Nested Petri Nets: Multi-level and Recursive Systems.

Article *in* Fundamenta Informaticae · August 2001

Source: DBLP

CITATIONS
62

READS
533

1 author:

Irina A. Lomazova
National Research University Higher School of Economics
114 PUBLICATIONS **850** CITATIONS

SEE PROFILE

Nested Petri Nets: Multi-level and Recursive Systems

Irina A.Lomazova*

Artificial Intelligence Center

Program Systems Institute of Russian Academy of Science

Pereslavl-Zalessky, 152140, Russia

irina@univ.botik.ru

Abstract. Nested Petri nets (NP-nets) is a formalism for modeling hierarchical multi-agent systems. Tokens in nested Petri nets are elements represented by nets themselves. The paper continues investigating semantic properties of NP-nets, started in [10], where two-level NP-nets were studied. Here we consider multi-level and recursive NP-nets, for which decidability of termination and some other properties are proved. A comparison of NP-nets with other Petri net models is given.

Keywords: hierarchical multi-agent systems, nested Petri nets, well-structured transition systems, decidability.

1. Introduction

Extending Petri nets by notions and structures of object programming draws attention of many researchers (see e.g. an overview [18]). In this paper we study Nested Petri nets (NP-nets) — a Petri net model with tokens allowed to be nets themselves. The idea of supplying tokens by their own net structure and behavior is due to R.Valk [16]. In Object Petri nets (OPNs) of Valk actually tokens are not nets, but references to some marked nets, and two distinct tokens may refer to the same object net. So, in OPNs an object net token in a marking may be in some sense distributed over a system net. This leads to a rather complicated firing rule [17]. R.Valk considers nets, where a system net and its tokens (object nets) are elementary net systems.

To use the analysis power of existing Petri net analysis tools Chameleon systems [13, 14] restrict OPNs of Valk to such nets that can be unfolded to classical Place/Transition nets. In Chameleon Systems

*

a system net is a simple state machine, which can move token nets from one place to another. The more general case, when copying, merging or vanishing of tokens is allowed, leads to problems on the semantical level [14]. Due to their more simple net structure Chameleon Systems allow the intrinsic extension to the multi-level case.

Another model based on OPNs of Valk is B. Farwer's Linear Logic Petri nets (LLPNs) [4, 5], where tokens are represented by Linear Logic formulae. A LLPN is representable as a finite Linear Logic formula, and thus can be used as a token in another LLPN. This gives a possibility to represent multi-level systems by LLPN formalism.

In contrast to OPNs of Valk in NP-nets a net token (in a fixed marking) is located in one place. Element nets in a NP-net may be copyed, evolve and disappear during a system run, and their number, as well as number of levels in them is unlimited. Thus NP-nets allow multi-level and recursive cases. Though the paper is self-contained, it may be considered as a continuation of [10], where two-level NP-nets were formally defined and some decidability/undecidability results for them were studied. Here we give a general formal definition of NP-nets. Two-level NP-nets from [10] form a special case with net tokens in a system net being ordinary Petri nets.

A behavior of a NP-net includes three kinds of steps. An autonomous step is a step in a net in some level, which may "move", "copy", "generate", or "remove" its elements (tokens), but doesn't change their inner markings. Thus, in autonomous steps an inner structure of tokens is not taken into account. In [10] we distinguished autonomous steps in system nets and in element nets and called them respectively a transfer step and an element-autonomous step. In a multi-level case the same net may simultaneously be a system net for its token and an element net for its parent net, so we call them just autonomous steps. There are also two kinds of synchronization steps. Horizontal synchronization means simultaneous firing of two element nets, located in the same place of a system net. Vertical synchronization means simultaneous firing of a system net together with its elements "involved" in this firing.

The paper is organized as follows. Section 2 contains formal definitions of NP-nets. In section 3 we prove that multi-level NP-nets form well-structured transition systems w.r.t. a special ordering on sets of reachable markings. This implies decidability of termination and some other problems. Section 4 contains decidability results for recursive NP-nets. In section 5 we compare NP-nets with other Petri net models and give a summary of decidability/undecidability results for them.

2. Multi-level and recursive nested Petri nets: definitions

In this section we give a general definition of multi-level NP-nets and NP-nets with recursive structure. Some examples of a two-level NP-net and a recursive NP-net may be found in [10]. We suppose the reader is familiar with some basic notions of Petri net theory.

Let *Var* be a set of *variable* names, and *Con* — a set of *constant* names. Further in the definition of NP-nets constants from Con are interpreted either as marked ordinary Petri nets (net tokens), or as individual tokens without inner structure (atom tokens). By A_{net} we denote the set of net tokens, by A_{atom} — the set of atom tokens. We suppose, that $\mathrm{A}_{atom} \neq \emptyset$ and fix a distinguished token, which corresponds to 1 in the language *Expr* and is usually depicted as a black dot token in net pictures.

We define a language *Expr* of expressions over $Var \cup Con$ with operations "$(,\dots,)_n$" and "$+$". An expression $(a_1, \dots, a_n)_n$ will be interpreted as a tuple of elements, "$+$" operation may be applied only to tuples of the same dimension and is interpreted as a multiset union. Thus, expressions from *Expr* have

multisets of n-tuples of tokens as their values. For $e \in \mathit{Expr}$, by $\mathit{Var}(e)$ we denote the set of variables occurring in e.

Let $\mathit{Lab} =_{\text{def}} \mathit{Lab}_v \cup \mathit{Lab}_h$ be two disjoint sets of labels for vertical and horizontal synchronization respectively. For $l \in \mathit{Lab}$ we define an *adjacent* label $\overline{l}$, such that for $l_1, l_2 \in \mathit{Lab}$,$l_1 \neq l_2$ implies $\overline{l_1} \neq \overline{l_2}$; and $\overline{\overline{l}} =_{\text{def}} l$.

Definition 2.1. A net component is a high-level Petri net $\mathfrak{N} = (N, \mathcal{L}, W, \Lambda)$, where

1. $N = (P, T, F)$ is a net, i.e. a bichromatic digraph with the set $P \cup T$ of nodes and the set F of arcs. Elements of P are called places, each place is ascribed a natural non-zero number (its arity). Elements of T are called transitions;

2. $\mathcal{L} = \mathit{Expr}$ — the language of expressions defined above;

3. W — a function mapping an arc $(x, y) \in F$ to an expression $W(x, y)$ with dimension n, where n is the arity of the place incident to an arc (x, y). Moreover, for each transition its arc expressions must satisfy the following *input arc restrictions*:
   – there are no net constants (with values in $\mathrm{A}_{\mathit{net}}$) in input arc expressions;
   – every variable has no more then one occurrence in each input arc expression;
   – for every two expressions $W(p_1, t)$ $W(p_2, t)$ ascribed to two different input arcs of a same transition t, $\mathit{Var}(W(p_1, t)) \cap \mathit{Var}(W(p_2, t)) = \emptyset$.
   There is also an *output arc restriction*: each variable in an output arc expression (w.r.t a transition t) must occur in one of the input arc expressions for t.

   If an arc expression is not given explicitly we suppose it to be $1 \in \mathbb{N}$;

4. Λ — a partial function, labeling transitions from T by labels from $\mathit{Lab}_v \cup \mathit{Lab}_h$.

In particular, when all arc expressions in a net component are natural numbers and the arity of all places equals 1, such a net component represents an ordinary Petri net.

Let $\overline{\mathfrak{N}} = (\mathfrak{N}_0, \ldots, \mathfrak{N}_k)$ be a finite set of net components, A_a — a finite set of atom constants. A *marking* of a net component $\mathfrak{N}_i$ over $\overline{\mathfrak{N}}$ and A_a is defined by induction.

Definition 2.2. 1. A function M, mapping each place p in a net $\mathfrak{N}_i$ $(1 \leq i \leq k)$ to a finite multiset $M(p)$ of tuples of atomic tokens from $a \in \mathrm{A}_a$, is a marking of $\mathfrak{N}_i$ (over $\overline{\mathfrak{N}}$ and A_a). Here the dimension of tuples from $M(p)$ must coincide with the arity of p. We call a pair $(\mathfrak{N}_i, M)$ a *marked net component* or a *net token*.

2. Let A_n be a set of marked net components. Then a function M, mapping each place p (of arity n) in a net $\mathfrak{N}_i$ $(1 \leq i \leq k)$ to a finite multiset $M(p)$ of n-tuples with elements from the set $\mathrm{A}_n \cup \mathrm{A}_a$ of atomic and net tokens, is a marking of $\mathfrak{N}_i$ (over $\overline{\mathfrak{N}}$ and A_a) and a pair $(\mathfrak{N}_i, M)$ forms a *marked net component*.

Denote by $\mathrm{A}_n = \mathrm{A}_n(\overline{\mathfrak{N}}, \mathrm{A}_a)$ the set of net tokens over $\overline{\mathfrak{N}}$ and A_a.

A marking M of a net component $\mathfrak{N}_0$ over $\overline{\mathfrak{N}}$ and A_a may naturally be represented by a finite labeled rooted tree $\mathcal{T}(M)$ (called a *marking tree*), such that

– The root of $\mathcal{T}(M)$ is labeled by $\mathfrak{N}_0$.

– A node $\mathfrak{N}_i$ has k_i sons (where k_i is the number of places in $\mathfrak{N}_i$), labeled by the names $p^i_1, \ldots, p^i_{k_i}$ of places in $\mathfrak{N}_i$.

– Each node, labeled by a place name p^i_j, has sons, labeled by the symbol "$(\,,\ldots,\,)_n$" of the n-tuple operation, where n is the arity of p^i_j. The number of sons of this node is equal to the number of tokens in $M(p^i_j)$ (possibly zero).

– A node, labeled by "$(\,,\ldots,\,)_n$" has n sons, labeled by numbers $1, 2, \ldots, n$ respectively.

– A node, labeled by a natural number (a position index in a tuple), has exactly one son, labeled either by an atomic token name, or by a name of a net component from $\mathfrak{N}_0, \ldots, \mathfrak{N}_k$. Here a node, labeled by an atomic token name is a leaf in $\mathcal{T}(M)$, and a node, labeled by $\mathfrak{N}_j$ is a root of a subtree, which forms a marking tree for a net component $\mathfrak{N}_j$.

– There is a one-to-one correspondence between tuples of tokens in $M(p)$ and subtrees induced by sons of p, such that for a token a in the i-th position in a tuple the son of the corresponding node, labeled by i, is either a leaf (in the case when a is an atomic token), or induces a subtree, which is a marking tree of the net token a.

The *depth* of a marking M is defined as the maximal number of nodes labeled by net token names, which lay on a path from the root to some leaf in $\mathcal{T}(M)$ (i.e. the maximal number of net components nestedness).

A marking M of a net component $\mathfrak{N}$ is called *feasible*, if for each net component name $\mathfrak{N}_i$ on each path from the root of $\mathcal{T}(M)$ to some of its leaves at most one node is labeled by $\mathfrak{N}_i$.

A *constant interpretation* $\mathcal{I}$ (over A_a and $\overline{\mathfrak{N}}$) maps each constant from *Con*$_a$ to an atomic token from A_a, and a constant from *Con*$_n$ — to a net token from $\mathrm{A}_n(\overline{\mathfrak{N}}, \mathrm{A}_a)$.

Definition 2.3. A *nested Petri net (NP-net)* is a tuple *NPN* $= (\overline{\mathfrak{N}}, \mathrm{A}_a, \mathcal{I}, \mathfrak{N}_0, M_0)$, where

– $\overline{\mathfrak{N}} = (\mathfrak{N}_0, \ldots, \mathfrak{N}_k)$ is a finite set of net components, such that all place and transition names in $\overline{\mathfrak{N}}$ are pairwise disjoint;

– A_a — a finite set of atomic tokens;

– $\mathcal{I}$ — an interpretation of constants in $\overline{\mathfrak{N}}$ over A_a and $\mathrm{A}_n(\overline{\mathfrak{N}}, \mathrm{A}_a)$.

– $\mathfrak{N}_0$ — a distinguished net component, called a *system net*;

– M_0 — a feasible marking of a system net $\mathfrak{N}_0$ over $\overline{\mathfrak{N}}$ and A_a, called *initial marking* of a NP-net *NPN*.

Note, that the definition of two-level NP-nets from [10] is a special case, where all net components except the system net are ordinary Petri nets. The other difference is that tuples of tokens instead of just tokens are allowed in NP-nets. There is also the additional restriction on output arc expressions, which is needed for the finiteness of branching in the corresponding transition system. For two-level NP-nets this was not mentioned explicitly and could be ensured by finiteness of the set of individual tokens.

A marking of a NP-net is a marking of its system net. It's easy to see, that the depth of all feasible markings for a NP-net is bounded.

For a NP-net *NPN* we define the *nestedness graph* $G_\lhd$(*NPN*), which shows possible nestedness dependencies between net components of *NPN*.

Definition 2.4. Let *NPN* $= (\overline{\mathfrak{N}}, \mathrm{A}_a, \mathcal{I}, \mathfrak{N}_0, M_0)$ be a NP-net. The nestedness graph $G_\lhd$(*NPN*) is a digraph with net components $\mathfrak{N}_0, \ldots, \mathfrak{N}_k$ as its nodes. An arc goes from a node $\mathfrak{N}_i$ to a node $\mathfrak{N}_j$ in $G_\lhd$(*NPN*) iff one of the following conditions holds:

1. There is a constant $c \in Con_n$ with $\mathcal{I}(c) = (\mathfrak{N}_j, M)$ in $\mathfrak{N}_i$.

2. There is a constant $c \in Con_n$ with $\mathcal{I}(c) = (\mathfrak{N}, M)$, s.t. there is a path in the marking tree $\mathcal{T}(M)$ from a node $\mathfrak{N}_i$ to a node $\mathfrak{N}_j$.

3. There is a path in the initial marking tree $\mathcal{T}(M_0)$ from a node $\mathfrak{N}_i$ to a node $\mathfrak{N}_j$.

We call a NP-net *NPN* a *multi-level NP-net*, if its nestedness graph doesn't contain directed cycles. Otherwise we call it *recursive NP-net*.

Next we discuss NP-net behavior. For a net component which is an ordinary Petri net, a transition firing is defined as usual. Let a net component $\mathfrak{N}$ be a high-level Petri net; t — a transition in $\mathfrak{N}$; ${}^\bullet t$, $t^\bullet$ — sets of its pre- and post-conditions respectively. By $W(t)$ we denote the set of all arc expressions ascribed to arcs incident to t. A *binding* of a transition t is a function b, assigning to each variable v, occurring in $W(t)$, a value $b(v)$ from $\mathrm{A}_a \cup \mathrm{A}_n(\overline{\mathfrak{N}}, \mathrm{A}_a)$. A transition t in *SN* is *enabled* in marking M w.r.t. a binding b iff $\forall p \in {}^\bullet t : W(p,t)(b) \subseteq M(p)$. The enabled transition fires yielding a new marking M', such that for all places p, $M'(p) = (M(p) - W(p,t)(b)) + W(t,p)(b)$. For net tokens from A_{net} that are chosen 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 may be brought to output places of t).

There are three kinds of steps in a NP-net *NPN*. Let M be a marking in *NPN*.

An autonomous step is a firing of some unlabeled transition t in some net token $(\mathfrak{N}, M_{\mathfrak{N}})$ of the marking M (i.e. the value of $\Lambda(t)$ is not defined). After such a firing the marking $M_{\mathfrak{N}}$ may change, while the net token itself remains in the same place in marking M.

It's easy to see, that an autonomous step does not change inner markings of net tokens inside $(\mathfrak{N}, M_{\mathfrak{N}})$ (if there are any), but it can transfer, copy or remove some of them. Also new net tokens may evolve as a result of an autonomous step.

A horizontal synchronization step consists of simultaneous (autonomous) firings of two transitions labeled by mutually adjacent horizontal labels in two net tokens $a_i = (\mathfrak{N}_1, M_1)$, $a_j = (\mathfrak{N}_2, M_2)$. Here net tokens a_i and a_j must be components of the same tuple $\overline{a} = (a_1, \ldots, a_n)$ in some place p of a net component $(\mathfrak{N}, M_{\mathfrak{N}})$ in the marking M.

If the arity of place p is 1 (p contains no tuples, but just tokens), then an adjacent net token (with an enabled transition labeled by the adjacent label) for a net token a_i must be chosen among net tokens located in the same place p.

A vertical synchronization step consists of simultaneous firings of a transition t, labeled by $l \in Lab_v$, in some net token in the marking M and transitions, labeled by $\overline{l}$, in all net tokens involved in this firing. Note, that exactly one transition (enabled and labeled by $\overline{l}$) is chosen in each involved net token, and only net tokens of two adjacent level may synchronize vertically.

For two markings M, M' in a NP-net we write $M[\rangle M'$ if there exists either an autonomous step, or a horizontal synchronization step, or a vertical synchronization step, which transfers M into M'. A marking M is called reachable, if there exists a sequence of steps $M_0[\rangle M_1[\rangle \ldots [\rangle M_k$ with $M = M_k$.

Proposition 2.5. Each reachable marking is feasible in a multi-level NP-net.

Proof: A net component $\mathfrak{N}_i$ in a NP-net *NPN* can generate a net $\mathfrak{N}_j$ as its net token only if there is an arc from $\mathfrak{N}_i$ to $\mathfrak{N}_j$ in the nestedness graph $G_\lhd(NPN)$. Since the nestedness graph of a multi-level NP-net is acyclic, there can't be a path from a node $\mathfrak{N}_i$ to a node with the same label in any reachable marking tree. □

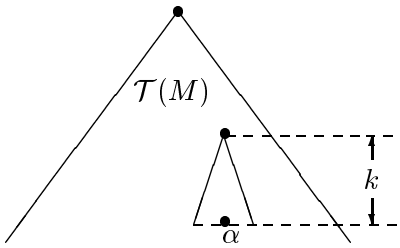

Figure 1. A subtree $\mathcal{T}^k(M,\alpha)$ in a marking tree $\mathcal{T}(M)$

Generally speaking, a behavior (changing of its inner marking) of a net token may depend on markings of its outer nets due to vertical synchronization. In the following definition we restrict behavior dependencies to a finite number of outer levels.

Let α be a net token in a marking M of *NPN*. Denote by $\mathcal{T}^k(M,\alpha)$ the maximal subtree of the height k in a marking tree $\mathcal{T}(M)$, s.t. $\mathcal{T}^k(M,\alpha)$ contains α as a leaf (cf. Fig. 2).

Definition 2.6. Let M be a marking in *NPN*. A net token α in M is called *localized* iff there exists a natural number $k \geq 0$ s.t. the behavior of α depends only on the subtree $\mathcal{T}^k(M,\alpha)$ in the marking tree of M. The number k is called the *localization degree of* α.

A NP-net *NPN* is called a *net with localized elements* iff for some $k \geq 0$ all net tokens in all reachable markings of *NPN* are localized with the degree k.

The special case of localization can be ensured by some restrictions on vertical synchronization steps. We define a *consuming vertical synchronization step*/, for which its firing removes all net tokens involved in it. In this case only generating and removing of a net token may depend on its parent marking. It is easy to see, that vertical synchronization is consuming, if, in particular, output arc expressions of each transition labeled for vertical synchronization does not contain any variables, which occur also in input arc expressions and can be bound by net values. Note, that this property can often be stated on the syntactical level.

Definition 2.7. Call a transition t with $\Lambda(t) \in \mathit{Lab}_v$ *consuming*, if $\mathit{Var}(W(p,t)) \cap \mathit{Var}(W(t,q)) \neq \emptyset$ implies that p does not contain net tokens in all reachable markings. A NP-net is called a *net with autonomous elements*, if all transitions marked for vertical synchronization are consuming.

It is easy to see, that autonomous elements are localized with localization degree zero.

An example of a recursive NP-net can be found in [10]. The net in this example is a net with autonomous elements.

3. Multi-level NP-nets as well-structured transition systems

While adding nested structure to Petri nets we tried to maintain the main features of the Petri net model. Thus some problems crucial for verification remain decidable for nested Petri nets. Decidability of termination and some other properties for non-recursive NP-nets are mostly based on the theory of Well-Structured Transition Systems (WSTS) [7].

Recall that a transition system is a pair $\mathcal{S} = \langle S, \rightarrow \rangle$ where S is an abstract set of states (or markings) and $\rightarrow \subseteq S \times S$ is any transition relation. For a transition system $\mathcal{S} = \langle S, \rightarrow \rangle$ we write *Succ*(s) for the set $\{s' \in S \mid s \rightarrow s'\}$ of immediate successors of s. $\mathcal{S}$ is finitely branching if all *Succ*(s) are finite.

A *quasi-ordering* is any reflexive and transitive relation $\leq$ (over some set X). A *well-quasi-ordering* (a wqo) is any quasi-ordering $\leq$ such that, for any infinite sequence $x_0, x_1, x_2, \ldots$, in X, there exist indices i, j with $i < j$ and $x_i \leq x_j$. Note, that if $\leq$ is a wqo then any infinite sequence contains an infinite increasing subsequence: $x_{i_0} \leq x_{i_1} \leq x_{i_2} \ldots$.

Definition 3.1. A transition system $\Sigma = \langle S, \rightarrow \rangle$ is *well-structured* (a WSTS) iff there is an ordering $\leq \subseteq S \times S$ on states such that $\leq$ is a wqo, and $\leq$ is "compatible" with $\rightarrow$, where "compatible" means that whenever $s_1 \leq t_1$, and a transition $s_1 \rightarrow s_2$ exists, there also has to exist a transition $t_1 \rightarrow t_2$, such that $s_2 \leq t_2$.

We define a wqo on markings for multi-level NP-nets and show that together with the direct reachability relation $[\rangle$ it forms WSTS. We use a special tree ordering.

Definition 3.2. Let $\langle X, \leq_X \rangle$ be a wqo. A labeled rooted commutative tree $t \in \mathbb{T}(X)$ over X (or just a tree) is a tuple $(\langle \mathit{Nodes}(t), \leq, \alpha_0 \rangle, \mathcal{L})$, where $\mathit{Nodes}(t) = \{\alpha_0, \beta, \gamma \ldots\}$ is a finite set of nodes, supplied by the wqo $\leq$ with a distinguished element α_0, called the root of the tree, and $\mathcal{L} : \mathit{Nodes}(t) \rightarrow X$ — a labeling function, s.t.

– $\alpha_0 \leq \beta$ for all $\beta \in \mathit{Nodes}(t)$,

– if $\beta \leq \gamma$ and $\beta \leq \delta$, then either $\gamma \leq \delta$, or $\delta \leq \gamma$ for all $\beta, \gamma, \delta \in \mathit{Nodes}(t)$.

Two nodes $\alpha, \beta \in \mathit{Nodes}(t)$ are called adjacent, iff $\alpha \leq \beta$ and there is no δ, for which $\alpha \leq \delta \leq \beta$. It's easy to see, that a tree here corresponds to a rooted oriented tree, as it's defined in graph theory. The relation $\leq$ corresponds to the path connectivity between nodes.

Definition 3.3. Let $t, t' \in \mathbb{T}(X)$. We say, that a tree t is *(isomorphically) embedded* in a tree t' (write $t \preceq_X t'$), if there exists a function $\varphi : \mathit{Nodes}(t) \rightarrow \mathit{Nodes}(t')$, such that

– φ is injective,

– φ is monotone and maps adjacent nodes into adjacent ones, i.e. for all adjacent nodes $\alpha, \beta \in \mathit{Nodes}(t)$ if $\alpha \leq \beta$, then $\varphi(\alpha) \leq \varphi(\beta)$ and the nodes $\varphi(\alpha), \varphi(\beta)$ are also adjacent.

– for each node $\alpha \in \mathit{Nodes}(t)$: $\mathcal{L}(\alpha) \leq_X \mathcal{L}'(\varphi(\alpha))$, where $\mathcal{L}, \mathcal{L}'$ are labeling functions for trees t and t' correspondingly.

By $\mathbb{T}_n(X)$ we denote the set of all trees in $\mathbb{T}(X)$, with the height not exceeding n. We prove, that trees of limited height is a wqo over $\preceq_X$, if $\leq_X$ is a wqo.

Lemma 3.4. For every natural $n \geq 0$ a relation $\preceq_X$ is a wqo on $\mathbb{T}_n(X)$ iff $\leq_X$ is a wqo.

This lemma is a variation of Kruskal's theorem [9] and can be proved e.g. by using the ideas of Nash-Williams [12].

Let $M_1, M_2 \in \mathfrak{M}(\mathit{NPN})$ be two markings in *NPN*, $\mathbb{T}(M_1), \mathbb{T}(M_2)$ — corresponding marking trees. Define $M_1 \preceq M_2$ iff $\mathbb{T}(M_1) \preceq_X \mathbb{T}(M_2)$.

Theorem 3.5. Every multi-level NP-net is a WSTS w.r.t. the partial ordering $\preceq$ on the set of its reachable markings.

Proof: The scheme of the proof is as follows. Let *NPN* be a NP-net and let X be the set of node labels in its marking trees. The set X consists of names of net components, places in these net components, tuple operations, natural numbers (not exceeding the maximal arity of places in *NPN*), and a finite number of atom tokens. Hence, X is finite and equality is a wqo on X.

By proposition 2.5 all reachable markings in a multi-level NP-net are feasible, and hence their depth is limited (does not exceed the maximal path length in the acyclic nestedness graph $G_{\lhd}(NPN)$). Then, by Lemma 3.4, the relation $\preceq$ is a wqo on a set of reachable markings in a multi-level NP-net. To show that multi-level NP-nets are WSTS, compatibility of $\preceq$ must also be stated. This can be done by the analysis of all cases of transition firings. □

Since multi-level NP-nets are WSTS, decidability of some important properties can be obtained for them. A net terminates if there exists no infinite execution (*Termination Problem*). The *Control-State Maintainability Problem* is to decide, given an initial marking M and a finite set $Q = \{q_1, q_2, \ldots, q_m\}$ of markings, whether there exists a computation starting from M with all its inner markings covering one of the q_i's. The dual problem, called the *Inevitability Problem*, is to decide whether all computations starting from M eventually visit a state not covering one of the q_i's, e.g. for Petri nets we can ask whether a given place will eventually be emptied. All these problems are decidable for WSTS with decidable $\leq$ and effective *Succ*(s) [7]. Hence, we get

Theorem 3.6. The termination problem, the control-state maintainability (w.r.t. $\preceq$) and the inevitability problems are decidable for multi-level NP-nets.

4. Coverability trees and the termination problem for recursive NP-nets

Solving the termination, the control-state maintainability and the inevitability problems for a multi-level NP-net can be done by building a coverability tree, which due to Theorem 3.5 turns out to be finite. For a recursive NP-net the depth of its reachable markings can be unlimited, and $\preceq$ is not a wqo. An attempt to use another ordering (i.e. Kruskal's tree embedding [9]) gives a wqo, but for the vertical synchronization step compatibility turns out not to be valid. The reason is in a possible dependence of "inner" behavior of a net token from its parent markings, moreover, this dependence my be not local (i.e. the number of influencing upper levels can be unlimited). Further we show, that a special kind of finite coverability tree can be built for recursive NP-nets with localized elements. We use it for solving the termination problem and a special case of the control-state maintainability problem.

The *transitions activity maintainability problem* for a NP-net *NPN* is to decide for a given set of transitions $t_1, \ldots, t_k$, whether there exists a run $M_0[\rangle M_1[\rangle \ldots$ for *NPN*, s.t. in each marking M_i in this rune at least one of $t_1, \ldots, t_k$ is enabled.

For multi-level NP-nets this problem is a special case of the control-state maintainability problem, since it can be reduced to checking $(M_i \geq M^1) \vee \cdots \vee (M_i \geq M^k)$, where M^j is a minimal enabling t_j marking. Compatibility is violated for recursive NP-nets, and in some sense the control-state maintainability loses its meaning in this case.

Note, that in a recursive NP-net the marking depth can increase only after firing a transition with a net constant on one of its outgoing arcs. So, while building a coverability tree for a recursive NP-net we will additionally examine cases, where a net token generates the same net token with the same marking within its own marking.

Definition 4.1. A k-*coverability tree* for a NP-net *RN* with inital marking M_0 is a finite directed tree with nodes labeled by markings in *RN*, s.t.

– the root of the tree is labeled by M_0 and is an inner node of the tree;

– an inner node labeled by M has a descendants labeled M' for each $M' \in Succ(M)$;

– if a node has no descendants, it is declared to be *final*;

– if there is a node, labeled by M, on the path from the root to some node with a label M', s.t. $M \preceq M'$, then M' is called a *covering node* and we say, that M' *covers* M,

– if there is a node, labeled by M, on the path from the root to some node with label M', s.t. both M and M' are obtained after firings, generating net tokens α and β with the same net structures and markings, β is an (not necessary direct) element of α and $\mathcal{T}^k(M, \alpha) \preceq \mathcal{T}^k(M', \beta)$, then the node M' is called an *iterative node*;

– covering and iterative nodes do not have descendants.

Lemma 4.2. For any $k \geq 0$ and for any NP-net *RN* its k-coverability tree is finite and can be effectively constructed.

Proof: On each root path in a k-coverability tree either the depth of all markings is limited and there are two markings $M \preceq M'$, or it is unlimited, and hence along this path a net token generates its own copy within itself an infinite number of times. Then by lemma 3.4 the infinite sequence of these markings contains two markings M and M' with leaf nodes α and β, s.t. $\mathcal{T}^k(M, \alpha) \preceq \mathcal{T}^k(M', \beta)$. □

Theorem 4.3. The termination problem and the transitions activity maintainability problem are decidable for recursive NP-nets with localized elements.

Proof: To solve these problems for a NP-net *NPN* with localized elements of degree k we construct the k-coverability tree for it. A path from the root to a final leaf in this tree corresponds to a finite run. Due to the compatibility property the path to a covering leaf can be repeated an unlimited number of times and corresponds to an infinite run. For an iterative leaf a sequence of firings from generating a net token α to generating β inside of it (cf. definition 4.1) can also be repeated an infinite number of times by virtue of the localization property. Hence, to solve the termination problem one has to check that all leaves in a finite k-coverability tree are final.

NPN has a run maintaining activity of at least one of the transitions $t_1, t_2, \ldots, t_k$ iff its k-coverability tree contains a path from the root to some leaf, s.t. in all its markings at least one of the transitions $t_1, t_2, \ldots, t_k$ is enabled. Obviously, this property can be checked effectively. □

5. Comparing with other models

In this section we compare NP-nets with some other Petri net extensions with respect to expressibility and decidability. First of all, from the decidability of termination (Theorems 3.6, 4.3) it follows, that NP-nets with localized elements are strictly weaker, than Turing machines. On the other hand, it was proved in [1] that recursive NP-nets with autonomous elements can generate any context-free language as its terminal language, while for ordinary Petri nets this is not true. Hence, recursive NP-nets with autonomous elements are strictly more expressive, then ordinary Petri nets.

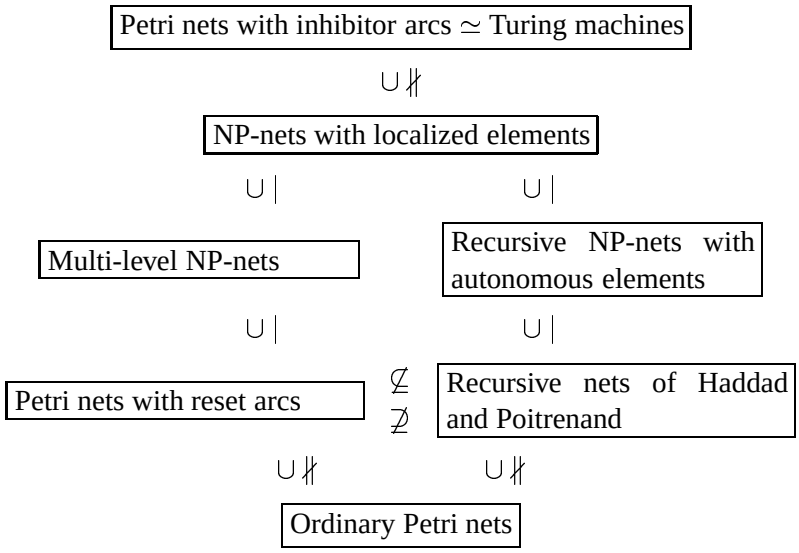

Figure 2.

Petri nets with reset arcs [3] extend the basic model with special "reset" arcs, which denote that the firing of some transitions resets (empties) the corresponding places. In [11] it was proved, that two-level NP-nets are strictly more expressive than Petri nets with reset arcs.

Now we compare recursive NP-nets with the recursive nets of S. Haddad and D. Poitrenand [8]. The recursive nets of Haddad and Poitrenand have the same structure as ordinary Petri nets, but their transitions are of three types: elementary, abstract and final. Firing of an elementary transition is defined by the usual rule. The firing of an abstract transition consumes its input tokens and generates a copy of the initial net with a marking depending on the transition. The firing of a final transition finishes the parent transition, putting tokens to output places of the parent transition and kills the child net. So, unlike recursive NP-nets, in recursive Petri nets transitions instead of tokens are structured objects. In this respect recursive Petri nets are similar to hierarchical nets of Cherkasova and Kotov [2] and nets from an unpublished paper of K.Reinhardt [15] (occasionally, they are also called nested nets).

In the nets of Reinhardt, as well as in the recursive Petri nets of Haddad and Poitrenand, inner nets are consumed, provided its marking enables a special transition. Thus the condition of finishing an inner net run is preserved for "bigger" markings. This monotonicity property makes the reachability problem decidable for the both classes of nets (by reducing to the case of ordinary Petri nets). In the hierarchical nets of Cherkasova and Kotov an inner net is consumed after completing its work (reaching a final marking). This peculiarity in their behavior allows the simulation of Petri nets with inhibitor arcs. Hence, for hierarchical Petri nets reachability and termination problems are undecidable.

The recursive Petri nets of Haddad and Poitrenand can be simulated by recursive NP-nets with autonomous elements, so that an abstract transition is simulated by a transition generating a net token in a special place and a final transition — by a vertical synchronization transition killing this net token.

Figure 2 represents the relative expressiveness of the examined classes of nets. General (without the localization property) NP-nets are not shown in this picture. The question, whether the termination problem is decidable for them, is still open.

References

[1] Bashkin, V.A. and Lomazova, I.A.: "On languages of nested Petri nets". *Proc. Int. Conf. ICIT'99. Pereslavl-Zalessky, December 6-9, 1999* Nauka, Moscow, 1999, 9–13 (In Russian).

[2] Cherkasova, L.A. and Kotov, V.E.: "Structured nets". *Proc. of the 6th MFCS*, LNCS 118, Springer-Verlag, 1981, 242–251.

[3] Dufourd, C., Finkel, A. and Schnoebelen Ph.: "Reset nets between decidability and undecidability", *Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98)*, LNCS 1443, Springer, 1998, 103–115.

[4] Farwer, B.: "A linear logic view of object Petri nets", *Fundamenta Informaticae*, **37**(3), 1999, 225–246.

[5] Farwer, B.: "Linear logic based calculi for object Petri nets", Dissertation, Logos Verlag, ISBN 3-89722-539-5, Berlin, 2000.

[6] Finkel, A.: "Reduction and covering of infinite reachability trees", *Information and Computation*, **89**(2), 1990, 144–179.

[7] Finkel, A. and Schnoebelen, Ph.: "Fundamental structures in well-structured infinite transition systems", *Proc. 3rd Latin American Theoretical Informatics Symposium (LATIN'98), Campinas, Brazil, Apr. 1998*, LNCS 1380, Springer, 1998, 102–118.

[8] Haddad, S. and Poitrenand, D.: "Theoretical aspects of recursive Petri nets", *Proc. ATPN'99*, LNCS 1639, Springer, 1999, 228–247.

[9] Kruskal, J.B.: "Well-quasi-ordering, the tree theorem, and Vázsonyi's conjecture", *Trans. Amer. Math. Soc.* **95**, 1960, 210–225.

[10] Lomazova, I.A.: "Nested Petri nets — a Formalism for Specification and Verification of Multi-Agent Distributed Systems", *Fundamenta Informaticae*, **43**(1–4), 2000, 195–214.

[11] Lomazova, I.A. and Schnoebelen, Ph.: "Some Decidability Results for Nested Petri Nets", *Proc. Andrei Ershov 3rd Int. Conf. Perspectives of System Informatics (PSI'99), Novosibirsk, Russia, Jul. 1999*, LNCS 1755, Springer-Verlag, 1999, 207–219.

[12] Nash-Williams, C.St.J.A.: "On well-quasi-ordering finite trees", *Proc. of the Cambridge Philosophical Society*, **59**(4), 833–835.

[13] Neuendorf, K.-P., Schmidt, K., Kiritsis D. and Xirouchakis, P.: "Workflow Modelling and Analysis with Chameleon Nets", H.-D. Burkhard, L. Czaja and P. Starke (eds.) *Workshop Concurrency, Specification and Programming, 28-30 September 1998*, Humboldt-Universit¨at, Informatik-Berichte, No 110. Berlin, 1998, P. 156–161.

[14] Neuendorf, K.-P., Kiritsis, D. and Xirouchakis, P.: "Chameleon Systems – a Class of Multi-level Petri nets", S. Philippi (ed.) *Workshop Algorithmen und Werkzeuge f¨ur Petrinetze, 2.-3. Oktober 2000*, Koblenz, Germany. Fachberichte Informatik, No 7, Universit¨at Koblenz-Landau, Institut für Informatik, 2000, 90–95.

[15] Reinhardt, K.: "Reachability in Petri nets with inhibitor arcs", *Unpublished manuscript*, www-fs.informatik.uni-tuenbingen.de/reinhard/publ.html#Rei95, 1995.

[16] Valk, R.: "Petri Nets as Token Objects: An Introduction to Elementary Object Nets", *Proc. Int. Conf. on Application and Theory of Petri Nets*, LNCS 1420, Springer-Verlag, 1998, 1–25.

[17] Valk, R.: "Relating different semantics for Object Petri nets", *Technical report*, FBI-HH-B-226/00, Dept.of Computer Sience, University of Hamburg, 2000.

[18] Zapf, M. and Heinzl, A.: "Techniques for integrating Petri nets and object-oriented concepts", *Working Papers in Information Systems*, No 1/1998. University of Bayreuth, 1998.