind of behavioral over-approximation of an Eos and among the set of all p/t nets it is the best approximation. However, in the mining context this theoretical significant difference is of less importance, since we only consider event logs and in logs we can only observe possible Eos sequences which are also present in the over-approximation. Therefore since we already know that the sequence has been possible in the Eos it is sufficient to mine this reference net. A major plus point is that we can re-use existing process mining algorithms. To conclude: The reference net is sufficient to reconstruct the Eos as long as we have the additional information, which places belong to the system net and which to the object nets.

Compositional vs. Decompositional Approaches We like to avoid to develop an Eos-specific process mining algorithm. Instead, we are interested in re-using existing, well-optimised process mining algorithms. However, it is not clear which kind of pre-processing of our logs is needed and which post-processing steps are needed to generate an Eos from the mined nets. We see two alternative approaches to mine Eos, which differ in the order of mining and (de)composition:

1. (Composition) If we identify from the log the parts belonging to specific object nets N_i, we can filter the log for each and generate several object nets; analogously for the system net.

$$\text{Log } L \xrightarrow{\text{filter}} \text{Logs } L_i \xrightarrow{\text{mine}^n} \text{Nets } N_i \xrightarrow{\text{integrate}} \text{Eos } OS$$

   However, it is subtle how this integration (or: synchronisation) is carried out in detail, as we have a many-to-many synchronisation for Eos.
2. (Decomposition) In the second case, we obtain the following tool chain. From the log we mine a p/t net, which is expected to be the reference net of the generating Eos. By using additional information which places belong to the surrounding system and which to the nested net-tokens we decompose the reference net into system- and object-net, i.e., an Eos:

$$\text{Log } L \xrightarrow{\text{mine}} \text{Net } N \simeq \text{Reference-Net } \textsc{Rn}(OS) \xrightarrow{\text{decompose}} \text{Eos } OS$$

   On the pro side we have the whole log information during the mining; on the contrary, we have to decompose and it is not clear which places belong to which part.

In the following we evaluate both approaches studying a scenario of a mobile robot.

5. Comparative Evaluation of the two Approaches

To evaluate the two approaches to mining nets-within-nets, we use a simple scenario, introduced in the following section. Afterwards, both approaches are tested and compared.

5.1. Scenario: A Coffee Serving Robot

We study a simplified variant of the mobile household robot [5]. Imagine a typical day of a university professor. When working, the professor can write papers or give lessons. Thanks to modern communication techniques, he or she does not have to leave the comfort of the office for these tasks. Every once a while, the professor needs coffee - other duties certainly have to wait then. To spend leftover funding, the department our example professor works in has purchased a coffee delivering robot system. The professor just has to order an coffee and wait for it. The coffee robot does all the work.

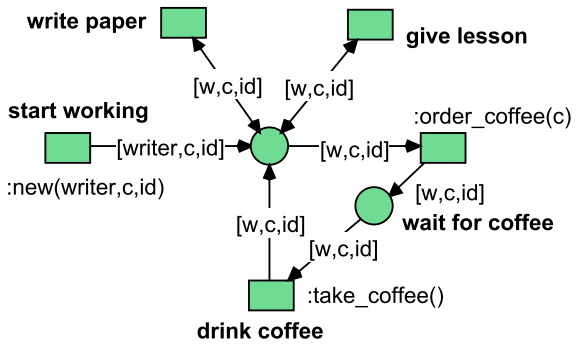

Figure 3: Household System: Object Net Prof (simplified)

Figure 3 shows the object net specifying the professor. Note that this and the other nets shown in this section are in principal executable reference nets in Renew format, but certain details have been left out to avoid visual clutter (e.g., net inscriptions for logging or type definitions).

Some remarks on the Prof net: Inscriptions in **bold font** are names or explanations. Arcs are inscribed with variables packed in tuples - these tuples are the markings in Renew reference nets. Transition inscription starting with a colon, like :new(...) are synchronous channels, linking this net to other nets.

We can easily see that our Prof net is a net-token of another net. This would be the system net, controlling the overall system behaviour. Let's have a look at the system net.

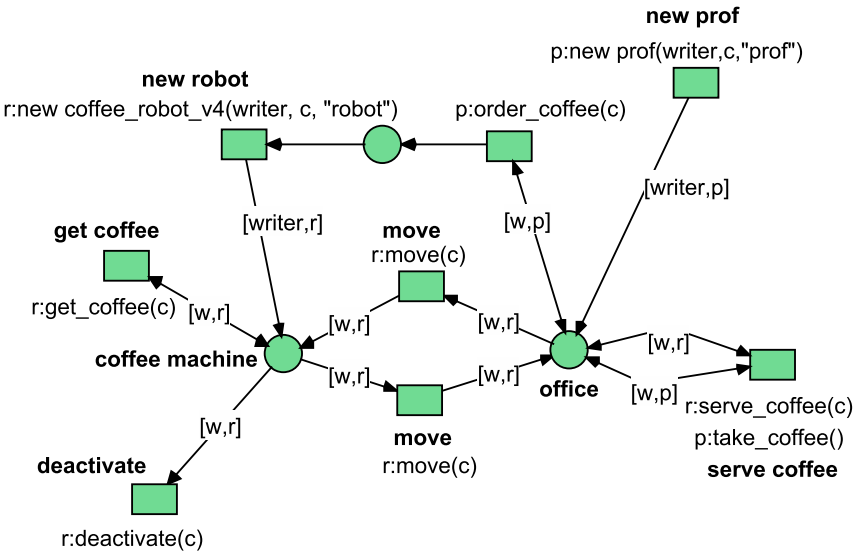

Figure 4: Household System: The System Net $\widehat{N}$ (simplified)

Figure 4 shows the system net $\widehat{N}$. The system net defines the spatial contexts of our simple example, **coffee machine** and **office**. Once a Prof is introduced to the system, she or he cannot leave the office anymore[3]. Actions like researching and teaching are not considered here. The only interactions to the outside world are ordering coffee and taking coffee from the robot.

For the coffee robot, more possible interactions are modeled. A coffee robot is activated (**new robot**) once a professor orders coffee. The coffee robot can get coffee, deactivate itself, move between coffee machine location and offices, and serve coffee. This is where to spatial contexts come into play: coffee can only be fetched at the coffee machine and served at the professor's office. The robot has to move to fulfill its tasks.

Let's have a look at the coffee robot, also implemented in terms of an object net: Figure 5 shows the object net specifying the robot.

Once a robot is activated by someone ordering a coffee, the robot can move around and get coffee. Holding a coffee, the coffee can be served and the robot can be deactivated afterwards. Note that there is no planning or scheduling procedure for the robot – it is possible that the robot moves aimlessly without end and the professor waits for the coffee forever. An important observation is the interplay between spatial context (the system net) and the robot: Certain actions like getting a coffee or serving it are only possible at the right place. The robot has to move between places to fulfill its tasks. A log of the system can be found in Table 1.

It is not important to look at every detail of the log, but an important observation is that certain events lead to two or more log entries, namely synchronized events between system and object nets. Examples are the move events (e.g., move to office and move, in red) or the serving of the coffee in green. The latter spans over all three nets, as can be seen in the last column (net instance). Based on log

[3]In our view, this is a realistic assumption.

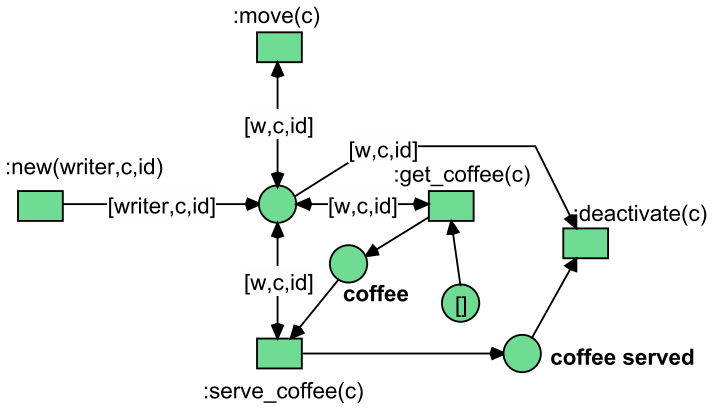

Figure 5: Household System: Object Net Robot (simplified)

case	event	timestamp	net instance
1	new prof	03/24/2025 10:38:06:942	system
1	start working	03/24/2025 10:38:06:943	prof
1	give lesson	03/24/2025 10:38:06:946	prof
1	give lesson	03/24/2025 10:38:06:948	prof
1	give lesson	03/24/2025 10:38:06:949	prof
1	give lesson	03/24/2025 10:38:06:951	prof
1	order coffee	03/24/2025 10:38:06:952	prof
1	new robot	03/24/2025 10:38:06:957	system
1	move to office	03/24/2025 10:38:06:959	system
1	move	03/24/2025 10:38:06:960	robot
1	move	03/24/2025 10:38:06:962	robot
1	move to coffee	03/24/2025 10:38:06:962	system
1	get coffee	03/24/2025 10:38:06:964	system
1	get coffee	03/24/2025 10:38:06:965	robot
1	move to office	03/24/2025 10:38:06:966	system
1	move	03/24/2025 10:38:06:967	robot
1	serve coffee	03/24/2025 10:38:06:982	system
1	drink coffee	03/24/2025 10:38:06:983	prof
1	serve coffee	03/24/2025 10:38:06:983	robot
1	move	03/24/2025 10:38:06:985	robot
1	move to coffee	03/24/2025 10:38:06:985	system
1	write paper	03/24/2025 10:38:06:986	prof
1	remove robot	03/24/2025 10:38:06:988	system
1	deactivate	03/24/2025 10:38:06:988	robot

Table 1
A Log of the Household System

files like the one in the example, we can try to reconstruct the overall net system, as outlined in the previous section.

5.2. Decompositional Approach

An obvious and very simple approach is to dump the complete log file into an existing mining algorithm and see what happens. That is the idea behind the decompositional approach: mine the entire log file and decompose the mined net into the nets that generated the log. Ideally, there should be a loose coupling between these subnets, so that there are identifiable cuts.

A little bit of thinking reveals that this cannot always be the case, as it is easy to construct nets-within-nets with intertwined control flow handed over from net to net. However, let's have a look at

the mining results of our example.

For the mining, we use the PM4Py framework.[4] For importing the log data, the only necessary preprocessing step was to describe the date format, so that it can be properly included to a pandas[5] dataframe. Note that the log we used here was the result of a longer run of the example net system and included several profs and robots acting in parallel. Figure 6 shows the net mined.

We tried several mining algorithms, the inductive miner produced the most meaningful results. However, the mining results are unsatisfactory. The mined net allows processes that are not possible in the original net and vice versa. One example: new prof → start working → deactivate, without a robot being created before. In addition, all elements of the system are intertwined and hard to disentangle. We have no idea how to split the mined flat net into the object nets without heavily relying on domain knowledge. If this is not possible for a rather simple net system of three object nets, we do not have much hope for the general case. Therefore, the decompositional approach cannot be used for our purposes.

5.3. Compositional Approach

The log data in Table 1 contains the net instance name (system, robot, prof) in the last column. It is straightforward to separate the log file to object net logs, consisting only of the events of the respective object net.

Doing so, we can follow the compositional approach and mine the object nets separately. An interesting fact is that for the separate object nets, the alpha plus miner sometimes works better than the inductive miner, while it more or less failed for the mining of the complete log.

Figure 7 shows the system net mined, relying on the alpha plus miner for this task. The system net is quite similar to the net we defined, as can be seen by comparing Figure 7 to Figure 4. An important difference is that the order coffee transition is unconnected in the mined net. The rest is not too bad and more or less resembles the original behavior.

Please note that we did several runs of the robot-prof system leading to log files of various sizes. Mining these logs led to subtle differences in the mined nets. All mined nets showed one or more deviations from the original net system, similar to the problem described above (order coffee). We only show one result here. The shortcomings of this example are similar to those of all the other outcomes.

Let us have a look at the other object nets. Figure 8 shows the object net mined for the professor. The professor described by the mined net seems to be more of a caffeine-addict than our original one. Without coffee, no work (writing papers, giving lessons) is done, making the mined professor maybe even a bit more real. Beside this difference and some not really necessary silent transitions, the professor net can be counted a success. The professor net was mined using the inductive miner, as the alpha plus miner produced a professor that can only drink coffee and is never working.

Figure 9 shows the mined net for the robot, using the inductive miner. On first sight, this net looks even better than professor net, as it is more compact. However, on closer inspection one can see that the mined robot can be deactivated while still carrying a coffee. That would be a pity and can lead to a deadlock in the overall system, as the professor would wait forever for the coffee without getting anything done. Another notable difference is that the robot can serve one coffee several times. It is important to know, that this is not due to shortcomings in the log file.

It is interesting that the alpha plus miner produced better results for the robot, as can be seen in Figure 10: Coffee has to be served before deactivating, and the robot can move at any time. So far, we cannot give a universal rule which mining algorithm produces better results for our purposes.

The results of the experiments with the compositional approach shown above are more promising than those of the decompositional approach. While it is not surprising that we get separate object nets, these nets more or less resemble the object nets designed by us and can serve as a starting point for the composition step. The composition itself is straightforward, but we do not have an implemented algorithm for it. The nets produced by the miner (shown above) have to be inscribed with synchronous

[4] see https://pypi.org/project/pm4py/
[5] see https://pandas.pydata.org/

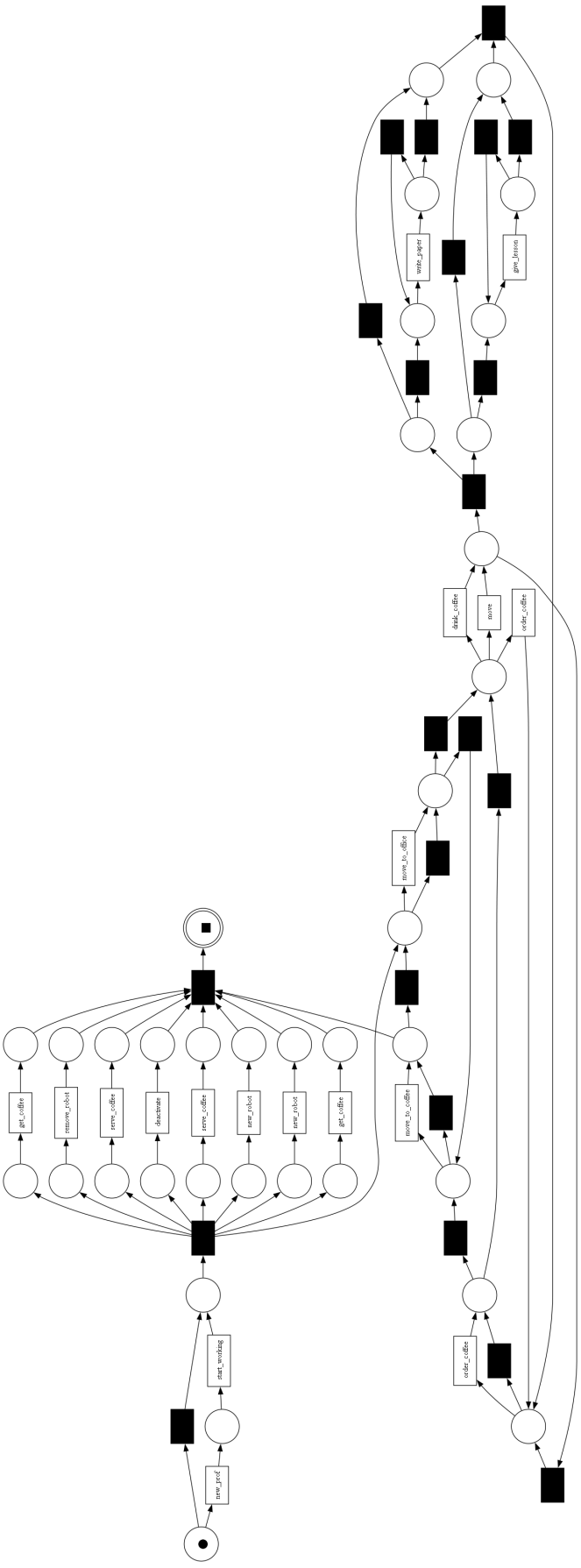

Figure 6: Output of the inductive miner in pm4py for the example log

channels at those transitions that should occur simultaneously (synchronized transitions). The object nets have to be instantiated (special channel new) in the system net and the arcs of the system net have to be inscribed with identifier variables for the object nets. Doing so by hand revealed some minor issues but produced an executable net system that can be simulated using the Renew software.

One minor issue is that the mining algorithms produce a marked starting place for each net, while this is only necessary for the system net and potentially produces a deadlock in the object nets. Another issue we discovered was due to some sloppiness in our logging: we sometimes used the names of the transitions as log entries instead of the channel inscriptions - if available. This has to be stratified when automating our approach in the future.

Besides these small shortcomings the compositional approach can be counted a success.

5.4. Discussion and Next Steps

In the experiments described above, we tried out both the decompositional approach and the compositional approach. The decompositional approach – using the entire log file of the net system as the source for the mining and decomposing the mined net afterwards – did not produce any meaningful results. The compositional approach, i.e. splitting or filtering the log file according to the separate subnets, mining those logs separately and composing the mined nets afterwards, seems to be a possible approach for going further.

During the experiments we discovered an additional problem, namely the usage of the "right" mining algorithm. We have to carry out additional experiments in this direction. Hopefully, we can find a suitable way or at least some best practice.

An important next step is to automate the parts of our experiment that we did manually, like the net composition. We are optimistic that this can be done, but additional problems might occur when generalizing the findings of the small case study presented here.

Other steps to be done:

- Automating the logging in Renew
- Deciding about the best mining algorithm
- Possibly integrating the mining into the Renew tool for convenience
- Automating the net composition

We will tackle these steps in the near future, as we do not expect bigger problems here. When done, we have achieved a round-trip mining of nets-within-nets. We will base further research on these achievements, as the more general goal is to be able to mine net systems without the exact knowledge about the subnets.

6. Conclusion and Outlook

In this paper we extended Petri net based process mining to nets-within-nets, i.e., nets where the tokens are Petri nets again. This is useful especially for scenarios where events are contextualised, either because they happen within some physical location or within some abstract context like an outer self-adaption loop, like in the MAPE-K pattern.

To capture the structure of nets-within-nets, i.e., the nesting structure of nets and their vertical synchronisation, during the mining process we made experiments with two alternative candidates, which we call the compositional and the decompositional approach. The compositional approach integrates the nets N_i that are mined for the log files L_i filtered w.r.t. different aspects of the system, while the decompositional approach mines the reference net $\textsc{Rn}(OS)$ and tries to decompose it into the Eos OS.

We observed from our experiments that the decompositional approach seems to be less promising than the compositional one. The compositional approach in its present state still needs some additional work, as outlined in the section above.

We plan the following research threads for future research: Firstly, we like to extend our work mine *probabilistic* Nets-within-Nets [15]. This extension seems to be straightforward since the log files contains the event frequencies anyway. Secondly, we plan to investigate the mining of net systems without knowing the exact subnets that produced the log file.

Thirdly, we would like to go beyond Eos. A major extension is to allow algebraic expressions for arcs with the meaning that the topology of net-tokens is modified during firing. This aspect is covered by eHornets [16]. In this case we are faced with a new challenge whenever mining the structure of objects nets: Since firing modifies the net-tokens, the order (or time stamps) of events becomes important, since a dependency present in an early part of the log might be no longer valid in a later part. So we either have to separate the log whenever a system net transition modifies the net-tokens topology or we introduce some 'aging' mechanism into the mining algorithm such that older events are given less confidence than more recent ones.

Acknowledgment The author wishes to thank Michael Köhler-Bussmeier for his help on previous version of this paper, especially in the sections defining the Eos.

Declaration on Generative AI

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

References

[1] W. M. P. van der Aalst, Process Mining: Data Science in Action, 2 ed., Springer, Heidelberg, 2016. doi:10.1007/978-3-662-49851-4.

[2] L. Cardelli, A. D. Gordon, G. Ghelli, Mobility types for mobile ambients, in: Proceedings of the Conference on Automata, Languages, and Programming (ICALP'99), volume 1644 of *Lecture Notes in Computer Science*, Springer-Verlag, 1999, pp. 230–239.

[3] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, parts 1-2, Information and computation 100 (1992) 1–77.

[4] 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.

[5] M. Köhler, D. Moldt, H. Rölke, Modelling mobility and mobile agents using nets within nets, in: W. v. d. Aalst, E. Best (Eds.), International Conference on Application and Theory of Petri Nets 2003, volume 2679 of *Lecture Notes in Computer Science*, Springer-Verlag, 2003, pp. 121–140.

[6] D. Weyns, An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective, John Wiley & Sons Ltd, 2020.

[7] 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.

[8] W. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: P. C. Ölveczky, G. Salaün (Eds.), Software Engineering and Formal Methods (SEFM 2019), volume 11724 of *Lecture Notes in Computer Science*, Springer-Verlag, 2019, pp. 3–25.

[9] G. Agha, I. Mansons, S. Smith, C. Talcott, Towards a theory of actor computation, in: W. R. Cleaveland (Ed.), Third international conference on concurrency theory (CONCUR 92), volume 630 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992, pp. 565–579.

[10] M. Köhler-Bußmeier, A survey on decidability results for elementary object systems, Fundamenta Informaticae 130 (2014) 99–123.

[11] X. Lu, A. Gal, H. A. Reijers, Discovering hierarchical processes using flexible activity trees for event abstraction, in: 2020 2nd International Conference on Process Mining (ICPM), 2020, pp. 145–152. doi:10.1109/ICPM49681.2020.00030.

[12] S. J. Leemans, K. Goel, S. J. van Zelst, Using multi-level information in hierarchical process mining: Balancing behavioural quality and model complexity, in: 2020 2nd International Conference on Process Mining (ICPM), 2020, pp. 137–144. doi:10.1109/ICPM49681.2020.00029.

[13] A. Begicheva, I. Lomazova, R. Nesterov, Discovering hierarchical process models: an approach based on events partitioning, Modeling and Analysis of Information Systems 31 (2024) 294–315. doi:10.18255/1818-1015-2024-3-294-315.

[14] S. Christensen, L. Petrucci, Modular state space analysis of coloured Petri nets, in: Proceeding of the 16th International Conference on Application and Theory of Petri Nets, 1995, pp. 201–217.

[15] M. Köhler-Bußmeier, L. Capra, Analysing probabilistic Hornets, in: E. Amparore, Łukasz Mikulski (Eds.), Proceedings of Application and Theory of Petri Nets and Concurrency 2025, 2025.

[16] M. Köhler-Bußmeier, Restricting Hornets to support adaptive systems, in: W. van der Aalst, E. Best (Eds.), PETRI NETS 2017, Lecture Notes in Computer Science, Springer-Verlag, 2017.

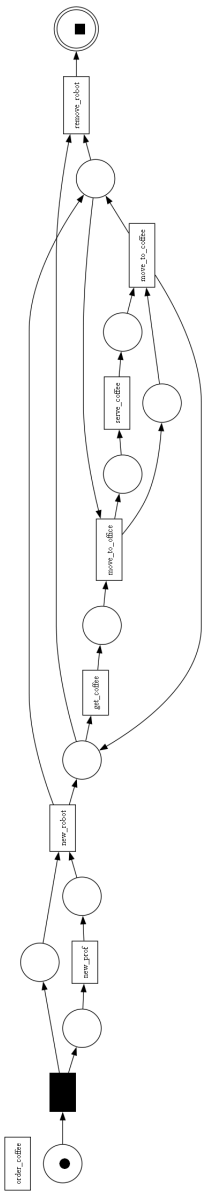

Figure 7: Output System

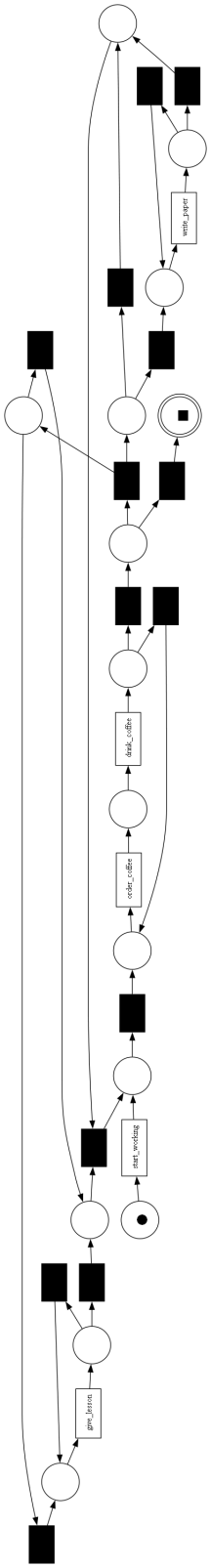

Figure 8: Mined Object Net for Prof

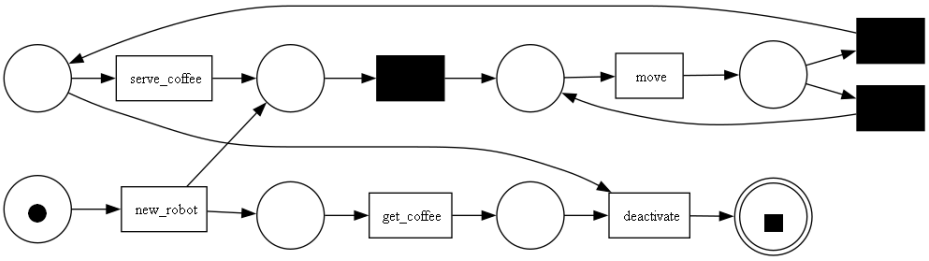

Figure 9: Mined Object Net for Robot

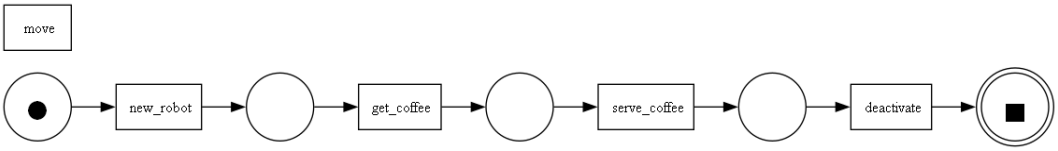

Figure 10: Mined Object Net for Robot using the alpha plus miner