# Verification of Nested Petri Nets Using an Unfolding Approach*

Irina A. Lomazova, and Vera O. Ermakova

National Research University Higher School of Economics,
Myasnitskaya ul. 20, 101000 Moscow, Russia
ilomazova@hse.ru voermakova@edu.hse.ru

**Abstract.** Nested Petri nets (NP-nets) is an extension of the Petri nets formalism within the "nets-within-nets" approach, allowing to model systems of interacting dynamic agents in a natural way. One of the main problems in verifying of such systems is the State Explosion Problem. To tackle this problem for highly concurrent systems the unfolding method has proved to be very helpful. In this paper we continue our research on applying unfoldings for NP-nets verification and compare unfolding of NP-net translated into classical Petri net with direct component-wise unfolding.

**Keywords:** Multi-agent systems, verification, Petri nets, nested Petri nets, unfoldings.

## 1 Introduction

Multi-agent systems have been studied explicitly for the last decades and can be regarded as one of the most advanced research and development area in computer science today. They are used in various practical fields and areas, such as artificial intelligence, cloud services, grid systems, augmented reality systems with interactive environment objects, information gathering, mobile agent cooperation, sensor information and communication.

Petri nets have been proved to be one of the best formalisms for modeling and analysis of distributed systems. However, due to the flat structure of classical Petri nets, they are not so good for modeling complex multi-agent systems. For such systems a special extension of Petri nets, called nested Petri nets [1], can be used. Nested Petri nets follow 'nets-within-nets' approach [2] and naturally represent multi-agent systems structure, because tokens in the main system net are Petri nets themselves, and can have their own behavior.

To check nested Petri net model properties one of the most popular verification method, *model checking*, could be used. The basic idea of model checking is to build a reachability (transition) graph and check properties on this graph.

* This work is supported by the Basic Research Program at the National Research University Higher School of Economics and Russian Foundation for Basic Research, project No.16-01-00546.

However, there is a crucial problem for verification of highly concurrent systems using model checking approach — a large number of interleavings of concurrent processes. This leads to the so-called state-space explosion problem.

To tackle this problem unfolding theory [3,4] was introduced. In [5] applicability of unfoldings for nested Petri nets was studied and the method for constructing unfoldings for safe conservative nested Petri nets was proposed. It was proven there, that unfoldings for nested Petri nets satisfy the unfoldings fundamental property, and thus can be used for verification of conservative nested Petri nets similar to the classical unfoldings methods. Classical unfoldings are defined for P/T nets, but in this paper we deal with a restricted subclass of nested Petri nets — *conservative safe nested Petri nets*. This means that net tokens, representing agents, cannot be destroyed or created, but can change their location in the system net and can change their inner states. Thus, the number of agents is constant and each agent can be identified. It was shown in [5] that for conservative safe nested Petri nets unfoldings can be constructed in a component-wise manner, what makes practical verification of such models feasible.

However, safe conservative nested Petri nets are bounded. So, for such net it is possible to construct a P/T net with equivalent behavior, for which the standard unfolding techniques can be applied. Then the question is whether direct unfolding proposed in [5] is really better than constructing unfoldings via translation of nested Petri nets into safe P/N nets in terms of time complexity.

In this paper we study this question. For that we develop an algorithm for translating a safe conservative NP-net into a behaviorally equivalent P/T net. We prove that the reachability graphs of a source NP-net and the obtained P/T net are isomorphic, and hence both unfolding methods give the same (up to isomorphism) result. From general considerations translating an NP-net into a P/T net and then constructing unfoldings will be more time consuming, than constructing unfoldings directly. To check whether this time gap reveals itself in practice we implement all the algorithms and compare both methods experimentally.

*Related Work* Nested Petri nets (NP-nets) are widely used in modeling of distributed systems [6,7,8], serial or reconfigurable systems [9,10,11], protocol verification [12], coordination of sensor networks with mobile agents [13], innovative space system architectures [14], grid computing [15].

Several methods for NP-nets behavioral analysis were proposed in the literature, among them compositional methods for checking boundedness and liveness for nested Petri nets [16], translation of NP-nets into Colored Petri nets in order to verify them with CPNtools [17], verification of a subclass of recursive NP-nets with SPIN [18].

Unfolding approach and state-space explosion problem are explicitly studied in the literature. The original development in *unfoldings* (of P/T-nets) is due to [19]. McMillan [3] was the first to use unfoldings for verification. He introduced the concept of *complete finite prefixes* of unfoldings, and demonstrated the applicability of this approach to the verification of asynchronous circuits.

The original McMillan's algorithm was used to solve the *executability* problem — to check whether a given transition can fire in the net. This algorithm can be used also for checking deadlock-freedom and for solving some other problems. Later, numerous improvements to the algorithm have been proposed ([20,21,22] to name a few); and the approach has been applied to high-level Petri nets [23], process algebras [24] and M-nets [23].

The general method for truncating unfoldings, which abstracts from the information one wants to preserve in the finite prefix of the unfolding, was proposed in [25,26]. This method is based on the notion of a *cutting context*. We use this approach for defining branching processes and unfoldings of conservative nested Petri nets.

The paper is organized as follows. In Section 2 we present the basic notions of Petri nets, nested Petri nets, and classical unfoldings. In Section 3 an algorithm for nested Petri nets into P/T nets translation is described. In Section 4 direct unfoldings for safe conservative NP-nets are defined and compared with constructing unfoldings via into P/T nets translation. The last section gives some conclusions.

## 2 Preliminaries

*Multisets.* Let $S$ be a finite set. A *multiset* $m$ over a set $S$ is a function $m : S \to Nat$, where $Nat$ is the set of natural numbers (including zero), in other words, a multiset may contain several copies of the same element.

For two multisets $m, m'$ we write $m \subseteq m'$ iff $\forall s \in S : m(s) \leqslant m'(s)$ (the inclusion relation). The sum and the union of two multisets $m$ and $m'$ are defined as usual: $\forall s \in S : (m+m')(s) = m(s)+m'(s),\ (m \cup m')(s) = max(m(s), m'(s))$.

### 2.1 P/T-nets

Let $P$ and $T$ be two finite disjoint sets of *places* and *transitions* and let $F \subseteq (P \times T) \cup (T \times P)$ be a *flow relation*. Then $N = (P, T, F)$ is called a *P/T-net*.

A *marking* in a P/T-net $N = (P, T, F)$ is a multiset over the set of places $P$. By $\mathcal{M}(N)$ we denote a set of all markings in $N$. A *marked P/T-net* $(N, M_0)$ is a P/T-net together with its *initial marking* $M_0$.

Pictorially, $P$-elements are represented by circles, $T$-elements by boxes, and the flow relation $F$ by directed arcs. Places may carry tokens represented by filled circles. A current marking $m$ is designated by putting $m(p)$ tokens into each place $p \in P$.

For a transition $t \in T$, an arc $(x, t)$ is called an *input arc*, and an arc $(t, x)$ — an *output arc*. For each node $x \in P \cup T$, we define the *pre-set* as ${}^\bullet x = \{y \mid (y, x) \in F\}$ and the post-set as $x^\bullet = \{y \mid (x, y) \in F\}$.

We say that a transition $t$ in a P/T-net $N = (P, T, F)$ is *enabled* at a marking $M$ iff ${}^\bullet t \subseteq M$. An enabled transition may *fire*, yielding a new marking $M' = M - {}^\bullet t + t^\bullet$ (denoted $M \xrightarrow{t} M'$). A marking $M$ is called *reachable* if there exists

a (possibly empty) sequence of firings $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \to \cdots \to M$ from the initial marking to $M$. By $\mathcal{RM}(N, M_0)$ we denote the set of all reachable markings in $(N, M_0)$.

A marking $M$ is called *safe* iff for all places $p \in P$ we have $M(p) \leqslant 1$. A marked P/T-net $N$ is called *safe* iff every reachable marking $M \in \mathcal{RM}(N, M_0)$ is safe. A *reachability graph* of a P/T-net $(N, M_0)$ presents detailed information about the net behavior. It is a labeled directed graph, where vertices are reachable markings in $(N, M_0)$, and an arc labeled by a transition $t$ leads from a vertex $v$, corresponding to a marking $M$, to a vertex $v'$, corresponding to a marking $M'$ iff $M \xrightarrow{t} M'$ in $N$.

### 2.2 Classical Petri Nets Unfoldings

*Branching processes and unfoldings of P/T-nets.* Unfoldings are used to define non-sequential (true concurrent) semantics of P/T-nets, and complete prefixes of unfoldings are used for verification. Here we give necessary basic notions and definitions, connected with unfoldings. Further details can be found in [27,28].

Let $N = (P, T, F)$ be a P/T-net. The following relations are defined on the set $P \cup T$ of nodes in $N$:

1. the *causality* relation, denoted as $<$, is the transitive closure of $F$, and $\leqslant$ is the reflexive closure of $<$; if $x < y$, we say that $y$ causally depends on $x$.
2. the *conflict* relation, denoted as $\#$: nodes $x, y \in P \cup T$ are in conflict iff $\exists t, t' \in T, t \neq t' \wedge {}^\bullet t \cap {}^\bullet t' \neq \varnothing \wedge t \leqslant x \wedge t' \leqslant y$;
3. the *concurrency* relation, denoted as *co*: two nodes are *concurrent* if they are not in conflict and neither of them causally depends on the other.

For a set $B$ of nodes we write $co\,(B)$ iff all nodes in $B$ are pairwise concurrent.

An *occurrence net* is a safe P/T-net $ON = (B, E, G)$ s.t.

1. $ON$ is acyclic;
2. $\forall p \in B\colon |{}^\bullet p| \leqslant 1$;
3. $\forall x \in B \cup E$ the set $\{y \mid y < x\}$ is finite, i.e., each node in $ON$ has a finite set of predecessors;
4. $\forall x \in B \cup E\colon \neg(x \# x)$, i.e., no node is in self-conflict.

In occurrence nets, elements from $B$ are usually called *conditions* and elements from $E$ are called *events*.

A *configuration* $C$ in an occurrence net $ON = (B, E, G)$ is a non-conflicting subset of nodes, which is downwards-closed under $<$, i.e., $\forall x, y \in C\colon \neg(x \# y)$, and $(x < y) \wedge y \in C$ implies $x \in C$. For each $x \in B \cup E$ we define a *local configuration of* $x$ to be $[x] = \{y \mid y \in B \cup E, y < x\}$. The definition of a local configuration can be straightforwardly generalized to any non-conflicting set of nodes $X \subseteq B \cup E$, namely $[X] = \{y \mid y \in B \cup E, x \in X, y < x\}$.

We define the set of branching processes of a given marked P/T-net $N = (P, T, F, M_0)$ using the so-called *canonical representation*.

The set $\mathcal{C}$ of *canonical names* for $N$ is defined recursively to be the smallest set s.t. if $x \in P \cup T$ and $A$ is a finite subset of $\mathcal{C}$, then $(A, x) \in \mathcal{C}$.

A $\mathcal{C}$-Petri net is an occurrence net $(B, E, G)$ such that:

- $B \cup E \subseteq \mathcal{C}$;
- $\forall (A, x) \in B \cup E$, ${}^\bullet(A, x) = A$.

The initial marking of a $\mathcal{C}$-Petri net is a subset of nodes $\{(\varnothing, x) \mid (\varnothing, x) \in B\}$. For each $\mathcal{C}$-Petri net $CN$, the morphism $h$ maps the nodes of $CN$ to the nodes of $N$: $h((A, x)) = x$. If $h(y) = z$, we say that $y$ is labeled by $z$.

Let $S$ be a (finite or infinite) set of $\mathcal{C}$-Petri nets. The union of $S$ is defined component-wise, i.e.,

$$\bigcup S = (\bigcup_{(P,T,F,M)\in S} P, \bigcup_{(P,T,F,M)\in S} T, \bigcup_{(P,T,F,M)\in S} F, \bigcup_{(P,T,F,M)\in S} M).$$

The set of *branching processes* of a marked P/T-net $N = (P, T, F, M_0)$ is defined as the smallest set of $\mathcal{C}$-Petri nets satisfying the following conditions:

1. The $\mathcal{C}$-Petri net $(I, \varnothing, \varnothing)$, where $I = \{(\varnothing, p) \mid p \in M_0\}$ (consisting of conditions $I$ and having no events), is a branching process.
2. Let $\mathcal{B}_1$ be a branching process and $M$ be a reachable marking of $\mathcal{B}_1$, and $M' \subseteq M$, such that $h(M') = {}^\bullet t$ for some $t$ in $T$. Let $\mathcal{B}_2$ be a net obtained by adding an event $(M', t)$ and conditions $\{(\{(M', t)\}, p) \mid p \in t^\bullet\}$ to $\mathcal{B}_1$. Then $\mathcal{B}_2$ is a branching process.
3. Let $\mathcal{BB}$ be a (finite, or infinite) set of branching processes. The union $\bigcup \mathcal{BB}$ is a branching process.

An example of a P/T-net and its branching process is shown in Figs. 1 and 2. The P/T-net $PN_1$ has the initial marking $\{p_1\}$ and is shown in Fig. 1. One of its possible branching processes is shown in Fig. 2, where the labeling function $h$ is indicated by labels on nodes.

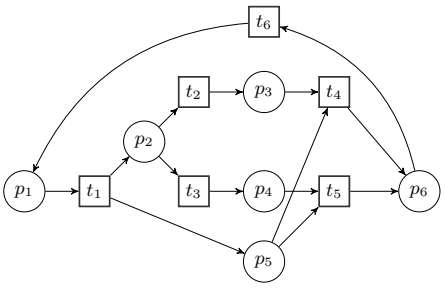


**Fig. 1.** Petri net $PN_1$

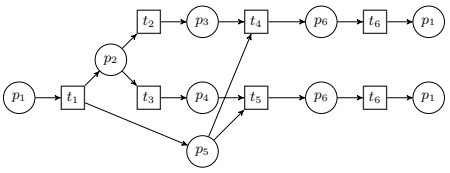


**Fig. 2.** Branching process of $PN_1$

A branching process $\mathcal{B}_1 = ((P_1, E_1, F_1), h_1)$ is called a *prefix* of a branching process $\mathcal{B}_2 = ((P_2, E_2, F_2), h_2)$ (denoted $\mathcal{B}_1 \sqsubseteq \mathcal{B}_2$) iff $P_1 \subseteq P_2$ and $E_1 \subseteq E_2$.

The union of branching processes is called the *unfolding* of $N$. It is easy to see, that the unfolding is the maximal branching process w.r.t the prefix relation $\sqsubseteq$.

The *fundamental property of P/T-nets unfoldings* [28] states that the behavior of the unfolding is equivalent to the behavior of the original net. Formally it can be formulated as follows.

*Fundamental property of P/T-nets unfoldings.* Let $M$ be a reachable marking in a P/T- net $N$, and let $M_U$ be a reachable marking in $U(N)$ s.t. $h(M_U) = M$. Then

1. if there is a step $M_U \xrightarrow{t_U} M'_U$ of $U(N)$, then there is a step $M \xrightarrow{t} M'$ of $N$, such that $h(t_U) = t \wedge h(M'_U) = M'$;
2. if there is a step $M \xrightarrow{t} M'$ of $N$, then there is a step $M_U \xrightarrow{t_U} M'_U$ in $U(N)$, such that $h(t_U) = t \wedge h(M'_U) = M'$.

In other words, the fundamental property of unfoldings states that the reachability graph of the unfolding is isomorphic to the reachability graph of the P/T-net. This property is crucial for the use of unfoldings in semantic study and verification.

Unfoldings were defined and studied for different classes of Petri nets, namely for high-level Petri nets [23], contextual nets [29], time Petri nets [30], Hypernets [31] (to name a few). All these constructions has similar properties, which act as a "sanity check". Further in the paper we define an unfolding operation for nested Petri nets, which posses a similar fundamental property.

### 2.3 Nested Petri Nets

In this paper we deal with nested Petri nets (NP-nets) — in particular, a proper subclass of NP-nets called *strictly conservative* NP-nets. The basic definition of nested Petri nets can be found in [1,8]. Here we give a reduced definition, sufficient for defining conservative NP-nets.

In *nested Petri nets (NP-nets)*, tokens may be Petri nets themselves. An NP-net consists of a *system net* and *element nets*. We call these nets the NP-net *components*. Marked element nets are *net tokens*. Net tokens, as well as usual black dot tokens, may reside in places of the system net. Some transitions in NP-net components may be labeled with *synchronization labels*. Unlabeled transitions in NP-net components may fire autonomously, according to the usual rules for Petri nets. Labeled transitions in the system net should synchronize with transitions (labeled by the same label) in net tokens involved in this transition firing.

In strictly conservative NP-nets, net tokens cannot evolve or disappear. They can "move" from one place in a system net to another and "change" their marking, i.e., inner state. In the basic NP-net formalism new net tokens may be created, copied and removed as usual Petri net tokens. It should be noted that although this restriction is rather strong, many interesting multi-agent systems can be modeled with conservative NP-nets.

Here we consider *safe* and *typed* NP-nets, i.e., each place in a system net can contain no more than one token: either a black dot token, or a net token of a specific type.

Figure 3 provides an example of a nested Petri net $NP_1$. On the left one can see a system net. The token residing in the place *Res* is a net token. Its structure and initial marking is shown on the right side of the figure. The net token

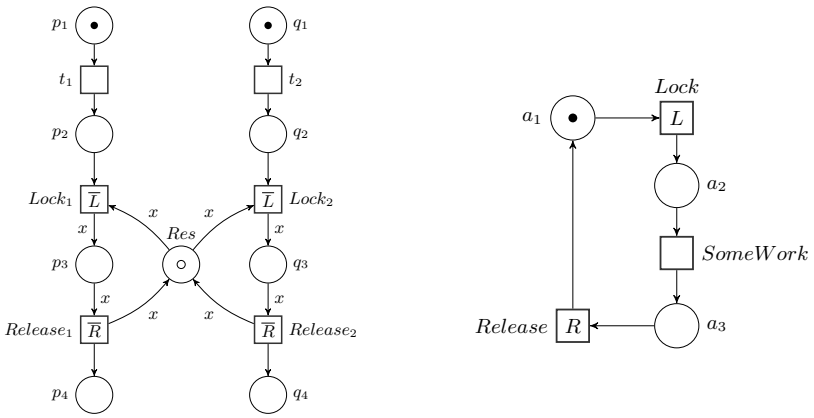


**Fig. 3.** NP-net $NP_1$

represents some sort of resource (for example, a networking or a computational one), capable of performing some internal work (actions). Two threads are trying to access the same resource, but the locking mechanism is preventing them from accessing it simultaneously. The system net synchronizes with the element nets via transitions $Lock_1$, $Lock_2$ and $Release_1$,$Release_2$.

**Definition 1 (Nested Petri nets).** *Let Type be a set of types, Var — a set of typed (over Type)* variables*, and Lab — a set of labels. A (typed)* nested Petri net (NP-net) *NP is a tuple* $(SN, (E_1, \ldots, E_k), \upsilon, \lambda, W)$*, where*

- $SN = (P_{SN}, T_{SN}, F_{SN})$ *is a P/T net called a* system net*;*
- *for* $i = \overline{1,k}$, $E_i = (P_{E_i}, T_{E_i}, F_{E_i})$ *is a P/T net called an* element net*, where all sets of places and transitions in the system and element nets are pairwise disjoint; we suppose, each element net is assigned a type from Type; without loss of generality we shall assume, that* $Type = \{E_1, \ldots, E_k\}$*;*
- $\upsilon : P_{SN} \to Type \cup \{\bullet\}$ *is a* place-typing function*;*
- $\lambda : T_{NP} \to Lab$ *is a partial transition labeling function, where* $T_{NP} = T_{SN} \cup T_{E_1} \cup \cdots \cup T_{E_k}$*; we write that* $\lambda(t) = \bot$ *when* $\lambda$ *is undefined at* $t$*.*
- $W : F_{SN} \to Var \cup \{\bullet\}$ *is an* arc labeling function *s.t. for an arc* $r$ *adjacent to a place* $p$ *the type of* $W(r)$ *coincides with the type of* $p$*.*

*A marked element net is called a* net token*.*

In what follows for a given NP-net by $A_{net} = \{(EN, m) \mid \exists i = 1, \ldots, k : EN = E_i, m \in \mathcal{M}(EN)\}$ we denote the set of all (possible) net tokens, and by $A = A_{net} \cup \{\bullet\}$ the set of all net tokens extended with a black dot token.

Now we come to defining NP-net behavior.

A *marking* $M$ in an NP-net $NP$ is a function mapping each $p \in P_{SN}$ to some (possibly empty) multiset $M(p)$ over $A$ in accordance with the type of $p$. Thus a marking in an NP-net is defined as a marking of its system net. By abuse of

notation, a set of all markings of an NP-net $NP$ will be denoted by $\mathcal{M}(NP)$. We say that a net token $(EN, m)$ resides in $p$ (under marking $M$), if $M(p) \in (EN, m)$.

Let $t$ be a transition in $SN$, ${}^\bullet t = \{p_1, \ldots, p_i\}$, $t^\bullet = \{q_1, \ldots, q_j\}$ be sets of its pre- and post-elements. Then $W(t) = \{W(p_1, t), \ldots, W(p_i, t), W(t, q_1), \ldots, W(t, q_j)\}$ will denote a set of all variables in arc labels adjacent to $t$. A *binding* of $t$ is a function $b$ assigning a value $b(v)$ (of the corresponding type) from $A$ to each variable $v$ occurring in $W(t)$.

A transition $t$ in $SN$ is *enabled* in a marking $M$ w.r.t. a binding $b$ iff $\forall p \in {}^\bullet t : W(p,t)(b) \subseteq M(p)$, i. e. each input place $p$ adjacent to $t$ contains a value of input arc label $W(p,t)$.

The enabled transition *fires* yielding a new marking $M'$, write $M \to M'$, such that for all places $p$, $M'(p) = (M(p) \backslash W(p,t)(b)) \cup W(t,p)(b)$.

For net tokens from $A_{net}$, which serve as values for input arc variables 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 $t$).

There are three kinds of steps in an NP-net $NP$.

*An element-autonomous step.* Let $t$ be a transition without synchronization labels in a net token. Then an autonomous step is a firing of $t$ according to the usual rules for P/T-nets. An autonomous step in a net token does not change the residence of this net token.

A *system-autonomous step* is the firing of an unlabeled transition $t \in T_{SN}$ in the system net according to the firing rule for high-level Petri nets (e.g., colored Petri nets [32]), as described above.

*A synchronization step.* Let $t$ be a transition labeled $\lambda$ in the system net $SN$, let $t$ be enabled in a marking $M$ w.r.t. a binding $b$ and let $\alpha_1, \ldots, \alpha_n \in \mathrm{A}_{net}$ be net tokens involved in this firing of $t$. Then $t$ can fire provided that in each $\alpha_i$ $(1 \leqslant i \leqslant n)$ a transition labeled by the same synchronization label $\lambda$ is also enabled. The synchronization step goes then in two stages: first, firing of transitions in all net tokens involved in the firing of $t$ and then, firing of $t$ in the system net w.r.t. binding $b$.

An NP-net $NP$ is called *safe* iff in every reachable marking in $NP$ there are not more than one token in each place in the system net, and not more that one token in each net token place. Hereinafter we consider only safe NP-nets.

### 2.4 Conservative NP-nets

Now we give a definition of *(strictly) conservative NP-nets*, as well as some related definitions. We then define an unfolding operation for a simple class of strictly conservative nets.

**Definition 2.** *A safe NP-net* $N = (SN, (E_1, \ldots, E_k), \upsilon, \lambda, W)$ *is called* strictly conservative *iff*

1. *For each* $t \in T_{SN}$ *and for each* $p \in {}^\bullet t$, $\exists! p' \in t^\bullet . W(p,t) = W(t,p')$ *or* $W(p,t) = \bullet$
2. *For each* $t \in T_{SN}$ *and for each* $p \in t^\bullet$, $\exists! p' \in {}^\bullet t . W(p',t) = W(t,p)$ *or* $W(p,t) = \bullet$

The definition of strict conservativeness ensures that no net token emerges or disappears after a transition firing in the system net.

Note that in [33] NP-nets are called conservative, iff tokens cannot disappear after a transition firing, but can be copied; hence, the number of net tokens in such conservative NP-nets can be unlimited. Here we consider a more restrictive subclass of NP-nets with a stable set of net tokens (tokens cannot be copied). Hereinafter we consider only strictly conservative NP-nets, and call them just conservative nets for short.

In conservative nets, instead of considering net tokens (marked element nets residing in places of the system net), we consider *identified* net tokens: triples $\langle id, EN, m\rangle$, where $id$ is a unique identifier of the token, $EN$ is a structure of the token (i.e., an element net from the set $\{E_1, \dots E_k\}$), and $m$ is a marking in $EN$. Then every net token in the system net has a unique identifier attached to it; thus, tokens with the same marking can be distinguished.

Further we use $NTok$ to denote a set of identified net tokens for a given net. Sometimes, by abuse of notation, for a net token $\eta = \langle id, EN, m\rangle$ in a place $x$ of a marking $M$, we write $M(x) = \eta$ meaning $M(x) = \{(EN, m)\}$. By $\tau(\eta)$ we denote a type of a net token $(EN, m)$, and by $P_\eta$ $(T_\eta)$ we denote the set of places (transitions) of the net token, i.e., $P_{EN}$ $(T_{EN})$. In the rest of the paper we will use the term *net token* to mean *identified net token*.

Given a system net $SN$, a set of net tokens $NTok$, and a function $\mathfrak{M}$ mapping places of $SN$ to identifiers of $NTok$, it is easy to restore the set of element nets (which is just a set of types from $NTok$), and a marking $M$ (which can be easily restored from $\mathfrak{M}$). Thus, we speak about net tokens in a marking as separate entities, and, in order to define an NP-net, we sometimes list identified net tokens.

For a marking $M$ in an NP-net $NP$ we define *marking projections* onto the components of $NP$:

1. The projection of $M$ onto a system net $SN$, denoted as $M_{\uparrow SN}$, is a marking of the flat P/T-net $SN$ obtained by replacing all the net tokens in $M$ by black dot tokens, i.e., $M_{\uparrow SN}(p) = |M(p)|$.
2. The projection of $M$ onto a net token $\eta = \langle id, EN, m\rangle$, denoted as $M_{\uparrow \eta}$, is just $m$.

## 3 Translation of Safe Conservative NP-nets into P/T-nets

As reachability graph of the unfolding is isomorphic to the reachability graph of the P/T-net, unfoldings can be used in verification. Since safe conservative nested Petri nets have finite number of states, it will be apparent to assume, that they can be translated into classical Petri nets and then can be unfolded according to the classical unfolding rules for further verification.

To make a correct translation we have to set a number of requirements for a translation. The main goal for building a model is the possibility to make a simulation. Simulation implies behavioral equivalence: a possibility to repeat

all possible moves of one model on another model. Behavioral equivalence is guaranteed by establishing strong bisimulation equivalence between states of two models. The second requirement is about constructing a reachability graph. It means that we need exact correspondence between nodes (states) of our model. If these two requirements are met, we can build a translation algorithm which allows us to use target model having the same behavioral properties like original for verification and analysis.

Now we present an algorithm for translating a conservative safe nested Petri net into a safe P/T net.

The algorithm will be illustrated by an example of a NP-net $NP_2$, shown in Fig. 4. Here the net on the left is a system net, and the nets on the right are net tokens residing in the places $p_1$ and $p_2$ of the system net. This net will be translated into a safe P/T net *PN*.

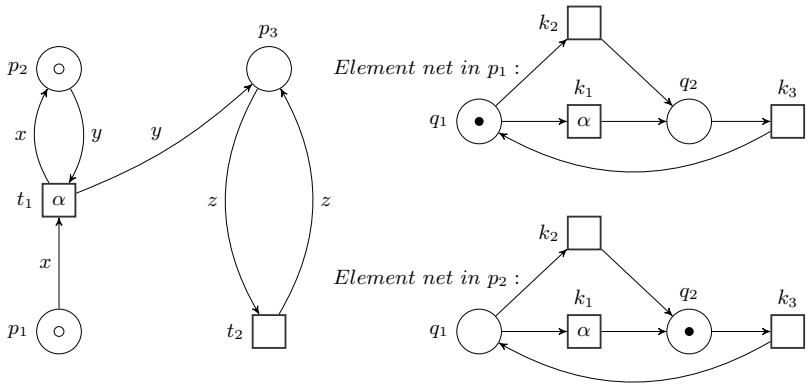


**Fig. 4.** NP-net $NP_2$

**The translation algorithm:** Let $NP = (SN, (E_1, \ldots, E_k), \upsilon, \lambda, W)$ be an NP-net with a set *NTok* of identified net tokens in the initial marking. By $I$ we denote the set of all identifiers used in *NTok*, and by $I_E \subseteq I$ the subset of all identifiers for net tokens of type $E$. The net *NP* will be translated into a P/T net $PN = (P_{PN}, T_PN, F_{PN})$ with an initial marking $m_0$.

1. First, we define the set $P_{PN}$ of places of the target net *PN*. For each type $E$ of some place in the system net *SN* we create a set $\mathfrak{S}_E$ of places for $P_{PN}$. The set $\mathfrak{S}_E$ will contain a copy of each place of type $E$ in the system net for each net token of type $E$ (labeled by net token identifiers) and a copy of each place in $P_E$ for each net token of type $E$, i.e. $\mathfrak{S}_E = \{(p, id) | p \in P_{SN}, \upsilon(p) = E, id \in I_E\} \cup \{(q, id) | q \in P_E, id \in I_E\}$. For a place $p$ in *SN* with black token

type we create just one copy of $p$ without any identifier. Then the set $P_{PN}$ of places for the target net $PN$ is created as the union of all these sets.

2. To define the initial marking for $PN$ we define an encoding of markings on places from $P_{NP}$ in a NP-net by markings on constructed places from $P_{PN}$. If a net token $\eta = (id, E, m)$ resides in a place $p$ in a marking $M$ of the system net, then in the target net there are black tokens in the place $(p, id)$, and all places $(q, id)$ for all $q$ s.t. $m(q) = 1$. If a place of black token type in $SN$ has a black token, then the only corresponding place in $PN$ is also marked by a black token. It is easy to see that this encoding defines a one-to-one correspondence between markings in a safe conservative NP-net and safe markings in $PN$.
   In our example the first element net resides in a place $p_1$, second - in $p_2$. Thus, correspondingly, we define marking in a places $(p_1, 1)$ and $(p_2, 2)$. The same way marking for places $(q_1, 1)$ and $(q_1, 2)$ is defined.
3. For each autonomous transition $t$ in a system net $SN$ we build a set $\mathcal{T}_t$ of transitions as follows. Each input arc variable of $t$ may be, generally speaking, be binded to any of identified net token of the corresponding type. So, for each such binding we construct a separate transition for $PN$ with appropriate input and output arcs.
   Thus for the transition $t_2$ we construct two transitions: $t_{2_1}$ and $t_{2_2}$. It is shown in Fig. 5.

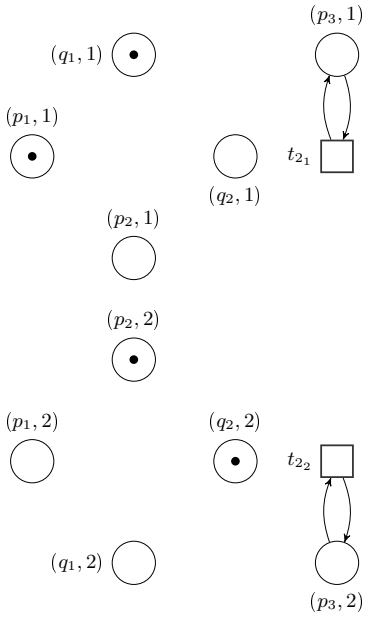


**Fig. 5.** System-autonomous step

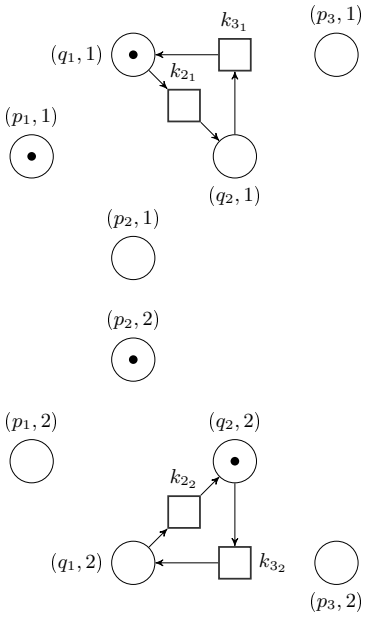


**Fig. 6.** Element-autonomous step

4. For each autonomous transition in a net token from *NTok* identifies=$d$ with $id$ we construct a similar transition on places labeled with $id$. Thus in our example net we obtain four transitions: $k_{2_1}$, $k_{2_2}$, $k_{3_1}$ and $k_{3_2}$. Element-autonomous step is illustrated in Fig. 6.
5. A firing of a synchronization transition supposes simultaneous firing of a transition, which belongs to a system net, and firing of some transition, which has the same label in each involved net token. So synchronization step is a combination of Step 3 and Step 4. Thus as in our example there are two element nets, we add transitions for each net, marked with $\alpha_1$ and $\alpha_2$. Suchwise we can model a synchronization step for every possible initial marking in a system net, which is shown in Fig. 7.

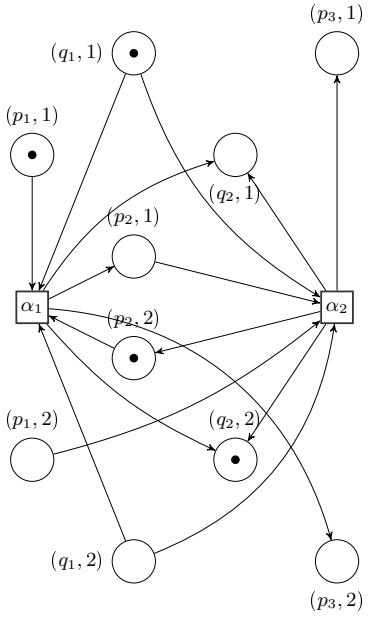


**Fig. 7.** Synchronization step

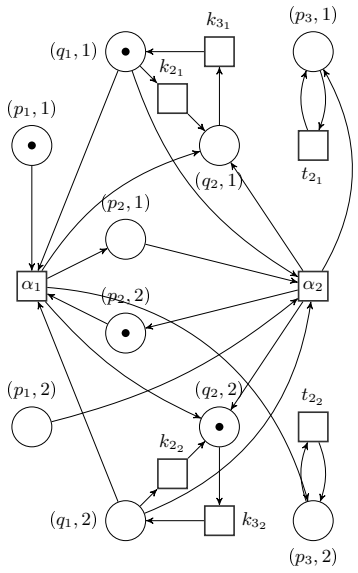


**Fig. 8.** The result of translating $NP_2$ into a P/T net

**Theorem 1.** *Let NP be a NP-net. Let also PN be a P/T net, obtained from NP by the translation, described above. Then reachability graphs of NP and PN are isomorphic.*

*Proof.* Step 2 of the algorithm defines a one-to-one correspondence between reachable markings of nets *NP* and *PN*. It is easy to see that according to translation definition corresponding firing steps in both nets do not violate this correspondence.

Thus we have proven that every safe conservative NP-net can be translated to behaviorally equivalent safe P/T net. Then the standard algorithm for safe P/T nets unfolding can be applied for NP-net verification. The problem here is that the size of the resulting P/T net can grows vastly. In the next section we describe the algorithm for direct unfolding of safe conservative NP-nets, presented previously in [5], and then we compare these two approaches.

## 4 Unfoldings

### 4.1 Branching Processes of a Conservative NP-net

In this section, we define unfoldings of conservative NP-nets into occurrence nets. We give an inductive definition of a branching process of an NP-net, and (similarly to [28]) define the unfolding as the maximal branching process.

First we introduce a concept of an *element-indexed $\mathcal{C}$-Petri net*, a construction similar to the construction of the canonical net for a P/T-net; however, each place of the element-indexed $\mathcal{C}$-net is paired with a net token (identifier). In this section we suppose that $NP = (SN, (E_1, \ldots, E_k), \upsilon, \lambda, M^0)$ is a conservative NP-net, where $SN = (P_{SN}, T_{SN}, F_{SN})$, $E_i = (P_{E_i}, T_{E_i}, F_{E_i})$, $0 < i \leqslant k$.

**Definition 3 (Element-indexed $\mathcal{C}$-Petri nets).** *An* element-indexed $\mathcal{C}$-net $\Theta$ *for some* element-indexing set $J$ *is a $\mathcal{C}$-net such that each place in $\Theta$ is marked with an element of $J$. For our purposes, the set $J$ will be the set of the identified net tokens.*

*Formally, for a fixed net NP, a set of canonical names $C$ is defined as follows:*

- *If $x \in (\bigcup_{E_i} P_{E_i} \cup P_{SN})$, $\eta_i \in NTok \cup \{\bullet\}$, and $X$ is a finite subset of $C$, then $(X, x, \eta_i) \in C$;*
- *If $x \subset (\bigcup_{E_i} T_{E_i} \cup T_{SN})$, and $X$ is a finite subset of $C$, then $(X, x) \in C$.*

*Then an indexed $\mathcal{C}$-net $(P, T, F, M_0)$ is a P/T-net, such that*

1. *$P \cup T \subseteq C$;*
2. *If $p = (X, x, \eta) \in P$, then ${}^\bullet p = X$;*
3. *If $t = (X, x) \in P$, then ${}^\bullet t = X$;*
4. *$(X, x, \eta) \in M_0$ iff $X = \varnothing$ and $x \in (\bigcup_{E_i} P_{E_i} \cup P_{SN})$.*

Just like for regular $\mathcal{C}$-Petri nets, there exists a function $h$ mapping the nodes of an element-indexed $\mathcal{C}$-Petri net to the nodes of *NP*:

$$h(x) = \begin{cases} t & \text{if } x = (A, t) \\ p & \text{if } x = (A, p, \eta_i) \end{cases} \tag{1}$$

The union of element-indexed $\mathcal{C}$-Petri nets is defined component-wise, exactly as it was done for regular $\mathcal{C}$-Petri nets.

We also define a notion of an *adjacent place*. According to Definition 2, for every pair $(p, t) \in P_{SN} \times T_{SN}$, where $\upsilon(p) \neq \bullet \wedge ((p, t) \in F_{SN} \vee (t, p) \in F_{SN})$,

there exists a unique place $p'$ in a system net such that $W(p,t) = W(t,p')$ or $W(p',t) = W(t,p)$. Such a place $p'$ is said to be *adjacent to $p$ via $t$* (denoted by $\widetilde{\langle p,t\rangle}$). For example, in Fig. 4 the place adjacent to $p_2$ via $t_1$ is $\widetilde{\langle p_2,t_1\rangle} = p_3$.

Now we are ready to define a set of *element-indexed branching processes* (or *branching processes* for short, when there is no ambiguity) for a given conservative NP-net *NP*.

**Definition 4 (Element-indexed branching processes for conservative nested Petri nets).**

*The set of* element-indexed branching processes *for NP is the smallest set of element-indexed $\mathcal{C}$-nets satisfying the following rules:*

1. *Let*

$$C = \{(\varnothing, p, \eta_i) \mid p \in P_{SN}, \eta_i \in M^0(p)\} \cup \{(\varnothing, p, \eta_i) \mid \eta_i \in NTok, p \in M^0_{\restriction \eta_i}\}$$

   *be a set of places. The net $\Theta = (C, \varnothing, \varnothing)$ consisting of conditions $C$ and having no transitions is a branching process. Such branching process is said to be* initial.
2. *Let $\Theta$ be a branching process, and $B$ be a subset of conditions of $\Theta$. If $B$ satisfies the PosEN rule's premise (Fig. 9), then the net obtained by adding an event $e$ and conditions $C$ to $\Theta$ is a branching process.*
3. *Let $\Theta$ be a branching process, and $B$ be a subset of conditions of $\Theta$. If $B$ satisfies the PosSN rule's premise (Fig. 9), then the net obtained by adding an event $e$ and conditions $C$ to $\Theta$ is a branching process.*
4. *Let $\Theta$ be a branching process, and let $B$ and $B_E$ be subsets of conditions of $\Theta$. If $B$ and $B_E$ satisfy the PosSync rule's premise (Fig. 9), then the net obtained by adding an event $e$ and conditions $C$ to $\Theta$ is a branching process. The SyncCond predicate is defined below.*
5. *Let $\mathcal{BS}$ be a (finite or infinite) set of branching processes. The union $\bigcup \mathcal{BS}$ is a branching process.*

*In rules (2)-(4), event $e$ is called a* possible extension *of $\Theta$.*

The *SyncCond* predicate in rule (4) makes sure that all the components involved in the synchronization step, synchronize correctly. The parameter $I$ contains the id's of all the net tokens involved in the step. The set $E$ consists of transitions $t_i$ in each of the net tokens $\eta_i$ ($i \in I$), and every $t_i$ carries the same label as the transition $t$ from the system net. In order for the synchronization step to go through, each of the $t_i$ needs to have its pre-set $\{c_j \mid j \in J_i\}$ active. The places of net tokens corresponding to those in the pre-sets are contained in $B_E$.

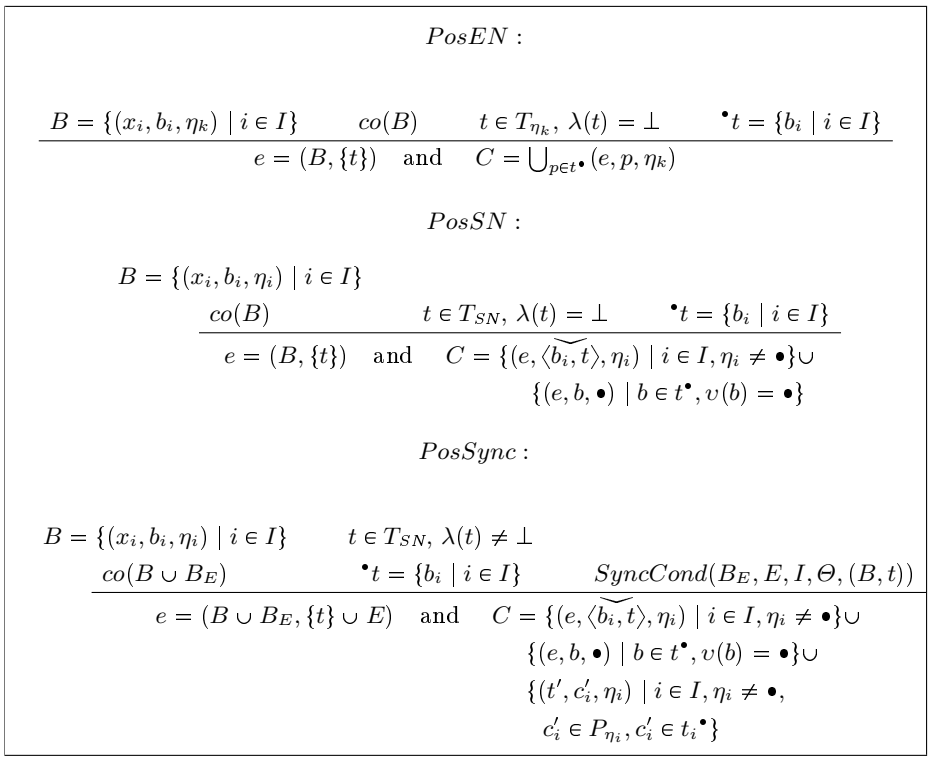


**Fig. 9.** Rules for possible extensions of a branching process

We say that the $SyncCond(B_E, E, I, \Theta, (B, t))$ predicate is true iff the following conditions hold:

1. $B_E = \bigcup_{i \in I} \{(y_j, c_j, \eta_i) \mid j \in J_i, \eta_i = (id_i, EN_i, \mu_i) \in NTok, c_j \in P_{EN_i}\} \wedge co(B_E)$, i.e., $B_E$ is a set of reachable conditions that correspond to places in net tokens;
2. $E = \{t_i \in T_{EN_i} \mid i \in I, \eta_i = (id_i, EN_i, \mu_i) \in NTok\}$, i.e., $E$ is a subset of transitions in each of the net tokens;
3. $\forall t_i \in E, \lambda(t_i) = \lambda(t)$
4. ${}^\bullet E = \bigcup_{i \in I} \{c_j \mid j \in J_i, \eta_i \in NTok\}$

The rules in Fig. 9 can be explained informally from the operational point of view. Rules $PosEN$, $PosSN$, and $PosSync$ are used for generating events that correspond to element-autonomous, system-autonomous, and synchronized firings, respectively.

A possible branching process of $NP_2$ is shown in Fig. 10. In Fig. 10, a transition is labeled with $t$, if it is of the form $(A, t)$, and a place is labeled with $(p, N)$ if it is of the form $(A, p, N)$.

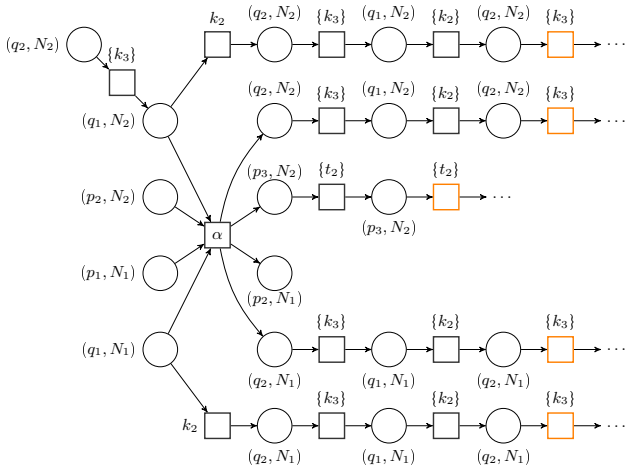


**Fig. 10.** Branching process of $NP_2$

It was proven in [5] that every element-indexed branching process is an occurrence net and that the fundamental property of unfoldings holds for the definition of conservative NP-nets unfolding.

Note also that every low-level P/T-net is a special case of an NP-net with the empty set of element nets and no vertical synchronization. It was shown also in [5] that the branching process definition for NP-nets is in accord with the branching process definition for low-level Petri nets., i.e. for a P/T-net $N$ the set of branching processes of $N$ is isomorphic to the set of element-indexed branching processes of $N$, when $N$ is considered as an NP-net.

### 4.2 Comparing Two Ways of Nested Petri Net Unfolding

We have shown that each conservative safe NP-net can be converted into a behaviorally equivalent classical Petri net, namely their reachability graphs are isomorphic. So, to construct unfoldings for a NP-net we can either translate it into a P/T net and then apply the classical P/T net unfolding procedure, or directly construct NP-nets unfoldings, as it is described in the previous subsection.

The fundamental property of unfoldings states that the reachability graph of the unfolding is isomorphic to the rechability graph of the initial net. Since the fundamental property holds both for P/T net unfoldings and for NP-net unfoldings, we can immediately conclude that both approaches give the same (up to isomorphism) branching process. For our example this is demonstrated by Fig. 10 and Fig. 12.

The difference is in the complexity of these two solutions. It is easy to see, that when there are several net tokens of the same type in the initial marking, the translation leads to a significant net growth. Thus e.g. for a system net

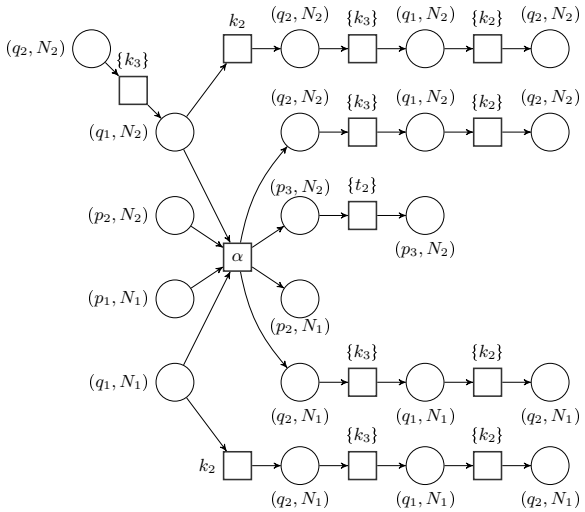


**Fig. 11.** Complete branching process of $NP_2$

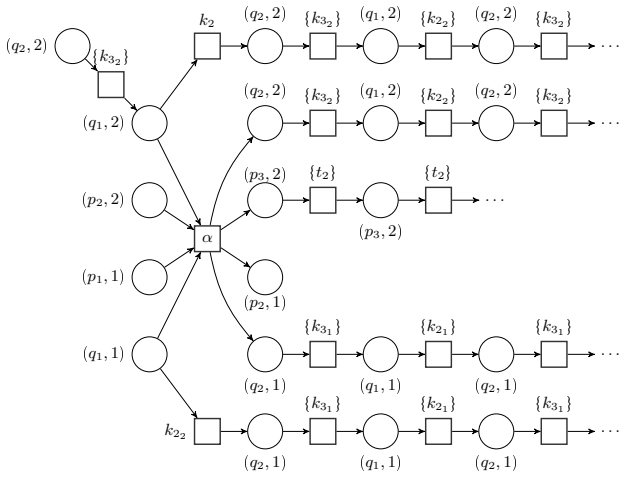


**Fig. 12.** Branching process of $NP_2$ obtained via translation to a P/T net

transition $t$ with $n$ input places of the same type and $k$ tokens of this type in the initial marking we are to construct $k^n$ copies of this transition in the target P/T net, corresponding to different bindings for $t$-firings. And it is rather clear, that we cannot avoid this, since we are to distinguish markings of net tokens residing in different places, and hence to construct a separate P/T net transition for each mode of a system net transition firing.

To check the advantage of the direct unfolding method w.r.t. time complexity for concrete examples we've developed a software application which allows

1. translation of a conservative safe nested Petri net into a P/T net and then building an unfolding for it;
2. building an unfolding directly for a nested Petri net.

We expected that a large number of net tokens will cause significant net growth during translation. The reason for this is that dealing with a system, which consists of a large number of net tokens and incoming arcs, translation of a nested Petri net into a P/T net leads to a significant growth of the net graph. Since we do not know in advance, which modes of transition firing will be used in the unfolding, we should build an intermediate P/T net with a lot of transitions unnecessary for unfolding, while in direct unfoldings these transition nodes do not appear.

So, we conducted experiments on nets having similar structure, but different number of element nets with different types. We've done a series of experiments with rather small models, which confirm our assumptions. Thus for our example net $NP_2$ we've got 0.38 ms. for the direct unfolding, and 0.54 ms. for unfolding via the translation into a P/T net. So, even in the case of two net tokens we get a noticeable difference in time.

To get representative experiment results we are to do more experiments with larger models of different structure.

*Application to verification.* Having the described above algorithm for NP-nets unfoldings the basic algorithm (described in [26]) for constructing finite prefixes of unfoldings of low-level P/T-nets can be modified in a straightforward way to obtain an algorithm for constructing finite prefixes of unfoldings of conservative NP-nets. In fact, the only part of the algorithm that needs to be modified is the PotExt function, which has to be changed in accordance with the possible extension rules in Fig. 9. This is attainable because all the necessary definitions (in particular, the definition of a *cutting context*) and the theory of canonical prefixes [25],[26] can be directly extended to cover NP-nets.

For example, let's consider the result of the standard algorithm applied to the NP-net $NP_2$ from Fig. 4 using the McMillan's cutting context (in the notation of [26], $C' \approx C'' \iff Mark(C') = Mark(C'')$ and $C' \lhd C'' \iff |C'| < |C''|$).

We have shown a canonical prefix for $NP_2$ in Fig. 11. This canonical prefix $BP_C$ allows us to solve the executability problem: a transition $t$ may fire in the NP-net iff an event labeled with $t$ is presented in the canonical branching process. For example, one can observe, that because a transition $e_1$ in $BP_2$ has is labeled with $\{t_1, k_1\}$, the transition $t_1$ is executable in the NP-net. Also, we can easily see that for both tokens $k_1$ may fire only once, but $k_2$ and $k_3$ are live transitions.

## 5 Conclusion

In this paper we've proposed and compared two ways of unfolding for safe conservative nested Petri nets. The first method is based on equivalent translation of NP-nets into safe P/T nets and then applying standard unfolding procedure

described in the literature. The second method is a direct unfolding, proposed and justified earlier in [5].

For that we've developed and justified an algorithm for translation of a safe conservative NP-net into an equivalent P/T net. Direct analysis of the algorithm complexity allows us to conclude that the direct unfolding has a distinct advantage in time complexity. To check this advantage with practical examples we've implemented the algorithms for translation and unfolding. Experiments on small nets have demonstrated the anticipated benefits of direct unfolding.

For further work, we plan to enlarge the complexity of nets and number of experiments.

## References


1. Lomazova, I.A.: Nested Petri nets—a formalism for specification and verification of multi-agent distributed systems. Fundamenta Informaticae **43**(1) (2000) 195–214
2. Valk, R.: Object petri nets. In: Lectures on Concurrency and Petri Nets. Springer (2004) 819–848
3. McMillan, K.L.: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: Computer Aided Verification, Springer (1992) 164–177
4. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains, part i. Theoretical Computer Science **13**(1) (1981) 85–108
5. Frumin, D., Lomazova, I.A.: Branching processes of conservative nested Petri nets. In: VPT@ CAV. (2014) 19–35
6. Lomazova, I.A., van Hee, K.M., Oanea, O., Serebrenik, A., Sidorova, N., Voorhoeve, M.: Nested nets for adaptive systems. Application and Theory of Petri Nets and Other Models of Concurrency, LNCS (2006) 241–260
7. Lomazova, I.A.: Modeling dynamic objects in distributed systems with nested petri nets. Fundamenta Informaticae **51**(1-2) (2002) 121–133
8. Lomazova, I.A.: Nested petri nets for adaptive process modeling. In: Pillars of computer science. Springer (2008) 460–474
9. López-Mellado, E., Villanueva-Paredes, N., Almeyda-Canepa, H.: Modelling of batch production systems using Petri nets with dynamic tokens. Mathematics and Computers in Simulation **67**(6) (2005) 541–558
10. Kahloul, L., Djouani, K., Chaoui, A.: Formal study of reconfigurable manufacturing systems: A high level Petri nets based approach. In: Industrial Applications of Holonic and Multi-Agent Systems. Springer (2013) 106–117
11. Zhang, L., Rodrigues, B.: Nested coloured timed Petri nets for production configuration of product families. International journal of production research **48**(6) (2010) 1805–1833
12. Venero, M.L.F., da Silva, F.S.C.: Modeling and simulating interaction protocols using nested Petri nets. In: Software Engineering and Formal Methods. Springer (2013) 135–150
13. Chang, L., He, X., Lian, J., Shatz, S.: Applying a nested Petri net modeling paradigm to coordination of sensor networks with mobile agents. In: Proc. of Workshop on Petri Nets and Distributed Systems, Xian, China. (2008) 132–145
14. Cristini, F., Tessier, C.: Nets-within-nets to model innovative space system architectures. In: Application and Theory of Petri Nets. Springer (2012) 348–367

15. Mascheroni, M., Farina, F.: Nets-within-nets paradigm and grid computing. In: Transactions on Petri Nets and Other Models of Concurrency V. Springer (2012) 201–220
16. Dworzański, L.W., Lomazova, I.A.: On compositionality of boundedness and liveness for nested Petri nets. Fundamenta Informaticae **120**(3-4) (2012) 275–293
17. Dworzański, L., Lomazova, I.A.: Cpn tools-assisted simulation and verification of nested petri nets. Automatic Control and Computer Sciences **47**(7) (2013) 393–402
18. Venero, M.L.F.: Verifying cross-organizational workflows over multi-agent based environments. In: Enterprise and Organizational Modeling and Simulation. Springer (2014) 38–58
19. Winskel, G.: Event structures. Springer (1986)
20. Bonet, B., Haslum, P., Hickmott, S., Thiébaux, S.: Directed unfolding of petri nets. In: Transactions on Petri Nets and Other Models of Concurrency I. Springer (2008) 172–198
21. McMillan, K.L.: A technique of state space search based on unfolding. Form. Methods Syst. Des. **6**(1) (1995) 45–65
22. Heljanko, K.: Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe petri nets. Fundamenta Informaticae **37**(3) (1999) 247–268
23. Khomenko, V., Koutny, M.: Branching processes of high-level Petri nets. In Garavel, H., Hatcliff, J., eds.: Tools and Algorithms for the Construction and Analysis of Systems. Volume 2619 of Lecture Notes in Computer Science. Springer (2003) 458–472
24. Langerak, R., Brinksma, E.: A complete finite prefix for process algebra. In: Computer Aided Verification, Springer (1999) 184–195
25. Khomenko, V., Koutny, M., Vogler, W.: Canonical prefixes of Petri net unfoldings. Acta Informatica **40**(2) (2003) 95–118
26. Khomenko, V.: Model Checking Based on Prefixes of Petri Net Unfoldings. Ph.D. Thesis, School of Computing Science, Newcastle University (2003)
27. Esparza, J., Heljanko, K.: Unfoldings: a partial-order approach to model checking. Springer (2008)
28. Engelfriet, J.: Branching processes of Petri nets. Acta Informatica **28**(6) (1991) 575–591
29. Baldan, P., Corradini, A., König, B., Schwoon, S.: Mcmillan's complete prefix for contextual nets. In Jensen, K., Aalst, W.M., Billington, J., eds.: Transactions on Petri Nets and Other Models of Concurrency I. Volume 5100 of Lecture Notes in Computer Science. Springer (2008) 199–220
30. Fleischhack, H., Stehno, C.: Computing a finite prefix of a time Petri net. In Esparza, J., Lakos, C., eds.: Application and Theory of Petri Nets 2002. Volume 2360 of Lecture Notes in Computer Science. Springer (2002) 163–181
31. Mascheroni, M.: Hypernets: a Class of Hierarchical Petri Nets. Ph.D. Thesis, Facolt di Scienze Naturali Fisiche e Naturali, Dipartimento di Informatica Sistemistica e Comunicazione, Università Degli Studi Di Milano Bicocca (2010)
32. Jensen, K., Kristensen, L.M.: Coloured Petri nets: modelling and validation of concurrent systems. Springer (2009)
33. Dworzański, L.W., Lomazova, I.A.: On compositionality of boundedness and liveness for nested Petri nets. Fundamenta Informaticae **120**(3) (2012) 275–293