# Compositional Conformance Checking of Nested Petri Nets and Event Logs of Multi-Agent Systems*

Khalil Mecheraoui[1,2], Julio C. Carrasquel[1], and Irina A. Lomazova[1]

[1] National Research University Higher School of Economics,
Myasnitskaya ul. 20, 101000 Moscow, Russia
k_mecheraoui@esi.dz, jcarrasquel@hse.ru, ilomazova@hse.ru
[2] University of Constantine 2 — Abdelhamid Mehri,
Nouvelle ville Ali Mendjeli BP : 67A, 25000 Constantine, Algeria

**Abstract.** This paper presents a compositional conformance checking approach between nested Petri nets and event logs of multi-agent systems. By projecting an event log onto model components, one can perform conformance checking between each projected log and the corresponding component. We formally demonstrate the validity of our approach proving that, to check fitness of a nested Petri net is equivalent to check fitness of each of its components. Leveraging the multi-agent system structure of nested Petri nets, this approach may provide specific conformance diagnostics for each system component as well as to avoid to compute artificial boundaries when decomposing a model for conformance checking.

**Keywords:** Process mining, conformance checking, Petri nets, nested Petri nets, multi-agent systems, fitness.

## 1 Introduction

Lift, thrust, drag, and gravity are the four forces helping an airplane fly. Process mining has similarly four forces to measure its quality namely fitness, generalization, precision, and simplicity [2]. Conformance checking, which is one of the three pillars of process mining, is actually the fitness force [3]. It allows to check how well modeled behavior conforms reality as recorded in an event log. Conformance checking has become relevant in areas such as business alignment [12], auditing [14], and financial software testing [5]. However, current conformance checking approaches fall short when analyzing large event logs of complex multi-agent systems. These systems are characterized by a large number of agents interacting, and exhibiting a high degree of concurrency. In this light, it makes sense to use compositional approaches, where a conformance problem can be decomposed into smaller problems (e.g. [13, 10, 6]). In [13], the author formalizes the so-called *valid decomposition* to decompose conformance problems. This decomposition approach represents no problem from a conformance point of view.

* This work is supported by the Basic Research Program at the National Research University Higher School of Economics.

In [10], the authors proposed to decompose models using the idea of *single-entry* and *single-exit* (SESE). A SESE component is a subnet that has a simple interface w.r.t the rest of the net. Another approach is presented in [6] to compute the overall conformance of a model by merging previously decomposed fragments.

Nevertheless, these approaches use process models whose structure may not provide a clear distinction between system components and their boundaries. This leads these approaches to increase their complexity to compute such boundaries (e.g., where to decompose a model, how many components, etc). Moreover, it may happen that the decomposition is *artificial*, i.e., fragments of a decomposed model do not represent a real division of a system, so diagnostics for each real component may not be provided. In this sense, we propose the use of models of multi-agent systems. In particular, we consider nested Petri nets (NP-nets) [7] — an extension of Petri nets, where tokens can be Petri nets themselves, allowing to model multi-agent systems. NP-nets have been already used in the broader context of process modeling and workflow management [4, 8].

Fig. 1 depicts an example of a NP-net describing an *automated assistant engine* that can serve multiple customers concurrently. A NP-net consists of a *system net*, i.e., modeling the system's environment, and a set of *net tokens*, denoting interacting agents. Each net token has an inner Petri net structure describing agent behavior.

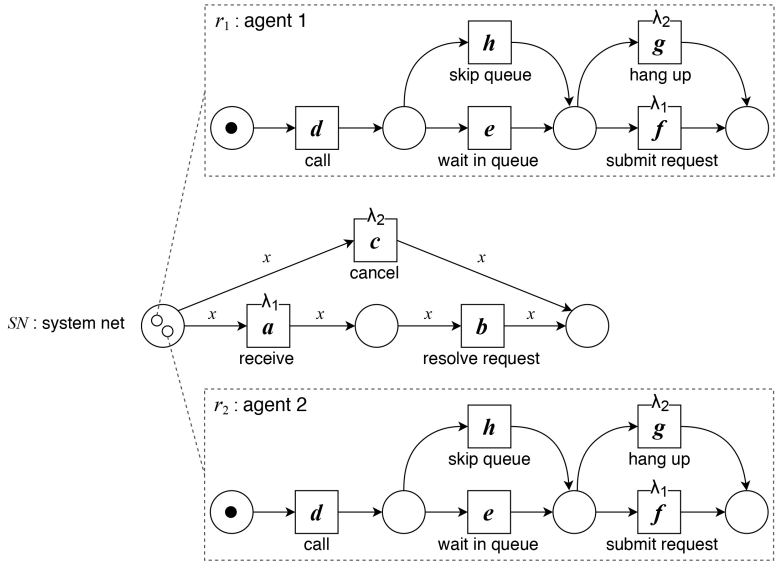


Fig. 1: A nested Petri net where the *system net SN* models an *automated assistant engine*, serving customers concurrently (in this case, *agents* $r_1$ and $r_2$).

In this paper, we present a compositional conformance checking approach between nested Petri nets and event logs of multi-agent systems. Given an event log of a multi-agent system, we decompose it into several *projections* according

to the model components. Then, a conformance checking technique (e.g., replay, alignment) can be performed separately between each projection and the corresponding model component (an agent or the system net). We assume that each agent in the event log corresponds to a net token in the nested Petri net model. For this task, we provide clear definitions regarding a subclass of nested Petri nets and event logs of multi-agent systems. To demonstrate the validity of our approach, we consider the notion of *fitness*. If a model has *perfect* fitness, then all log traces can be replayed on the model from beginning to end. In this work, we map such notion of an event log that *perfectly fits* a model, by defining how an event log of a multi-agent system fits a nested Petri net. Consequently, as an important result of this paper, we state and prove the following theorem: an event log of a multi-agent system *perfectly fits* a nested Petri net if and only if the event log is syntactically correct w.r.t to the nested Petri net and each projection perfectly fits the corresponding model component. This theorem justifies the validity of our compositional approach.

The remainder of this paper is structured as follows. In section 2, we describe nested Petri nets. In section 3, we define the structure for event logs of multi-agent systems. In section 4, we present the compositional conformance checking approach of nested Petri nets and event logs of multi-agent systems, including the aforementioned theorem and its proof. Finally, section 5 presents some conclusions and future work.

## 2 Nested Petri Nets

$\mathbb{N}$ denotes the set of natural numbers (including zero). Let $S$ be a set. The set of all subsets of $S$ is called a *power set*, denoted as $\mathcal{P}(S)$, e.g., the power set of $S = \{a, b\}$ is $\mathcal{P}(S) = \{\{a, b\}, \{a\}, \{b\}, \emptyset\}$. A *multiset* over $S$ is a mapping $m : S \to \mathbb{N}$. In other words, a multiset is a collection of elements, each of them with certain multiplicity, e.g., $\{b, a, b\}$, $\{a, a, b\}$, and $\emptyset$ are multisets over $S$. For compactness, we write $\{a^3, b^2\}$ for $\{a, a, a, b, b\}$. $\mathbb{P}_m(S)$ denotes the set of all multisets over $S$. $\sigma = \langle a_1, a_2, ..., a_n \rangle \in S^*$ denotes a *sequence* of length $n$ over a set $S$.

**Definition 1 (Petri net).** *A Petri net is a triple $N = (P, T, F)$, where $P$ is the set of places, $T$ is the set of transitions, $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs (flow relation).*

Petri nets [11] is a formalism for modeling and analyzing concurrent distributed systems. As defined above, a Petri net consists of *places* and *transitions*, which correspond respectively to *conditions* and *activities* of a system. Places may contain tokens, representing resources, control threads, etc. A *marking* $m : P \to \mathbb{N}$ is a function that assigns tokens to places, denoting a system's state. The *initial marking* is denoted as $m_0$, and the change into a new marking is defined by the *firing rule*. Let $N = (P, T, F)$ be a Petri net, $X = P \cup T$, the sets ${}^\bullet x = \{y \in X | (y, x) \in F\}$ and $x^\bullet = \{y \in X | (x, y) \in F\}$ denote the *preset* and the *postset* of $x \in X$. Transition $t \in T$ is *enabled* in a *marking* $m$ iff ${}^\bullet t \subseteq m$. Then, the *firing* of $t$ leads to a *new marking* $m' = m - {}^\bullet t + t^\bullet$.

**Definition 2 (Workflow net).** *Let $N = (P, T, F)$ be a Petri net. $N$ is a workflow net (WF-net) iff $P$ contains a source place $i$ and a sink place $o$ s.t. ${}^\bullet i = o^\bullet = \emptyset$, and each node in $N$is on a path from $i$ to $o$.*

When modeling individual agents in multi-agent systems, we consider *workflow nets* [1]. A WF-net has an initial and a final state, represented by markings $m_0 = \{i\}$ and $m_f = \{o\}$. Let $N = (P, T, F)$ be a WF-net, we consider an *activity labeling* function $\delta : T \to A$, which assigns an *activity* label to each transition $t \in T$, where $A$ is a finite set of activities. We define a sequence $\sigma = \langle a_1, ..., a_n \rangle \in A^*$ as a *run* of a WF-net $N$ if there exists a firing sequence $\langle t_1, ..., t_n \rangle$ that leads from the initial marking $m_0 = \{i\}$ of $N$ to its final marking $m_f = \{o\}$ s.t. $\delta(t_i) = a_i$ $(1 \leq i \leq n)$. The set of all possible runs of a WF-net $N$ is denoted by $\mathcal{B}(N)$ and is called the *behavior* of $N$.

For modeling complex systems, one can use colored Petri nets (CP-nets). In CP-nets, tokens are attached with values belonging to different domains (*color* types). Let $\mathcal{U}$ be the set of these different domains. Then, each place in a CP-net is typed with a domain in $\mathcal{U}$ indicating the type of tokens it contains. Arcs in CP-nets are annotated with expressions from a language $Expr$ defined over $Atom = V \cup C$, where $V$ and $C$ are sets of variables and constants. $Expr$ is defined as follows: $(i)$ An $atom \in Atom$ is an expression in $Expr$, $(ii)$ if $e_1$, $e_2 \in Expr$, then $(e_1 + e_2)$ is an expression in $Expr$.

**Definition 3 (Colored Petri net).** *A colored Petri net is a tuple $CPN = (P, T, F, \texttt{type}, W)$ where:*

- *$(P, T, F)$ is a Petri net;*
- *$\texttt{type} : P \to \mathcal{U}$ is a place-typing function, mapping each place to a type in $\mathcal{U}$;*
- *$W : F \to Expr$ is an arc expression function. $\forall r \in F$, if $r$ is adjacent to a place $p \in P$, then the type of $W(r)$ corresponds to the type of $p$.*

Let $CPN = (P, T, F, \texttt{type}, W)$ be a CP-net over a set of domains $\mathcal{U}$. A marking $M$ in CPN is a function that maps each place $p \in P$ into a multiset of tokens $M(p) \in \mathbb{P}_m(\texttt{type}(p))$. For a CPN, we distinguish an initial marking $M_0$ and a set of final markings $\Omega$. A binding $b$ of a transition $t$ is a function that assigns a value $b(v)$ to each variable $v$ occurring in the expression of an arc adjacent to $t$. For each variable $v$, $b(v) \in \bigcup_{Q \in \mathcal{U}} Q$. A pair $(t, b)$, where $b$ is a binding of $t$, is called a *binding element*. An evaluation $W(p, t)(b)$ determines token demands (multiset of tokens) on $p$ for $t$ to be enabled with the binding $b$, and the multiset of tokens that the transition $t$ removes from the place $p$ when $t$ occurs with the binding $b$. $W(t, p)(b)$ determines the multiset of tokens added to an output place $p$. A transition is enabled in a marking $M$ w.r.t a binding $b$ iff for all $p \in P$, $W(p, t)(b) \subseteq M(p)$. An enabled transition fires in a marking $M$ yielding a new marking $M'$, such that for all places $p$, $M'(p) = (M(p) \setminus W(p, t)(b)) \cup W(t, p)(b)$.

In the following we consider a subclass of nested Petri nets (NP-nets). A NP-net $NP$ consists of a colored Petri net called the *system net* $SN$, and a set of WF-nets $\mathcal{N} = \{E_1, ..., E_n\}$ called *element nets*, which define types of net tokens.

In a system net $SN$, places contain either a set of *net tokens* or a multiset of *atomic colored tokens*. A net token is a marked element net, whereas an atomic colored token is a data value of some domain $D \in \mathcal{D}$, where $\mathcal{D}$ is a finite set of domains. Regarding the system net, we consider a language of expressions $Expr$ defined over $Atom = V \cup C$, where: (i) $V$ is a finite set of variables, typed over the set of element nets $\mathcal{N}$ and data domains $\mathcal{D}$ (e.g., the type of $x \in V$ is $E_1$) and (ii) $C$ is a finite set of constants, typed only over the set of data domains $\mathcal{D}$. Each arc $r$ is supplied with an arc expression from $Expr$. This arc expression can be either: a sum of variables typed over $\mathcal{N}$ if $\texttt{type}(p) \in \mathcal{P}(\mathcal{N})$ where $p$ is a place adjacent to arc $r$ containing net tokens, or an arbitrary sum of distinct variables and constants typed over $\mathcal{D}$ if $\texttt{type}(p) \in \mathcal{D}$ where $p$ is a place adjacent to arc $r$ containing atomic colored tokens.

**Definition 4 (Nested Petri net).** *Let $\mathcal{D}$ be a finite set of domains, Lab — a finite set of synchronization labels and $A$ — a finite set of activities. A nested Petri net (NP-net) is a tuple $NP = (SN, (E_1, ..., E_k), \lambda, \delta)$, where:*

- *$SN = (P_{SN}, T_{SN}, F_{SN}, \texttt{type}, W)$ is a colored Petri net (called a system net) with two sets of places $P_{Net}$ and $P_{Atom}$ ($P_{Net} \cup P_{Atom} = P_{SN}$), such that for all $p \in P_{Net}$, $\texttt{type}(p) \subseteq \{E_1, ..., E_k\}$ and for all $p \in P_{Atom}$, $\texttt{type}(p) \in \mathcal{D}$;*
- *$for\ i = \overline{1,k}$, $E_i = (P_i, T_i, F_i)$ is a WF-net, called an element net, s.t. $(T_{SN} \cup T_i) \cap (P_{SN} \cup P_i) = \emptyset$;*
- *$\lambda : T_{NP} \nrightarrow$ Lab is a (partial) synchronization labeling function, where $T_{NP} = T_{SN} \cup T_1 \cup ... \cup T_k$;*
- *$\delta : T_{NP} \to A$ is an activity labeling function, s.t. $for\ i = \overline{1,k}$ $T_{SN} \cap T_i = \emptyset$.*

In what follows, we consider *conservative* NP-nets [9]. In a conservative NP-net $NP = (SN, (E_1, ..., E_k), \lambda, \delta)$, net tokens cannot be *cloned* or *disappear*. In a run, there is a *stable set* of net tokens which we distinguish using individual *agent names*. Let $R$ be a set of agent names, we propose a function $\texttt{class} : R \to \{E_1, ..., E_k\}$, which maps to each agent name $r$ an element net. We denote by $(r, m)$ a net token which is characterized by an individual agent name $r$ with the corresponding element net $\texttt{class}(r)$ and a marking $m$. The set of all possible net tokens is denoted by $S_{\texttt{agent}}$.

A marking $M$ in a NP-net $NP$ is a function mapping each place $p \in P_{SN}$ to a subset of $S_{\texttt{agent}}$ or a multiset over a domain $D \in \mathcal{D}$, in accordance with the type of $p$. Hence, elements in $M(p)$ are either distinguishable net tokens or atomic colored tokens which can be repeated. We say that a net token $(r, m)$ *resides* in a place $p$ (under marking $M$) if $(r, m) \in M(p)$. Thus, the marking of a NP-net is defined by the marking of its system net. For a NP-net, we distinguish an initial marking $M_0$ and a set of final markings $M_f \in \Omega$.

Let $t$ be a transition in the system net $SN$ of a NP-net, and let be ${}^\bullet t = \{p_1, ..., p_i\}$ and $t^\bullet = \{q_1, ..., q_j\}$ the sets of pre- and post-elements of transition $t$. $W(t) = \{W(p_1, t), ..., W(p_i, t), W(t, q_1), W(t, q_j)\}$ denotes the set of all arc expressions adjacent to $t$. A *binding* of $t$ is a function $b$ assigning to each variable $v$, occurring in each expression in $W(t)$, a value $b(v) \in \bigcup_{D \in \mathcal{D}} D \ \cup \ S_{\texttt{agent}}$. A transition $t$ in $SN$ is *enabled* in a marking $M$ w.r.t a binding $b$ if for all

$p \in {}^\bullet t\ W(p,t)(b) \subseteq M(p)$. An enabled transition fires in a marking $M$ yielding a new marking $M^{'}$, such that for all places $p \in P_{SN}$, $M^{'}(p) = (M(p) \backslash W(p,t)(b)) \cup W(t,p)(b)$. For net tokens from $S_{\texttt{agent}}$ serving as variable values in input arc expressions from $W(t)$, we say that they are *involved* in the firing of $t$. They are removed from input places and brought to output places of $t$.

We consider three kinds of steps in a NP-net:

*Element-autonomous step*: let $t$ be a transition without a synchronization label in a net token named $r$, i.e., $\lambda(t)$ is not defined. When $t$ is enabled in a marking $m$, an *element-autonomous step* is a firing of $t$ in marking $m$, producing a new marking $m'$, according to the usual firing rules of WF-nets. This is also written as:

$$m \xrightarrow{\delta(t),r} m'$$

*System-autonomous step*: let $t \in T_{SN}$ be a transition without a synchronization label in the system net $SN$. A *system-autonomous step* (also called a *transfer step* when net tokens are involved) is the firing of transition $t$ according to the firing rule described above for a NP-net. The occurrence of this step in a marking $M$ w.r.t a binding $b$, producing a new marking $M'$, is denoted by:

$$M \xrightarrow{\delta(t),b} M'$$

*Synchronization step*: let $t \in T_{SN}$ be a transition with a synchronization label $\lambda(t)$, and enabled in a marking $M$ w.r.t a binding $b$, and let $(r_1, m_1), ..., (r_n, m_n) \in S_{\texttt{agent}}$ be net tokens involved in the firing of $t$. Then, $t$ can fire provided that in each $(r_i, m_i)$ $(1 \leq i \leq n)$ there is an enabled transition labeled with the same value $\lambda(t)$. Thus, a *synchronization step* goes in two stages: first, the firing of transitions $t_1, ..., t_n$ in all net tokens involved in the firing of $t$, and then, the firing of $t$ in $SN$ w.r.t. a binding $b$. This step is denoted by:

$$M \xrightarrow{\delta(t),\{(\delta(t_i),r_i),1\leq i\leq n\},b} M'$$

**Definition 5 (Run, Behavior of a nested Petri net).** *Let $NP$ be a conservative nested Petri net and $\sigma$ — a sequence of steps in $NP$. The occurence of $\sigma$ from the initial marking $M_0$ of $NP$, results in some final marking $M_f \in \Omega$, is called a run. The set of all possible runs is denoted by $\mathcal{B}(NP)$ and is called the behavior of $NP$.*

## 3 Event Logs of Multi-Agent Systems

An *event log* of a multi-agent system is a multiset of traces, where a *trace* is a sequence of events. Events consist of an *activity* name, *resources* which executed the activity or were *involved* in its execution, and an (optional) multiset of data values. As possible resources we consider a system $SN$ or a finite set of agents with distinct names $r_1, r_2, ..., r_n$. As shown in table 1, we consider three event types: (1) execution of an activity $a_1$ by some resource $r_1$, (2) execution of an activity $a$ by the system $SN$ where $n$ resources (agents) are involved, or (3) the

simultaneous execution of activity $a$ by $SN$, and activities $a_1, ..., a_n$ by resources $r_1, ..., r_n$. For cases (2) and (3), events may contain $m$ *data values* used by $SN$. We proceed to formally define a trace and an event log of a multi-agent system.

Table 1: Event attributes in event logs of multi-agent systems.

| **Event type** | **Attributes** | | |
|---|---|---|---|
| *event refers to...* | **Activity** | **Resource** | **Data** |
| (1) an *element-autonomous* step | $a_1$ | $r_1$ | none. |
| (2) a *system-autonomous* step | $a$ | $(SN, \{r_1, ..., r_n\})$ | $\{d_1, ..., d_m\}$ |
| (3) a *synchronization* step | $(a, a_1, ..., a_n)$ | $(SN, r_1, ..., r_n)$ | $\{d_1, ..., d_m\}$ |

**Definition 6 (Trace, Event log of a multi-agent system).** *Let $SN$ be a system name, $S$ — a set of system activities, $D$ — a set of data, $B$ — a set of agent activities, $R_B$ — a set agent names, and $\mathcal{C}$ — a set where $C \in \mathcal{C} \Leftrightarrow C = C_1 \cup C_2 \,/\, C_1 \subseteq R_B$ and $C_2 \in \mathbb{P}_m(D)$. A trace is a sequence $\sigma \in (\mathcal{E}_{agent} \cup \mathcal{E}_{system} \cup \mathcal{E}_{sync})^*$ where $\mathcal{E}_{agent} = B \times R_B$, $\mathcal{E}_{system} = (S \times \{SN\} \times \mathcal{C})$, and $\mathcal{E}_{sync} = (S \times \{SN\} \times \mathcal{P}(B \times R_B) \times \mathbb{P}_m(D))$. $\mathcal{L} \in \mathbb{P}_m(\mathcal{E}_{agent}, \mathcal{E}_{system}, \mathcal{E}_{sync})^*$ is an event log, i.e., a multiset of traces.*

Table 2: An event log of a multi-agent system $\mathcal{L}$ in tabular form, whose expected behavior is modeled in fig. 1.

| **Trace** | **Freq.** |
|---|---|
| $\sigma_1 = \langle\, (d, r_1), (d, r_2), (h, r_1), (h, r_2), (a, SN, \{(f, r_1)\}), (a, SN, \{(f, r_2)\}), (b, SN, \{r_2\}), (b, SN, \{r_1\}) \,\rangle$ | 4 |
| $\sigma_2 = \langle\, (d, r_2), (h, r_2), (d, r_1), (h, r_1)(a, SN, \{(f, r_2)\}), (a, SN, \{(f, r_1)\}), (b, SN, \{r_1\}), (b, SN, \{r_2\}) \,\rangle$ | 1 |
| $\sigma_3 = \langle\, (d, r_2), (e, r_2), (d, r_1), (h, r_1), ((c, SN, \{(g, r_1)\}), (c, SN, \{(g, r_2)\}) \,\rangle$ | 1 |
| $\sigma_4 = \langle\, (d, r_1), (d, r_2), (h, r_2), (e, r_1), (c, SN, \{(g, r_2)\}), (c, SN, \{(g, r_1)\}) \,\rangle$ | 1 |
| $\sigma_5 = \langle\, (d, r_1), (d, r_2), (e, r_1), (e, r_2), (a, SN, \{(f, r_1)\}), (c, SN, \{(g, r_2)\}), (b, SN, \{r_1\}) \,\rangle$ | 2 |

Table 2 shows an event log $\mathcal{L}$ of the multi-agent system modeled in fig. 1. $\mathcal{L}$ contains information on nine traces. A distinct trace can occur multiple times. For instance, trace $\sigma_5$ occurred two times. It is a sequence of seven events. First, both activities $d$ and $e$ were executed by agents $r_1$ and $r_2$. Next, the system $SN$ executed two synchronization steps with agents $r_1$ and $r_2$, where activities $a$ and $f$, and later $c$ and $g$, were executed simultaneously. The trace ended by a system-autonomous step where $SN$ executed $b$ for agent $r_1$.

Let $\mathcal{L}$ be an event log of a multi-agent system and $NP$ — a nested Petri net. $\mathcal{L}$ is *syntactically correct* w.r.t. $NP$ if each event in $\mathcal{L}$ is *syntactically correct* w.r.t. a step in $NP$ where:

- An event $(a, r)$ is syntactically correct w.r.t. a step in $NP$ if there is a transition $t$ without synchronization label in a net token named $r$ where $\delta(t) = a$, and $t$ can fire in a marking $m$ producing a new marking $m'$, i.e., $m \xrightarrow{\delta(t), r} m'$ in $NP$ where $t \in E_i, 1 \leq i \leq k$, and $\texttt{class}(E_i) = r$.

– An event $(a, SN, d_1, ..., d_m)$ is syntactically correct w.r.t. a step in $NP$ if there is a transition $t \in T_{SN}$ without synchronization label where $\delta(t) = a$, and $t$ can fire in a marking $M$ w.r.t. a binding $b$ such that $b$ assigns the atomic tokens $d_1, ..., d_m$ to the variables in $W(t)$, i.e., $M \xrightarrow{\delta(t),b} M'$ in $NP$.
– An event $(a, SN, \{(a_1, r_1), ..., (a_q, r_q)\}, \{d_1, ..., d_p\})$ is syntactically correct w.r.t. a step in $NP$ if there is a transition $t \in T_{SN}$ with synchronization label $\lambda(t)$ where $\delta(t) = a$, $(r_1, m_1), ..., (r_n, m_n)$ are net tokens involved in the firing of $t$ such that in each $(r_i, m_i)$ there is an enabled transition $t_i$ $(\delta(t_i) = a_i)$ labeled with the same value $\lambda(t)$, and $t$ can fire in a marking $M$ w.r.t. a binding $b$ assigning the atomic tokens $d_1, ..., d_p$ and also the agent names $r_1, ..., r_q$ to the variables in $W(t)$, i.e., $M \xrightarrow{\delta(t),\{(\delta(t_i),r_i),1\leq i\leq n\},b} M'$ in $NP$.

## 4 Compositional Conformance Checking of Nested Petri Nets and Event Logs of Multi-Agent Systems

In this section, we propose a solution to check conformance of event logs of multi-agent systems and nested Petri nets. We prove that an event log perfectly fits a NP-net iff the event log is syntactically correct w.r.t. that NP-net, and each projection of the event log onto a NP-net component perfectly fits that component.
Let $\langle\rangle$ denote an empty sequence, $\sigma_1.\sigma_2$ — concatenation of two sequences, and $\sigma{\restriction}_e$ — the *projection* of sequence $\sigma$ on an element $e$.

**Definition 7 (Trace projection onto an agent).** *Let $X = (\mathcal{E}_{agent} \cup \mathcal{E}_{system} \cup \mathcal{E}_{sync})$ be a set of events, $B$ — a set of agent activities, and $R_B$ — a set of agent names. ${\restriction}_{r \in R_B} \in X^* \longrightarrow B^*$ is a projection function defined recursively: (1) $\langle\rangle{\restriction}_{r \in R_B} = \langle\rangle$ (2) For $\sigma \in X^*$ and $e \in X$:*
– *If $e = (x, y) \in \mathcal{E}_{agent}, y = r$, then $(\langle e\rangle.\sigma){\restriction}_{r \in R_B} = \langle x\rangle.\sigma{\restriction}_{r \in R_B}$;*
– *If $e = (x, SN, \{(x_1, y_1), ..., (x_n, y_n)\}, \{d_1, ..., d_p\}) \in \mathcal{E}_{sync}, y_i = r, 1 \leq i \leq n$, then $(\langle e\rangle.\sigma){\restriction}_{r \in R_B} = \langle x_i\rangle.\sigma{\restriction}_{r \in R_B}$;*
– *Otherwise $(\langle e\rangle.\sigma){\restriction}_{r \in R_B} = \sigma{\restriction}_{r \in R_B}$ .*

**Definition 8 (Trace projection onto a system net).** *Let $X = (\mathcal{E}_{agent} \cup \mathcal{E}_{system} \cup \mathcal{E}_{sync})$ and $Y \subseteq S \times \mathcal{C}$ where $S$ is a set of system activities, $C \in \mathcal{C} \Leftrightarrow C = C_1 \cup C_2 / C_1 \subseteq R_B$ and $C_2 \in \mathbb{P}_m(D)$, $R_B$ is a set of agent names, and $D$ is a set of data. ${\restriction}_{SN} \in X^* \longrightarrow Y^*$ is a projection function defined recursively: (1) $\langle\rangle{\restriction}_{SN} = \langle\rangle$ (2) For $\sigma \in X^*$ and $e \in X$:*
– *If $e = (x, SN, \{d_1, ..., d_p\}) \in \mathcal{E}_{system}$, then $(\langle e\rangle.\sigma){\restriction}_{SN} = \langle(x, \{d_1, ..., d_p\})\rangle.\sigma{\restriction}_{SN}$;*
– *If $e = (x, SN, \{(x_1, y_1), ..., (x_n, y_n)\}, \{d_1, ..., d_p\}) \in \mathcal{E}_{sync}$, then $(\langle e\rangle.\sigma){\restriction}_{SN} = \langle(x, \{y_1, ..., y_n, d_1, ..., d_p\})\rangle.\sigma{\restriction}_{SN}$;*
– *Otherwise $(\langle e\rangle.\sigma){\restriction}_{SN} = \sigma{\restriction}_{SN}$ .*

Def. 7 (resp. Def. 8) is used to project traces of an event log onto net tokens (resp. a system net). A projection of a trace onto a net token yields the sequence

of agent activities. A projection onto the system net yields a sequence of pairs where each pair consists of an activity and the set of resources and data involved in this activity execution. For instance, consider the projection of the event log $\mathcal{L}$ (cf. Table 2) onto components of the NP-net $N$ (cf. Fig. 1). Table 3 shows the three decomposed event logs $L_{SN}$, $L_1$, and $L_2$ resulting from the projection of $\mathcal{L}$ onto (a) the system net, (b) agent $r_1$, and (c) agent $r_2$ respectively.

Table 3: Projections $L_{SN}$, $L_1$, and $L_2$ from the event log $\mathcal{L}$ (cf. Table 2) onto (a) the system net $SN$, and agents (b) $r_1$ and (c) $r_2$ from $N$ (cf. Fig. 1).

(a) $L_{SN}$

| **Trace** | **Freq.** |
|---|---|
| $\langle(a,\{r_1\}),(a,\{r_2\}),(b,\{r_2\}),(b,\{r_1\})\rangle$ | 4 |
| $\langle(a,\{r_2\}),(a,\{r_1\}),(b,\{r_1\}),(b,\{r_2\})\rangle$ | 1 |
| $\langle(c,\{r_1\}),(c,\{r_2\})\rangle$ | 1 |
| $\langle(c,\{r_2\}),(c,\{r_1\})\rangle$ | 1 |
| $\langle(a,\{r_1\}),(c,\{r_2\}),(b,\{r_1\})\rangle$ | 2 |

(b) $L_1$

| **Trace** | **Freq.** |
|---|---|
| $\langle d,h,f\rangle$ | 5 |
| $\langle d,h,g\rangle$ | 1 |
| $\langle d,e,g\rangle$ | 1 |
| $\langle d,e,f\rangle$ | 2 |

(c) $L_2$

| **Trace** | **Freq.** |
|---|---|
| $\langle d,h,f\rangle$ | 5 |
| $\langle d,e,g\rangle$ | 3 |
| $\langle d,h,g\rangle$ | 1 |

Thus, a conformance checking technique can be applied to each projection and the corresponding NP-net component, ignoring their synchronization labels. In particular, for the system net, we replace net tokens by their agent names, which are atomic colored tokens. We consider synchronization steps as autonomous steps, and for a marking $M$ in a NP-net $NP$, marking projections onto $NP$ components are defined as follows: (1) the projection of $M$ onto a system net $SN$, denoted as $M{\restriction}_{SN}$, is a marking of the colored Petri net $SN$ obtained by replacing all net tokens in $M$ by their agent names, and (2) the projection of $M$ onto a net token $(r,m)$, denoted as $M{\restriction}_{(r,m)}$, is just $m$. A system net component $SN$, with a marking $M{\restriction}_{SN}$ and without synchronization labels, is a CP-net labeled by activity names. A sequence of binding elements $\langle(t_1,b_1),...,(t_n,b_n)\rangle$, starting from the initial marking $M_0{\restriction}_{SN}$ and ending in a final marking $M_f{\restriction}_{SN}$, projected onto the set of system net activities is called a *run*.

**Definition 9 (Perfectly fitting event log).** *Let $\mathcal{L}$ be an event log of a multi-agent system and $NP$ — a nested Petri net. $\mathcal{L}$ perfectly fits $NP$ if and only if for all $\sigma = \langle e_1,...,e_n\rangle \in \mathcal{L}$ there is a run $\sigma' = \langle s_1,...,s_n\rangle \in \mathcal{B}(NP)$ such that for $i = \overline{1,n}$, $e_i$ is syntactically correct w.r.t $s_i$.*

Let $NP$ be a nested Petri net, $\mathcal{L}$ — an event log, $(r_1,m_1),...,(r_n,m_n)$ — net tokens of $NP$, $L_1,...,L_n$ — corresponding projections of $\mathcal{L}$, and $L_{SN}$ — a projection of $\mathcal{L}$ onto actions of the system net. $L_{SN}$ perfectly fits $SN$ if and only if for all $\sigma = \langle e_1,...,e_m\rangle \in L_{SN}$, there is a run $\sigma'$ in the system net component where

$$\sigma' = (M_0 \xrightarrow{\delta(t_1),b_1} M_1 \xrightarrow{\delta(t_2),b_2} M_2 ... M_{m-1} \xrightarrow{\delta(t_m),b_m} M_f)$$

and for $i = \overline{1,m}$, $e_i = (a_i,\{d_1^i,...,d_p^i\})$, $\delta(t_i) = a_i$, and $b_i$ is the binding assigns $\{d_1^i,...,d_p^i\}$ to the variables in $W(t_i)$. For $i = \overline{1,n}$, $L_i$ perfectly fits $(r_i,m_i)$ if and

only if for all $\sigma = \langle e_1, ..., e_m \rangle \in L_i$, there is a run $\sigma'$ in the element net $\texttt{class}(r_i)$ where

$$\sigma' = (m_0 \xrightarrow{\delta(t_1)} m_1 \xrightarrow{\delta(t_2)} m_2 ... m_{m-1} \xrightarrow{\delta(t_m)} m_f)$$

and for $j = \overline{1,m}$, $e_j = a_j$ and $\delta(t_j) = a_j$.

An event log perfectly fits a model if all traces in the log can be replayed on the model from beginning to end. For instance, let us consider the event log $\mathcal{L}$ (cf. Table 2) and the NP-net $N$ depicted in fig. 1. Clearly, $\mathcal{L}$ perfectly fits $NP$. Also, each projected event log $L_{SN}$, $L_1$, or $L_2$ (cf. Table 3) perfectly fits the corresponding component in $NP$.

**Theorem 1.** *Given a nested Petri net $NP = (SN, (E_1, ..., E_k), \lambda, \delta)$ and an event log $\mathcal{L} \in \mathbb{P}_m(\mathcal{E}_{\texttt{agent}}, \mathcal{E}_{\texttt{system}}, \mathcal{E}_{\texttt{sync}})^*$, let $(r_1, m_1), ..., (r_n, m_n)$ be net tokens of $NP$, $L_1, ..., L_n$ – corresponding projections of $\mathcal{L}$, and $L_{SN}$ – a projection of $\mathcal{L}$ onto the system net $SN$. $\mathcal{L}$ perfectly fits $NP$ if and only if:*

1. *$\mathcal{L}$ is syntactically correct w.r.t $NP$;*
2. *$L_{SN}$ perfectly fits $SN$;*
3. *$L_i$ perfectly fits $(r_i, m_i)$, $1 \leqslant i \leqslant n$.*

*Proof.* Let $\mathcal{L}$ be an event log of a multi-agent system, $NP = (SN, (E_1, ..., E_k), \lambda, \delta)$ — a nested Petri net, and $(r_1, m_1), ..., (r_n, m_n)$ — net tokens of $NP$.

($\Rightarrow$) Let $\sigma = \langle e_1, ..., e_m \rangle \in \mathcal{L}$ be such that there is a run $\sigma' = \langle s_1, ..., s_m \rangle \in \mathcal{B}(NP)$ and for $j = \overline{1,m}$:

- if $e_i = (a, r)$, then $s_i = m \xrightarrow{\delta(t), r} m'$ where $t \in E_i, 1 \leq i \leq k, \texttt{class}(E_i) = r$, and $\delta(t) = a$. **(1)**
- if $e_i = (a, SN, d_1, ..., d_p)$, then $s_i = M \xrightarrow{\delta(t), b} M'$ where $\delta(t) = a$, and $b$ is a binding assigning the atomic tokens $d_1, ..., d_p$ to the variables in $W(t)$. **(2)**
- if $e_i = (a, SN, \{(a_1, r_1), ..., (a_q, r_q)\}, \{d_1, ..., d_p\})$, then $s_i = M \xrightarrow{\delta(t), \{(\delta(t_i), r_i), 1 \leq i \leq q\}, b} M'$ where $\delta(t) = a,$, $\delta(t_i) = a_i, 1 \leq i \leq q$, and $b$ is a binding assigning the atomic tokens $d_1, ..., d_p$ and also the agent names $r_1, ..., r_q$ to the variables in $W(t)$. **(3)**

i.e., $\sigma$ perfectly fits $NP$. We need to prove that

- $\sigma$ is syntactically correct w.r.t $NP$;
- $\sigma \restriction_{SN} = \langle e^s_1, ..., e^s_{m'} \rangle$ perfectly fits the system net component, i.e., there is a run $\sigma_{SN}$ in the system net component where:

  $$\sigma_{SN} = (M_0 \xrightarrow{\delta(t_1), b_1} M_1 \xrightarrow{\delta(t_2), b_2} M_2 ... M_{m'-1} \xrightarrow{\delta(t_{m'}), b_{m'}} M_f)$$

  , and for $i = \overline{1,m'}$, $e^s_i = (a_i, \{d^i_1, ..., d^i_p\})$, $\delta(t_i) = a_i$, and $b_i$ is the binding that assigns $\{d^i_1, ..., d^i_p\}$ to the variables in $W(t_i)$;

– for $i = \overline{1,n}$, $\sigma \upharpoonright_{r_i} = \langle a^i_1, ..., a^i_{m''} \rangle$ perfectly fits $(r_i, m_i)$, i.e., there is a run $\sigma_{r_i}$ in the element net $\texttt{class}(r_i)$ where:

$$\sigma_{r_i} = (m_0 \xrightarrow{\delta(t_1)} m_1 \xrightarrow{\delta(t_2)} m_2 ... m_{m-1} \xrightarrow{\delta(t_{m''})} m_f)$$

and for $j = \overline{1,m''}, \delta(t_j) = a^i_j$.

By the fact that $\sigma$ perfectly fits $NP$, it follows trivially that $\sigma$ is syntactically correct w.r.t $NP$ (cf. Def. 9).

Taking into account that $\sigma' = \langle s_1, ..., s_m \rangle$ is a run in $NP$ (which can hold synchronization labels) where for $i = \overline{1,m}$ we have (1), (2) and (3), and that

$$\langle e_i \rangle \upharpoonright_{SN} = \begin{cases} \langle\rangle, & if\ e_i = (a, r) \\ \langle (a, \{d_1, ..., d_p\}) \rangle, & if\ e_i = (a, SN, d_1, ..., d_p) \\ \langle (a, \{r_1, ..., r_q, d_1, ..., d_p\}) \rangle, & if\ e_i = (a, SN, \{(a_1, r_1), ..., (a_q, r_q)\}, \{d_1, ..., d_p\}) \end{cases} \quad \textbf{(4)}$$

(cf. Def. 8) we deduce that for $\sigma \upharpoonright_{SN} = \langle e^s_1, ..., e^s_{m'} \rangle$, there is a run $\sigma_{SN}$ in the system net component where:

$$\sigma_{SN} = (M_0 \xrightarrow{\delta(t_1), b_1} M_1 \xrightarrow{\delta(t_2), b_2} M_2 ... M_{m'-1} \xrightarrow{\delta(t_{m'}), b_{m'}} M_f)$$

and for $i = \overline{1,m'}$, $e^s_i = (a_i, \{d^i_1, ..., d^i_p\})$, $\delta(t_i) = a_i$, and $b_i$ is the binding that assigns $\{d^i_1, ..., d^i_p\}$ to the variables in $W(t_i)$. Therefore, $\sigma \upharpoonright_{SN}$ perfectly fits the system net component.

Now by the fact that $\sigma'$ is a run in $NP$ (which can hold synchronization labels) where for $i = \overline{1,m}$ we have (1), (2) and (3), and that for $j = \overline{1,n}$

$$\langle e_i \rangle \upharpoonright_{r_j} = \begin{cases} \langle a \rangle, & if\ e_i = (a, r_j) \\ \langle a_j \rangle, 1 \leq j \leq q, & if\ e_i = (a, SN, \{(a_1, r_1), ..., (a_q, r_q)\}, \{d_1, ..., d_p\}) \\ \langle\rangle, & otherwise \end{cases} \quad \textbf{(5)}$$

(cf. Def. 7) it follows that for $\sigma \upharpoonright_{r_i} = \langle a_1, ..., a_{m''} \rangle, 1 \leq i \leq n$, there is a run $\sigma_{r_i}$ in the element net $\texttt{class}(r_i)$ where:

$$\sigma_{r_i} = (m_0 \xrightarrow{\delta(t_1)} m_1 \xrightarrow{\delta(t_2)} m_2 ... m_{m-1} \xrightarrow{\delta(t_{m''})} m_f)$$

and for $j = \overline{1,m''}, \delta(t_j) = a_j$. Therefore, $\sigma \upharpoonright_{r_i}$ perfectly fits $(r_i, m_i)$.

($\Leftarrow$) Let $\sigma = \langle e_1, ..., e_m \rangle \in \mathcal{L}$ be such that $\sigma$ is syntactically correct w.r.t $NP$, $\sigma \upharpoonright_{SN}$ perfectly fits the system net component, and for $i = \overline{1,n}$, $\sigma \upharpoonright_{r_i}$ perfectly fits $(r_i, m_i)$. We need to prove that $\sigma$ perfectly fits $NP$.

Taking into account (4) and (5), and that $\sigma$ is syntactically correct w.r.t $NP$, we deduce that by associating to each element of the projected sequences the corresponding resource, elements can be merged together into the trace $\sigma$. Therefore, being $\sigma \upharpoonright_{SN}$ perfectly fits the system net component and for $i = \overline{1,n}$ $\sigma \upharpoonright_{r_i}$ perfectly fits $(r_i, m_i)$, then $\sigma$ perfectly fits $NP$ and this concludes the proof.

## 5 Conclusions and Future Work

In this paper, we proposed a compositional approach for conformance checking of nested Petri nets and event logs of multi-agent systems. Nested Petri nets are a well-known Petri net extension where tokens can be Petri nets themselves, allowing to model multi-agent systems. An event log can be projected onto NP-net components (system net and all agents), so conformance checking can be performed between each projection and the corresponding component. This approach can provide specific conformance diagnostics for each system component. We demonstrated the validity of our approach proving that, an event log perfectly fits a nested Petri net if and only if it is syntactically correct w.r.t the model and each projection perfectly fits the corresponding model component. For future research, we consider the experimental evaluation of our approach against other approaches when checking conformance of multi-agent systems.

## References


1. van der Aalst, W.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers **08**(01), 21–66 (1998)
2. van der Aalst, W.: Process Mining: Data Science in Action. Springer, 2nd edn. (2016)
3. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking: Relating Processes and Models. Springer (2018)
4. Hee, van, K., Oanea, O., Serebrenik, A., Sidorova, N., Voorhoeve, M., Lomazova, I.: Checking properties of adaptive workflow nets. Fundamenta Informaticae **79**(3-4), 347–362 (2007)
5. Itkin, I.: Mind the gap between testing and production: applying process mining to test the resilience of exchange platforms (2019). `https://tinyurl.com/y55sndcv`
6. Lee, W., Verbeek, H., Munoz-Gama, J., van der Aalst, W., Sepúlveda, M.: Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining. Information Sciences **466**, 55–91 (2018)
7. Lomazova, I.A.: Nested Petri Nets - a Formalism for Specification and Verification of Multi-Agent Distributed Systems. Fundamenta Informaticae **43**, 195–214 (2000)
8. Lomazova, I.A.: Nested Petri Nets for Adaptive Process Modeling, vol. 4800, pp. 460–474. Springer (2008)
9. Lomazova, I.A., Ermakova, V.O.: Verification of Nested Petri Nets Using an Unfolding Approach. In: Cabac, L., Kristensen, L.M., Rölke, H. (eds.) Petri Nets and Software Engineering. CEUR Workshop Proceedings, vol. 1591 (2016)
10. Munoz-Gama, J., Carmona, J., Van Der Aalst, W.: Single-entry single-exit decomposed conformance checking. Information Systems **46**, 102–122 (2014)
11. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
12. van der Aalst, W.: Business alignment: using process mining as a tool for delta analysis and conformance testing. Requirements Engineering **10**, 198–211 (2005)
13. van der Aalst, W.: Decomposing petri nets for process mining : a generic approach. Distributed and Parallel Databases **31**(4), 471–507 (2013)
14. van der Aalst, W., van Hee, K.M., van der Werf, J.M., Verdonk, M.: Auditing 2.0: Using process mining to support tomorrow's auditor. Computer **43**(3), 90–93 (2010)