Fundamenta Informaticae XXI (2008) 1001–1027 1001
IOS Press

Name creation vs. Replication in Petri Net Systems *

Fernando Rosa-Velardo, David de Frutos-Escrig
Sistemas Informáticos y Computación
Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid
{*fernandorosa,defrutos*}*@sip.ucm.es*

Abstract. We study the relationship between name creation and replication in a setting of infinite-state communicating automata. By name creation we mean the capacity of dynamically producing pure names, with no relation between them other than equality or inequality. By replication we understand the ability of systems of creating new parallel identical threads, that can synchronize with each other. We have developed our study in the framework of Petri nets, by considering several extensions of P/T nets. In particular, we prove that in this setting name creation and replication are equivalent, but only when a garbage collection mechanism is added for idle threads. However, when simultaneously considering both extensions the obtained model is, a bit surprisingly, Turing complete and therefore, more expressive than when considered separately.

Keywords: Petri nets, pure names, infinite state systems, decidability, multithreading, security, choreography.

1. Introduction

Extensive work has been developed in the last years in the field of multithreaded programs [22]. In general, once thread synchronization is allowed, reachability becomes undecidable [21], so that the effort is devoted to compute overapproximations of the set of reachable states. For instance, [3] studies the case where threads are finite state, [4] considers a framework to statically analyze multithreaded systems with a fixed number of infinite-state communicating threads, and in [5] that work is extended to cope with dynamic creation of threads.

Address for correspondence: Facultad de Ciencias Matemáticas. Plaza de las Ciencias, 3-28040 Madrid (Spain)
*This paper is an extended version of [27], and is partially supported by the Spanish projects DESAFIOS TIN2006-15660-C02-02, WEST TIN2006-15578-C02-01 and PROMESAS-CAM S-0505/TIC/0407.

Dynamic name generation has also been thoroughly studied, mainly in the field of security and mobility [19, 15]. Paradigmatic examples of nominal calculi are the π-calculus [18], and the Ambient Calculus [7]. Since the early versions of all these calculi are Turing complete, efforts have been focused on the static analysis of undecidable properties [8, 20] and in the restriction of the original formalisms [6, 28] to get more manageable languages where some interesting properties become decidable.

In this paper we investigate the relationships between name creation and replication in this setting of infinite-state communicating automata, that in our case are Petri nets. By name creation we mean a mechanism to generate fresh names that can be communicated between components, with no relation between them other than equality or inequality, which have been called pure names in [15]. By replication we understand the capability of processes of creating an exact replica of themselves, though always starting its execution in a fixed initial state. Once created, these replicated processes evolve in an independent way, though possibly interacting with other processes.

We will start by defining *basic net systems*, which are essentially sets of ordinary P/T nets that can synchronize with each other. Then we will extend the basic model first with a mechanism for name creation (*ν-net systems*), then with a mechanism for replication (*RN systems*) and, finally, we will extend it in both directions (*ν-RN systems*). We will prove that these two mechanisms are equivalent, in the sense that they simulate each other. On the one hand, names can be used to represent the states of the different threads of a process that are running in parallel over a single copy of the (virtually) replicated process. On the other hand, different copies of the same component can be used to mimic the effect of adding pure names. In the proof of this equivalence it is essential to consider a garbage collection mechanism that removes idle threads, given that the corresponding garbage collection mechanism for names is implicit in the model (the set of used names is not part of the state). In fact, reachability is undecidable for the extension with name creation (equivalently with replication and garbage collection), but decidable if no such mechanism is considered.

However, though name creation and replication are equivalent, it turns out that they do not overlap, in the sense that a model dealing simultaneously with both surpasses the strength of any of these two features, reaching indeed Turing completeness. The intuitive explanation of why this is true is that once we have names, they can be used to distinguish between what at first were indistinguishable components, so that now threads can have a unique personality.

Another formalism that encompasses both dynamic name generation and replication is TDL [11]. In TDL there are primitives both for name creation and thread creation. However, no garbage collection mechanism is considered, neither for names nor for threads. As a consequence, and quite surprisingly, reachability is decidable, though coverability is undecidable.

All of our models are structured by means of components. In the case of basic net systems and ν-net systems, components are not an essential feature of the model. In fact, it is easy to prove that they can be flattened to an equivalent system with a single component. In the case of RN systems, it is natural to structure systems by means of components that, when replicate, create copies of themselves. Even so, we could choose to have RN systems in which all components have the same structure. However, we prefer to maintain the structuring of systems by means of different components for several reasons. On the one hand, they can be interesting per se and the difficulty of the development is not significantly increased. When we first introduce each of the models we will immediately prove the corresponding flattening result (with different subtleties) and in the main proofs we will only consider the simulation of systems with a single component. On the other hand, having components allows us to carry a more incremental presentation of the different models. For instance, RN systems are just basic systems with a special type

of transitions, that cause the replication of the component. Finally, some times it is convenient to be able to deal with different components, as in the examples or in the proof of Turing completeness of ν-RN systems or with synchronizations as in the proof of the simulation of replication using names.

The remainder of the paper is structured as follows. In Sect. 2 we set the notations we will use throughout the paper. Sect. 3 presents the basic model, that will be the starting point of our work. Sect. 4 extends the basic model with a mechanism for name creation and name communication, and gives a brief insight of the expressive power of the obtained model. Sect. 5 extends again the basic model, but now with a replication primitive. In Sect. 6 we will prove the equivalence between the two extensions. In Sect. 7 we consider the model obtained when introducing the two features at the same time, and prove its Turing-completeness. Finally, Sect. 8 presents our conclusions and some directions for further work.

2. Preliminaries

First, let us introduce some notations that we will use throughout the paper. Given an arbitrary set A, we will denote by $\mathcal{MS}(A)$ the set of finite multisets of A, that is, the set of mappings $m : A \to \mathbb{N}$. We denote by $S(m)$ the support of m, that is, the set $\{a \in A \mid m(a) > 0\}$ and by $|m| = \sum_{a \in S(m)} m(a)$ the cardinality of m. Given two multisets $m_1, m_2 \in \mathcal{MS}(A)$ we denote by $m_1 + m_2$ the multiset defined by $(m_1 + m_2)(a) = m_1(a) + m_2(a)$. We will write $m_1 \subseteq m_2$ if $m_1(a) \leq m_2(a)$ for every $a \in A$. In this case, we can define $m_2 - m_1$, given by $(m_2 - m_1)(a) = m_2(a) - m_1(a)$. We will denote by $\sum$ the extended multiset sum operator and by $\emptyset \in \mathcal{MS}(A)$ the multiset $\emptyset(a) = 0$, for every $a \in A$. If $f : A \to B$ and $m \in \mathcal{MS}(A)$, then we define $f(m) \in \mathcal{MS}(B)$ by $f(m)(b) = \sum_{f(a)=b} m(a)$. Every partial order $\leq$ defined in A induces a partial order $\sqsubseteq$ in $\mathcal{MS}(A)$, given by $\mathcal{A} \sqsubseteq \mathcal{B}$ if there is an injection $h : \{(a, i) \mid a \in S(\mathcal{A}), i \in \{1, \ldots, \mathcal{A}(a)\}\} \to \{(b, j) \mid b \in S(\mathcal{B}), j \in \{1, \ldots, \mathcal{B}(b)\}$ such that for all (a, i), if $h(a, i) = (b, j)$ then $a \leq b$[1]. We write $s < s'$ if $s \leq s'$ and $s' \not\leq s$. A partial order is a well-quasi order (wqo) [13] if for every infinite chain $s_0, s_1, \ldots$ there are i and j with $i < j$ such that $s_i \leq s_j$.

Next we define Place/Transition nets with weighted arcs, in order to establish the notations we will use along the paper.

Definition 2.1. A Place/Transition net (P/T net) is a tuple $N = (P, T, F)$, where P and T are disjoint sets of places and transitions, respectively, and $F : (P \times T) \cup (T \times F) \to \mathbb{N}$. For a transition $t \in T$ we take ${}^\bullet t \in \mathcal{MS}(P)$ as ${}^\bullet t(p) = F(p, t)$. Analogously, $t^\bullet(p) = F(t, p)$. A marking of N is $M \in \mathcal{MS}(P)$. We say a transition t is enabled for M if ${}^\bullet t \subseteq M$, in which case it can be fired, reaching marking $M' = M - {}^\bullet t + t^\bullet$. We will write $M \xrightarrow{t} M'$ if M' is reached by M when t is fired.

As it is standard, we draw places as circles and transitions and boxes. Whenever $F(p, t) > 0$ we draw an arrow from p to t, labelled by $F(p, t)$ if $F(p, t) > 1$ or without a label if $F(p, t) = 1$ (analogously for $F(t, p)$). We write $M \to M'$ if there is some t such that $M \xrightarrow{t} M'$, and $\to^*$ to denote the reflexive and transitive closure of $\to$. We will use analogous notations for the rest of the models in the paper.

Along the paper we assert several times that a model $\mathbf{M}'$ simulates another model $\mathbf{M}$. By that, unless we explicitly say something else, we mean that for every system $\mathcal{N}$ in $\mathbf{M}$ there is $\mathcal{N}' = F(\mathcal{N})$ in $\mathbf{M}'$,

[1] It is more standard to present the multiset order as $\{a_1, \ldots, a_n\} \sqsubseteq \{b_1, \ldots, b_m\}$ if there is $h : \{1, \ldots, n\} \to \{1, \ldots, m\}$ injective such that $a_i \leq b_{h(i)}$ for all i, though we prefer our equivalent presentation for convenience in our proofs.

where F is a computable function, such that the transition systems generated by the semantics of $\mathcal{N}$ and $\mathcal{N}'$ are isomorphic. Therefore, reachability in $\mathcal{N}$ and $\mathcal{N}'$ are equivalent. In all of those models we will consider a partial order $\sqsubseteq$ relating their states, that in this setting we call markings. This order induces a coverability problem, namely that of deciding if a marking M can be covered, that is, if there is a reachable marking M' such that $M \sqsubseteq M'$. The simulations are such that the mentioned isomorphisms also preserve the considered orders between markings. More precisely, if h is that isomorphism then we will prove in each case that $M \sqsubseteq M' \Leftrightarrow h(M) \sqsubseteq h(M')$. When this happens, M can be covered from M_0 if and only if $h(M)$ can be covered from $h(M_0)$, so that coverability is also preserved. In our figures we will denote by $\mathcal{N} \rightsquigarrow \mathcal{N}'$ the simulation of $\mathcal{N}$ by $\mathcal{N}'$.

3. The basic model

In order to concentrate on the two features of interest we will first consider a very basic model, based on Petri Nets, which could be considered not too useful in practice, but which will be an adequate starting point for later extensions. This model will consist on sets of ordinary P/T nets, that we call component nets, that are like the ones defined in the previous section, except for their capability to interact with each other. For that purpose we will consider a set $\mathcal{S}$ of service names, endowed with a function $arity : \mathcal{S} \to \mathbb{N}$ and we take the set of synchronizing labels $Sync = \{s(i) \mid s \in \mathcal{S},\ 1 \leq i \leq arity(s)\}$. If $arity(s) = 2$ then we will write $s?$ and $s!$ instead of $s(1)$ and $s(2)$, respectively, that can be interpreted as the offer and request of some service. We denote by $\mathcal{A}$ the set of labels with arity one, and identify $s \in \mathcal{A}$ with $s(1)$.

Definition 3.1. A component net is a labelled Petri net $N = (P, T, F, \lambda)$, that is, (P, T, F) is a P/T net and λ is a function from T to the set of labels $Sync$. A marking M of N is a finite multiset of places of N, that is, $M \in \mathcal{MS}(P)$. A basic net system is a set $\mathcal{N}$ of pairwise disjoint component nets. A marking of $\mathcal{N}$ is a set of markings of its components, one marking per component.

Therefore, net components are as ordinary P/T nets, except for the fact that their transitions are labelled by elements in $Sync$. For a net system $\mathcal{N}$ we will denote by P, T, F and λ the union of the corresponding elements in each net component when there is no confusion. Analogously, we will consider markings as multisets of P, related also by multiset inclusion. Now let us define the firing rules of transitions.

Definition 3.2. Let $\mathcal{N}$ be a basic net system and $s \in \mathcal{S}$ with $arity(s) = n$. The transitions in a tuple $\overline{t} = (t_1, \ldots, t_n)$ are said to be compatible for s if $\lambda(t_i) = s(i)$ for all $i \in \{1, \ldots, n\}$. We write ${}^\bullet\overline{t} = \sum_{i=1}^{n} {}^\bullet t_i$ and $\overline{t}^\bullet = \sum_{i=1}^{n} t_i^\bullet$. A tuple of synchronizing transitions $\overline{t}$ is enabled at marking M if ${}^\bullet\overline{t} \subseteq M$. The reached marking of $\mathcal{N}$ after the firing of $\overline{t}$ is $M' = M - {}^\bullet\overline{t} + \overline{t}^\bullet$.

Basic net systems are composed of sets of ordinary Place/Transition nets with weighted arcs. However, their transitions do not fire in isolation, when every precondition has enough tokens, as in P/T nets, but when this happens for a tuple of compatible transitions, which then can fire synchronously (see right of Fig. 1). The compatibility condition is merely syntactical: their labels must match. Autonomous transitions, those that are labelled by a label of arity one, behave exactly as the transitions in P/T nets (see left of Fig. 1).

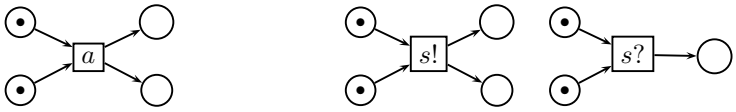

Figure 1. Autonomous (left) and synchronizing (right) transitions

As we have said before, basic net systems will be the starting point for the other extensions we will see in forthcoming sections and, in fact, basic nets only add some syntactic sugar to P/T nets, as we prove in the following proposition. From now on, we will write $t \in (t_1, \ldots, t_n)$ if $t \in \{t_1, \ldots, t_n\}$.

Proposition 3.1. Every Basic Net System can be simulated by a P/T net.

Proof:
The proof basically consists on having a different transition for every tuple of compatible synchronizing transitions, so that the firing of the tuple $\overline{t}$ is simulated by the firing of transition $\overline{t}$. In [14] we proved this result for a model which also considered localities, so that each component is localized and can only synchronize with co-located components, thus proving that these locations do not introduce any further expressive power.

Formally, let us consider a Basic Net System $\mathcal{N} = \{N_1, \ldots, N_n\}$ with $N_i = (P_i, T_i, F_i, \lambda_i)$. As usual, we denote by P, T, F and λ the union of the corresponding elements. We define the P/T net $\mathcal{N}^* = (P, T^*, F^*)$, where:

- $T^* = \{(t_1, \ldots, t_m) \mid t_i \in T,\ \lambda(t_i) = s(i) \text{ and } arity(s) = m\}$,
- $F^*(p, \overline{t}) = \sum_{t \in \overline{t}} F(p, t)$ and $F^*(\overline{t}, p) = \sum_{t \in \overline{t}} F(t, p)$.

Since the set of places coincide, every marking of $\mathcal{N}$ represents also a marking of $\mathcal{N}^*$. Then we can see that $M \xrightarrow{\overline{t}} M'$ is a firing of $\mathcal{N}$ if and only if it is a firing of $\mathcal{N}^*$. Indeed, it holds that ${}^\bullet(t_1, \ldots, t_m) = \sum_{i=1}^{m} {}^\bullet t_i$ and, analogously, $(t_1, \ldots, t_m)^\bullet = \sum_{i=1}^{m} t_i^\bullet$. Then ${}^\bullet\overline{t}$ considering $\overline{t} \in T \times \ldots \times T$ is equal to ${}^\bullet\overline{t}$ considering $\overline{t} \in T^*$, and the same happens with the postconditions of $\overline{t}$. Therefore, the generated transition systems are isomorphic. Trivially, the order is preserved and the thesis follows. □

Since reachability and coverability are decidable for P/T nets, we have the following:

Corollary 3.1. Reachability and coverability are decidable for Basic Net Systems.

4. Name creation

In this section we extend the previous model with the capability of name management. Names can be created, communicated between components and used to restrict synchronizations to happen only between components that know a particular name, as we have illustrated with several examples in [24]. We can use this mechanism to deal with authentication issues in our systems. We formalize the latter by replacing ordinary tokens by distinguishable tokens taken from an arbitrary infinite set Id, thus adding a touch of colour to our nets. In order to handle these colours, we need matching variables labelling the arcs of the nets, taken from a set Var. Moreover, we add a primitive capable of creating fresh names, formalized by means of a special variable $\nu \in Var$, that can only appear in postcondition arcs.

Definition 4.1. A ν-net component is a labelled coloured Petri Net $N = (P, T, F, \lambda)$, where

- P and T are finite disjoint sets of places and transitions of the net, respectively,
- $F : (P \times T) \cup (T \times P) \to \mathcal{MS}(Var)$ defines the set of arcs of the net,
- $\lambda : T \to Sync$ is a function labelling transitions,

such that $\nu \notin pre(t)$ for every $t \in T$, where $pre(t) = \bigcup_{p \in P} S(F(p,t))$, $post(t) = \bigcup_{p \in P} S(F(t,p))$ and $Var(t) = pre(t) \cup post(t)$.

A ν-net component is a special kind of labelled coloured Petri Net with only one colour type for identifiers, taken from an infinite set Id, except for the special variable ν. Unlike for ordinary Coloured Petri Nets, where arbitrary expressions over some syntax can label arcs, we only allow multisets of variables, that are used to specify the flow of tokens from preconditions to postconditions. In particular, this means that only equality of identifiers can be checked by matching.[2] When $F(p,t) = \emptyset$ we do not draw any arc from p to t. Otherwise, we draw an arc from p to t labelled by $F(p,t)$. If $F(p,t)$ is a singleton $\{x\}$ we will just label it by x (analogously for $F(t,p)$).

Markings of ν-nets must specify not only how many tokens are there in each place, but also what are those tokens.

Definition 4.2. A marking of a ν-net $N = (P, T, F, \lambda)$ is a function $M : P \to \mathcal{MS}(Id)$. We denote by $S(M)$ the set of names in M, that is, $S(M) = \bigcup_{p \in P} S(M(p))$.

Definition 4.3. A ν-net system $\mathcal{N}$ is a set of disjoint ν-net components. A marking of $\mathcal{N}$ is a collection of markings of its components, one marking each.

For a tuple of transitions $\overline{t} = (t_1, \ldots, t_n)$ we denote by $post(\overline{t}) = \bigcup_{i=1}^{n} post(t_i)$, $pre(\overline{t}) = \bigcup_{i=1}^{n} pre(t_i)$ and $Var(\overline{t}) = post(\overline{t}) \cup pre(\overline{t})$. If ι is a function mapping identifiers to identifiers and M is a marking, we denote by $\iota(M)$ the marking given by $\iota(M)(p) = \iota(M(p))$. Moreover, we will write $F(p, \overline{t}) = \sum_{t \in \overline{t}} F(p,t)$ and $F(\overline{t}, p) = \sum_{t \in \overline{t}} F(t,p)$. Next let us see the compatibility conditions for tuples of transitions in ν-net systems.

Definition 4.4. Let $\overline{t} = (t_1, \ldots, t_n)$ be a tuple of transitions of a ν-net system. We say the transitions in $\overline{t}$ are compatible if:

- $\lambda(t_i) = s(i)$ for some $s \in \mathcal{S}$ with $arity(s) = n$,
- $post(\overline{t}) \setminus \{\nu\} \subseteq pre(\overline{t})$.

Therefore, for a tuple of transitions to be compatible they have to be compatible in the sense of basic net systems, and satisfy an extra condition, namely that every variable appearing in an outgoing arc of any transition (except ν) must also appear in some of the incoming arcs of some of the transitions (see Fig. 2).

[2] In fact, analogous results could be obtained if we could also check inequality.

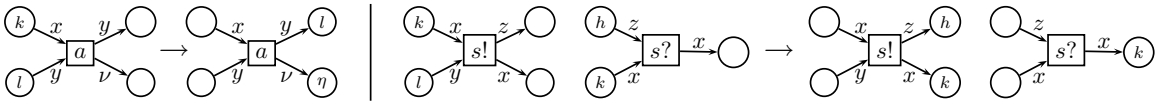

Figure 2. Autonomous (left) and synchronizing (right) transitions

Intuitively, ordinary variables can only cause the flow of tokens from preconditions to postconditions. Tokens can be removed from preconditions or even replicated in postconditions, but not created. This is the task of the variable ν. The only way transitions can produce new names is by means of the variable ν attached to some of its outgoing arcs, which always produces a new name, not present in the current marking of the system.

Notice that, in particular, if t is an autonomous transition then it must be the case that every variable annotating a postcondition arc, except ν, also annotates some precondition arc (see left of Fig. 2).

Like in every kind of high-level net, transitions are fired with respect to some mode, that establishes which tokens are taken from preconditions. A mode of a tuple of compatible transitions $\overline{t}$ is a mapping $\sigma : Var(\overline{t}) \to Id$, instantiating every variable in the adjacent arcs of $\overline{t}$ to some identifier.

Definition 4.5. Let $\mathcal{N}$ be a ν-net system and M a marking of $\mathcal{N}$. We say that M enables the tuple of compatible transitions $\overline{t}$ with mode σ whenever:

- If $\nu \in Var(t)$ then $\sigma(\nu) \notin S(M)$,
- $\sigma(F(p,\overline{t})) \subseteq M(p)$ for all $p \in P$.

The reached marking of $\mathcal{N}$ after the firing of $\overline{t}$ with mode σ is $M'(p) = M(p) - \sigma(F(p,\overline{t})) + \sigma(F(\overline{t},p))$ for every $p \in P$, denoted by $M \stackrel{\overline{t}(\sigma)}{\to} M'$.

Notice the condition $\sigma(\nu) \notin S(M)$ for the enabling of a transition, that causes the creation of fresh (equal) identifiers in all the places reached by arcs labelled by that special variable. Since $F(p,\overline{t})$ and $F(\overline{t},p)$ are both multisets of variables and σ maps variables to identifiers, $\sigma(F(p,\overline{t}))$ and $\sigma(F(\overline{t},p))$ are both multisets of identifiers, that can be removed from, or added to, $M(p)$.

By means of synchronization we achieve both name communication and restriction of communications by name matching. If $\overline{t} = (t_1, t_2)$, the former is obtained by using a variable in $post(t_1) \setminus pre(t_1)$ and in $pre(t_2)$ (or vice versa), as the variable z in Fig. 2 right, which produces the communication of the name h. The latter is obtained by using the same label both in $pre(t_1)$ and $pre(t_2)$, which forces the matching between the consumed tokens, as happens in Fig. 2 right with the variable x and the name k.

Let us define the coverability problem in the setting of ν-net systems. We could think that the order in which we are interested is given by $M_1 \sqsubseteq M_2 \Leftrightarrow M_1(p) \subseteq M_2(p)$ for all $p \in P$. However, this order is too restrictive, because it does not take into account the abstract nature of fresh names, that is, the fact that any name not in the current marking can be chosen when a transition with a variable ν is fired.

In fact, it is easy to see that $\sqsubseteq$ is not a wqo: If N is a ν-net component with a single place, let $(a_i)_{i=1}^{\infty}$ be a sequence of pairwise different identifiers. Then it suffices to consider $M_i(p) = \{a_i\}$ for all i to obtain a sequence $M_1, M_2, M_3, \ldots$ that satisfies for all $i < j$, $M_i \not\sqsubseteq M_j$.

Therefore, reachability (or coverability) of a given marking is equivalent to reachability (or coverability) of any marking obtained by renaming those new names. However, for the sake of homogeneity, we allow the renaming of every name, even those that already appeared in the initial marking (which are a fixed number of identifiers). In order to capture these intuitions, we introduced in [24] a notion of α-equivalence between markings, borrowed from the renaming of bound names in the π-calculus [18]:

we identify markings under renaming of pure names. We define $\equiv_\alpha$ as the least equivalence relation between markings such that $M \equiv_\alpha M[\eta'/\eta]$ for $\eta \in S(M)$ and $\eta' \notin S(M)$ and take

$$M_1 \sqsubseteq_\alpha M_2 \Leftrightarrow \text{there is some } M \text{ such that } M_1 \equiv_\alpha M \sqsubseteq M_2.$$

An alternative way of saying that $M_1 \sqsubseteq_\alpha M_2$ is stating the existence of an injection $\iota : S(M_1) \to S(M_2)$ such that $\iota(M_1) \sqsubseteq M_2$. Let us call ν-APNs to those ν-net components that only have autonomous transitions. Moreover, we will only consider the subclass of ν-APNs, that we will call ν^*-APNs, whose name creating transitions only involve two different variables, that is, those such that for every $t \in T$ with $\nu \in Var(t)$ it holds $|Var(t)| = 2$. Then we have the following result.

Proposition 4.1. Every ν-net system can be simulated by a ν^*-APN.

Proof:
See the appendix. This simulation is not a step by step simulation, like the ones discussed in Sect. 2, but it preserves decidability of reachability and coverability. □

In [24] we proved several interesting (un)decidability results for a model we call Mobile Synchronizing Petri Nets, that are essentially these ν-net systems, but with some syntactic sugar supporting a flat kind of mobility and allowing only two-way synchronizations. In particular, though reachability turns out to be undecidable when adding names, as proved in [16] for minimal OO-nets, coverability remains decidable, meaning that the expressive power of this model lies somewhere in between ordinary P/T nets and Turing machines. The latter was proved by taking into account the abstract nature of created names. More precisely, we proved that they are well structured systems [13] and, in particular, the order $\sqsubseteq_\alpha$ is a wqo. All these results can be transferred to the model considered in this paper. For the details see the appendix.

Corollary 4.1. Reachability is undecidable, and coverability is decidable, for ν-net systems.

5. Replication

In this section we introduce a replication primitive, that creates an identical copy of the net component that executes it, marked in some fixed way. This primitive makes very desirable the structuring of the system by means of components. Replication, together with synchronization, can be used to implement a spawning primitive, that creates a different component taken from a finite set.

Definition 5.1. A Replicated Net (RN) is a labelled Petri net $N = (P, T, F, \lambda)$, that is, (P, T, F) is a P/T net and λ is a function from T to the set of labels $Sync \cup \mathcal{MS}(P)$. A marking of N is $M \in \mathcal{MS}(P)$. We denote by $Markings(N)$ the set of markings of N.

The only syntactical difference with basic nets is that now some transitions may be labelled by a multiset of places, that is, by a marking. That multiset corresponds to the initial marking of the net that is created when firing it, always fired in isolation, without having to synchronize with other transitions. The meanings of $t^\bullet$ and ${}^\bullet t$ are the same as in Basic Net Systems.

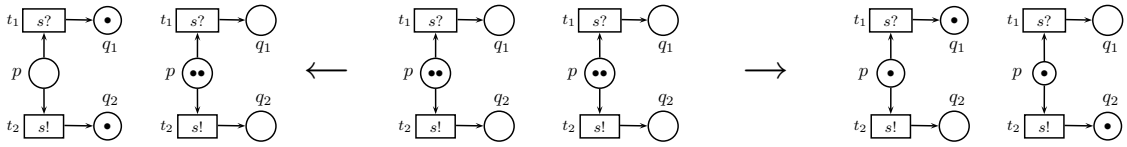

Figure 3. RN system with marking $\mathcal{M} = \{M, M\}$, where $M = \{p, p\}$ and two possible firings

Definition 5.2. An RN system $\mathcal{N}$ is a set of pairwise disjoint RNs. A marking $\mathcal{M}$ of $\mathcal{N}$ is a finite multiset of markings of its components. We denote by $Markings(\mathcal{N})$ the set of markings of $\mathcal{N}$.

We represent the presence in the system of several copies of a net by considering several markings of that net, so that now each $N \in \mathcal{N}$ does not represent a single net component, but one of the possible types of nets that can appear in the system. Therefore, unlike for ordinary basic nets, a marking is not just a collection of individual markings, one marking per component, but a multiset of markings, and not necessarily one marking per component, but any natural number for each component.

Let us now define the behaviour of RN systems. The key concept is that of mode. In the case of RN systems modes specify which of the components that belong to the current marking $\mathcal{M}$ fire each of the transitions. In the case of autonomous transitions, it is enough to specify a marking $M \in S(\mathcal{M})$. If there are several components identified by M in $\mathcal{M}$ then it does not matter which one actually fires the transition, since the reached marking will be the same whichever that component was. However, in the case of synchronizing (non-autonomous) transitions we have to be more careful. Consider the RN system in the middle of Fig. 3, that can fire the tuple of transitions (t_1, t_2). In this case, it is not enough to identify for each transition a marking in $S(\mathcal{M})$, since the two components are represented by the same marking M. It could be the case that one of the components synchronized with itself, reaching the marking in the left of Fig. 3, or that the two components synchronized with each other, thus reaching the marking in the right. Therefore, in order for modes to completely identify the components firing each of the transitions they have to specify not only the marking that identifies the component, but also which transitions are fired by the same component and which are not. One way of doing this is to (temporarily) number the components identified by the same marking. In the case of Fig. 3 we can say that the component in the left is the first one and the one in the right is the second component. Then, the two different modes in which the pair (t_1, t_2) can be fired are the one in which both transitions are fired by the first component, and the one in which t_1 is fired by the first component and t_2 is fired by the second component. Moreover, the particular numbering of the components we choose is irrelevant. Indeed, we could also choose the second component for the firing of both t_1 and t_2 in the first case, or the second component for t_1 and the first component for t_2, in the second case, and we would have obtained the same markings. We will formalize this intuition in Prop. 5.1.

Definition 5.3. Let $\mathcal{N}$ be an RN system $\overline{t} = (t_1, \ldots, t_n)$ a tuple of transitions of $\mathcal{N}$. A mode of $\overline{t}$ is a mapping $\sigma : \overline{t} \to Markings(\mathcal{N}) \times \mathbb{N}$ such that if t is a transition of $N \in \mathcal{N}$ then $\sigma(t) = (M, i)$ with $M \in Markings(N)$.

Therefore, for a given (M, i), all the transitions t such that $\sigma(t) = (M, i)$ are fired by the i-th copy of M. Notice that, in the case of tuples with a single transition, the previous definition boils down to specifying a pair (M, i) and, as we said before, the particular i chosen is irrelevant. Therefore, for singleton tuples it is enough to say what marking is firing the transition, so that we can safely ommit the second component and say that a mode for a singleton tuple is just a marking M. As usual, we say the transitions in $\overline{t} = (t_1, \ldots, t_n)$ are compatible synchronizing transitions if $\lambda(t_i) = s(i) \in Sync$ for some

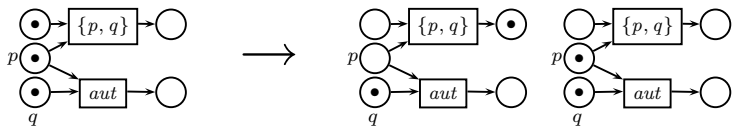

Figure 4. RN system

s with $arity(s) = n$.

Definition 5.4. Given an RN system $\mathcal{N}$, $\mathcal{M}$ a marking of $\mathcal{N}$ and $\overline{t}$ a tuple of compatible synchronizing transitions, we say $\overline{t}$ is enabled in mode σ if:

- $\sigma(t) = (M, i) \Rightarrow M \in S(\mathcal{M})$ and $i \in \{1, \ldots, \mathcal{M}(M)\}$,
- for every (M, i), $\sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} {}^\bullet t \subseteq M$.

The reached marking after the firing of $\overline{t}$ in mode σ is

$$\mathcal{M}' = \{M - \sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} {}^\bullet t + \sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} t^\bullet \mid M \in S(\mathcal{M}),\ 1 \leq i \leq \mathcal{M}(M)\}.$$

A component (M, i) must fire all transitions $t \in \overline{t}$ such that $\sigma(t) = (M, i)$, for which two conditions must hold. In the first place, M must be in the current marking $\mathcal{M}$ (condition $M \in S(\mathcal{M})$) and there must be at least i copies of M in $\mathcal{M}$ (condition $i \leq \mathcal{M}(M)$). The second condition is the usual enabling condition, stating that preconditions must have enough tokens.

In Fig. 3 the pair (t_1, t_2) is fired with mode $\sigma_1(t_1) = \sigma_2(t_2) = (M, 1)$ (left of the figure) and with mode $\sigma_2(t_1) = (M, 1)$ and $\sigma_2(t_2) = (M, 2)$ (right or the figure). However, we might as well have considered a different order, and we would have obtained the same results after their firing, as formalized in the following proposition.

Proposition 5.1. Let $\mathcal{M}$ be a marking, $\overline{t}$ a tuple of compatible transitions, σ a mode for $\overline{t}$ and τ_M a permutation of $\{1, \ldots, \mathcal{M}(M)\}$ for every $M \in S(\mathcal{M})$. If $\bar{\sigma}$ is the mode of $\overline{t}$ defined by $\bar{\sigma}(t) = (M, \tau_M(i))$ when $\sigma(t) = (M, i)$ then $\mathcal{M} \stackrel{\overline{t}(\sigma)}{\rightarrow} \mathcal{M}' \Leftrightarrow \mathcal{M} \stackrel{\overline{t}(\bar{\sigma})}{\rightarrow} \mathcal{M}'$.

Proof:
By definition of $\bar{\sigma}$, $\sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} {}^\bullet t = \sum_{\substack{t \in \overline{t} \\ \bar{\sigma}(t) = (M,\tau_M(i))}} {}^\bullet t$. Then,
$t(\sigma)$ is enabled $\Leftrightarrow$ for all (M, i), $\sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} {}^\bullet t \subseteq M \Leftrightarrow \sum_{\substack{t \in \overline{t} \\ \bar{\sigma}(t) = (M,\tau_M(i))}} {}^\bullet t \subseteq M$. Since τ_M is a bijection, this happens if and only if $\sum_{\substack{t \in \overline{t} \\ \bar{\sigma}(t) = (M,i)}} {}^\bullet t \subseteq M$ for all $(M, i) \Leftrightarrow \overline{t}(\bar{\sigma})$ is enabled in $\mathcal{M}$.

Analogously, we can see that the reached markings are the same, and the thesis follows. □

Replicating transitions are fired in isolation, without having to synchronize with other transitions. In particular, as we said before, modes are specified by markings, so that we can write without confusion $\sigma(t) \in \mathcal{MS}(P)$, when σ is a mode of a replicating transition t. With these notations the definitions of enabling and firing of replicating transitions are much simpler.

Definition 5.5. Let t be a transition of an RN system $\mathcal{N}$ with $\lambda(t) \in \mathcal{MS}(P)$ and $\mathcal{M}$ a marking of $\mathcal{N}$. We say t is enabled in mode σ if $\sigma(t) \in S(\mathcal{M})$ and ${}^\bullet t \subseteq \sigma(t)$. In this case $t(\sigma)$ can be fired, reaching $\mathcal{M}' = \mathcal{M} - \{\sigma(t)\} + \{\sigma(t) - {}^\bullet t + t^\bullet, \lambda(t)\}$.

Therefore, replicating transitions behave like transitions of arity one, as defined in Def. 5.3 and Def. 5.4, except for the fact that the new component $\lambda(t)$ is created. Fig. 4 illustrates an RN system, initially with one component that can either fire an autonomous transition (labelled by aut) or create a replica that can only fire that autonomous transition, and chooses to do the latter.

So far, net components can only be created, but never removed. In such a case, we proved in [26] that both reachability and coverability are decidable. The former is proved by taking into account that markings must remember the number of created components, even if some of them are deadlocked. Since this assumption is not very realistic, next we introduce a garbage collection mechanism, that allows us to remove from markings those nets that do not currently have any token, which form a subset of deadlocked components, if we assume that every transition has at least one precondition. Therefore, from now on we will only consider RN components in which there are no transitions without preconditions. This can be easily done without loss of generality. Indeed, if there is a transition without a precondition we can add a new place that is both a precondition and a postcondition of it, marked in the initial marking. This new place will always be marked, so that the behaviour of the net does not change.

Let us define the mentioned garbage collection mechanism, simply by disregarding the empty marking as a possible marking of a net.

Definition 5.6. Given an RN system $\mathcal{N}$ we define $\equiv$ as the least congruence on multisets of markings of $\mathcal{N}$ such that $M \equiv M + \{\emptyset\}$.

If we identify markings up to $\equiv$, every time a net component becomes empty, we will ignore it. We will write g-RN to denote RN systems with garbage collection.

Markings of g-RN systems are multisets of markings of its components. Therefore, the order between markings of components (multiset inclusion) induces an order between markings of g-RN systems, as defined in Sect. 2. We will just denote this order by $\sqsubseteq$. Intuitively, a marking $\mathcal{M}_1$ is less or equal than $\mathcal{M}_2$ whenever each component in $\mathcal{M}_1$ can be covered by a different component in $\mathcal{M}_2$.

To conclude this section let us see a technical result, stating that it is enough to consider g-RN systems with a single type of components.

Proposition 5.2. Every g-RN system can be simulated by a g-RN component.

Proof:
Given a g-RN system $\mathcal{N}$ we have to simulate it by means of a g-RN component N. The simulation consists in considering N the union of every component in $\mathcal{N}$, so that a copy of type $N_i \in \mathcal{N}$ is represented by a copy of N, only marked in the places that correspond to N_i. Given a g-RN system $\mathcal{N} = \{N_1, \ldots, N_n\}$ with $N_i = (P_i, T_i, F_i, \lambda_i)$, let us consider the g-RN component $N = (P, T, F, \lambda)$, where:

- $P = \bigcup_{i=1}^{n} P_i$ and $T = \bigcup_{i=1}^{n} T_i$,

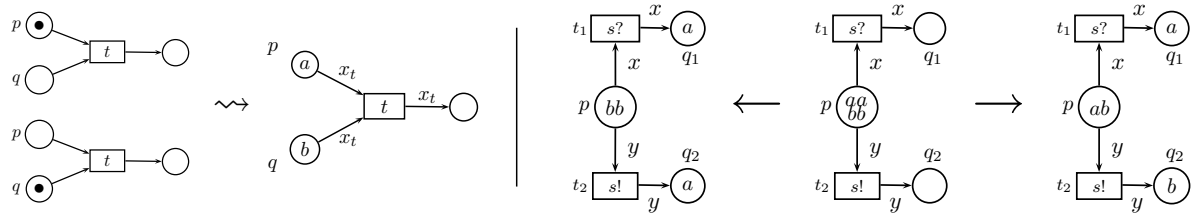

Figure 5. Not enabled transitions (left) and simulation of the RN system in Fig. 3 (right)

- $F(p,t) = \begin{cases} F_i(p,t) & \text{if } p \in P_i \text{ and } t \in T_i \text{ for some } i, \\ 0 & \text{otherwise.} \end{cases}$ and

  $F(t,p) = \begin{cases} F_i(t,p) & \text{if } p \in P_i \text{ and } t \in T_i \text{ for some } i, \\ 0 & \text{otherwise.} \end{cases}$

- $\lambda(t) = \lambda_i(t)$ for $t \in T_i$.

A marking M of any N_i can be viewed as a marking of N because $P_i \subseteq P$, so that a marking $\mathcal{M}$ of $\mathcal{N}$ is also a marking of N. Then, a mode of a transition t of some N_i is also a mode of t, seen as a transition of N. With these definitions it holds that $\mathcal{M} \stackrel{\overline{t}(\sigma)}{\rightarrow} \mathcal{M}'$ is a firing of $\mathcal{N}$ if and only if it is a firing of N. Moreover, since we are assuming that every transition has a precondition, it holds that for every M in every reachable marking $\mathcal{M}$ of N, M is the marking of some component N_i, that is, M cannot mark places both in P_i and P_j whenever $i \neq j$, so that every reachable marking in N is a marking of $\mathcal{N}$. Then, trivially, this simulation preserves reachability and coverability. □

This construction is considerably different from the analogous ones in the previous sections that flattened arbitrary systems to systems with a single component. The reason is that we cannot apply the reasoning we followed in the previous flattening results about combining tuples of compatible transitions, since in the current model these transitions may be fired by different components. In fact, it is possible to see that RN components cannot be flattened to RN components that cannot synchronize.[3]

Notice that the result of flattening a basic net system to a P/T net (following Prop. 3.1) is different from the result that would be obtained by following the previous construction (which could be done because basic net systems are a subclass of g-RN systems, those without replicating transitions). However, though they are syntactically different, we have proved that they are equivalent in the sense we are using along this paper, which preserves the properties we are studying.

6. Name creation vs. Replication

In this section we will prove the results relating name creation and replication, that essentially consist in the simulation of replication by means of name creation and vice versa. In the previous sections we have proved that every system (basic, ν-net or replicated) can be flattened to an equivalent one (regarding the properties we are interested in) with a single component, so that in the following results it is enough to consider the simplified versions of our models.

Proposition 6.1. Every g-RN system can be simulated by a ν-net system.

[3] We have recently proved that reachability is decidable for the class of g-RN components without synchronizations.

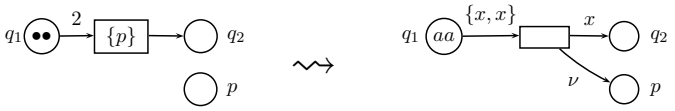

Figure 6. Simulation of replicating transitions

Proof:

Thanks to Prop. 5.2 it is enough to prove that given a g-RN component $N = (P, T, F, \lambda)$ there is a ν-net component that simulates it. The proof consists on simulating the behaviour of every copy of the component within the same ν-net, using a different identifier to simulate the ordinary black tokens in each of the copies, thus distinguishing between them. In particular, garbage collection of empty components is obtained for free by means of the disappearance of the identifier that represents it. We guarantee that different components do not incorrectly interact with each other by labelling every arc adjacent to the same transition t with the same variable x_t, so that they only manipulate tokens of the same type (representing one single net component). For instance, in the left of the Fig. 5 it is shown the simulation of an RN system with two components. Each of the components has an autonomous transition that is not enabled. This situation is simulated by a ν-net with two different identifiers. Since the arcs are labelled by the same variable, the transition is not enabled, because it cannot take the same name from its two preconditions. The right of the same figure shows the simulation of the g-RN system in Fig. 3 (taking $x = x_{t_1}$ and $y = x_{t_2}$). That system could fire in two modes, that in which a component synchronizes with itself and that in which both components synchronize with each other. Both steps are simulated by the synchronizing of the two transitions. In the first case, the same identifier is taken from both preconditions, while in the second case the identifiers taken are different.

Creation of net components is mapped to creation of new tokens that reflect the initial marking of the new component, as illustrated in Fig. 6.

Formally, for each $t \in T$ we take a different variable $x_t \in Var \setminus \{\nu\}$. Then we define $N^* = (P, T, F^*, \lambda)$ given by:

- $F^*(p,t) = \{\overbrace{x_t, \ldots, x_t}^{F(p,t)}\}$,
- $F^*(t,p) = \{\overbrace{x_t, \ldots, x_t}^{F(t,p)}\}$ if $\lambda(t) \in Sync$,
- $F^*(t,p) = \{\overbrace{x_t, \ldots, x_t}^{F(t,p)}, \overbrace{\nu, \ldots, \nu}^{\lambda(t)(p)}\}$ if $\lambda(t) \in \mathcal{MS}(P)$,

Moreover, for each marking $\mathcal{M}$ of N we define the marking $\mathcal{M}^*$ of N^* as follows. For each $M \in S(\mathcal{M})$ and $i \in \{1, \ldots, \mathcal{M}(M)\}$ we take a different identifier $\eta_{(M,i)}$. We will say that $\eta_{(M,i)}$ represents the i-th copy of M. Then, we take $\mathcal{M}^*(p)(\eta_{(M,i)}) = M(p)$ for each $M \in S(\mathcal{M})$. Notice that if we choose different identifiers to represent each of the component then we obtain markings of N^* that are α-equivalent. Finally, given a mode $\sigma : \overline{t} \to Markings(N) \times \mathbb{N}$ we consider $\sigma^* : Var(\overline{t}) \to Id$ such that $\sigma^*(x_t) = \eta_{\sigma(t)}$ and $\sigma^*(\nu)$ is fresh in $\mathcal{M}^*$ when $\nu \in Var(\overline{t})$. Let us first see that $\mathcal{M} \overset{\overline{t}(\sigma)}{\to} \mathcal{M}'$ if and only if $\mathcal{M}^* \overset{\overline{t}(\sigma^*)}{\to} \mathcal{M}'^*$:

In the first place, $\sigma^*(F^*(p,t))(\eta_{(M,i)}) = \sigma^*(\{\overbrace{x_t, \ldots, x_t}^{F(p,t)}\})(\eta_{(M,i)}) =$

$$= \{\overbrace{\eta_{\sigma(t)}, \ldots, \eta_{\sigma(t)}}^{F(p,t)}\}(\eta_{(M,i)}) = \begin{cases} F(p,t) & \text{if } \sigma(t) = (M,i) \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\sigma^*(F(p,\overline{t}))(\eta_{(M,i)}) = \sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} F(p,t) = \sum_{\substack{t \in \overline{t} \\ \sigma(t) = (M,i)}} {}^\bullet t(p)$ for all $M \in S(\mathcal{M})$ and $i \in \{1, \ldots, \mathcal{M}(M)\}$.

In the same way we can see that $\sigma^*(F(\overline{t},p))(\eta_{(M,i)}) = \sum_{\substack{t\in\overline{t}\\ \sigma(t)=(M,i)}} F(p,t) = \sum_{\substack{t\in\overline{t}\\ \sigma(t)=(M,i)}} t^\bullet(p)$ for every $M \in S(\mathcal{M})$ and $i \in \{1,\ldots,\mathcal{M}(M)\}$.

Therefore, $\overline{t}(\sigma)$ is enabled if and only if

$$\sum_{\substack{t\in\overline{t}\\ \sigma(t)=(M,i)}} {}^\bullet t \quad \subseteq M \text{ for every } M \in S(\mathcal{M}) \text{ and } i \in \{1,\ldots,\mathcal{M}(M)\} \Leftrightarrow$$

$$\Leftrightarrow \sum_{\substack{t\in\overline{t}\\ \sigma(t)=(M,i)}} {}^\bullet t(p) \leq M(p) \text{ for all } M \in S(\mathcal{M}),\ i \in \{1,\ldots,\mathcal{M}(M)\} \text{ and } p \in P \Leftrightarrow$$

$$\Leftrightarrow \sigma^*(F(p,\overline{t}))(\eta_{(M,i)}) \leq \mathcal{M}^*(p)(\eta_{(M,i)}) \text{ for all } M \in S(\mathcal{M}),\ i \in \{1,\ldots,\mathcal{M}(M)\} \text{ and } p \in P \Leftrightarrow$$

$$\Leftrightarrow \sigma^*(F(p,\overline{t})) \subseteq \mathcal{M}^*(p) \text{ for all } p \in P, \text{ which happens if and only if } \overline{t}(\sigma^*) \text{ is enabled for } \mathcal{M}^* \text{ in } N^*.$$

Now let us see that $\mathcal{M}'^*$ is the marking reached from $\mathcal{M}^*$ whenever $\mathcal{M}'$ is reached from $\mathcal{M}$. Let us distinguish the two possible cases for $\overline{t}$.

- If $\overline{t}$ is a tuple of compatible transitions, then we can take for each $M' \in S(\mathcal{M}')$ and $i \in \{1,\ldots,\mathcal{M}'(M')\}$ an identifier $\eta_{(M',i)}$ such that $\eta_{(M,i)} = \eta_{(M',i)}$ if M' is reached by M. In this case, for every $p \in P$ and $M' \in \mathcal{M}'$,

$$\mathcal{M}'^*(p)(\eta_{(M',i)}) = M'(p) = M(p) - \sum_{\substack{t\in\overline{t}\\ \sigma(t)=(M,i)}} {}^\bullet t(p) + \sum_{\substack{t\in\overline{t}\\ \sigma(t)=(M,i)}} t^\bullet(p) =$$

$$= \mathcal{M}^*(p)(\eta_{(M,i)}) - \sigma^*(F(p,\overline{t}))(\eta_{(M,i)}) + \sigma^*(F(\overline{t},p))(\eta_{(M,i)}).$$

  Then $\mathcal{M}'^*(p) = \mathcal{M}^*(p) - \sigma^*(F^*(p,\overline{t})) + \sigma^*(F^*(\overline{t},p))$.

- Now let us suppose that $\overline{t}$ is a replicating transition t, that is, $\mathcal{M}' = \mathcal{M} - \{M\} + \{M', \lambda(t)\}$ with $\sigma(t) = M$ and $M' = M - {}^\bullet t + t^\bullet$. In this case, for the definition of $\mathcal{M}'^*$ we can take identifiers such that $\eta_M = \eta_{M'}$[4] and $\eta_{\lambda(t)} = \sigma^*(\nu)$. Analogously as in the previous case, we have that

$$\mathcal{M}'^*(p)(\eta_{M'}) = M'(p) = M(p) - {}^\bullet t(p) + t^\bullet(p) = M(p) - F(p,t) + F(t,p) =$$
$$= \mathcal{M}^*(p)(\eta_M) - \sigma^*(F^*(p,t))(\eta_M) + \sigma^*(F^*(t,p))(\eta_M).$$

  Moreover, $\mathcal{M}'^*(p)(\eta_{\lambda(t)}) = \lambda(t)(p) = \sigma^*(F^*(t,p))(\eta_{\lambda(t)})$ and, therefore we can conclude that

  $\mathcal{M}'^*(p) = \mathcal{M}^*(p) - \sigma^*(F^*(p,t)) + \sigma^*(F^*(p,t))$.

To conclude we have to see that $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$ if and only if $\mathcal{M}_1^* \sqsubseteq_\alpha \mathcal{M}_2^*$. If $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$ then for each (M,i) there is $(M',j) = h(M,i)$ such that $M \subseteq M'$. If for every (M',j) such that $(M',j) = h(M,i)$ for some (M,i) we take $\eta_{(M',j)} = \eta_{(M,i)}$ then let us see that $\mathcal{M}_1^*(p) \subseteq \mathcal{M}_2^*(p)$ for every p. Indeed, $\mathcal{M}_1^*(p)(\eta_{(M,i)}) = M(p) \leq M'(p) = \mathcal{M}_2^*(p)(\eta_{(M',j)}) = \mathcal{M}_2^*(p)(\eta_{(M,i)})$. Then $\mathcal{M}_1^*(p) \subseteq \mathcal{M}_2^*(p)$ for every p and, therefore, $\mathcal{M}_1^* \sqsubseteq \mathcal{M}_2^*$, which implies $\mathcal{M}_1^* \sqsubseteq_\alpha \mathcal{M}_2^*$. Conversely, if $\mathcal{M}_1^* \sqsubseteq_\alpha \mathcal{M}_2^*$ then there is an injection ι such that $\mathcal{M}_1^*(p)(\eta_{(M,i)}) \leq \mathcal{M}_2^*(p)(\iota(\eta_{(M,i)}))$. Let us define $h(M,i) = (M',j)$ whenever $\iota(\eta_{(M,i)}) = \eta_{(M',j)}$, that is an injection because ι is an injection. Then, for all (M,i) such that $h(M,i) = (M',j)$, $M(p) = \mathcal{M}_1^*(p)(\eta_{(M,i)}) \leq \mathcal{M}_2^*(p)(\eta_{(M',j)}) = M'(p)$ so that $M \subseteq M'$ and the thesis follows. □

[4] To be precise, we should take $\eta_{(M,i)}$ for some i, but since only one component is involved we use this more compact notation.

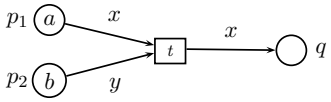

Figure 7. ν-net without name creation

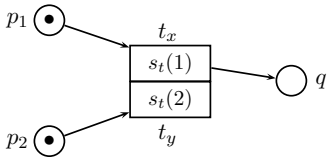

Figure 8. Simulation of the ν-net in Fig. 7 by means of g-RN systems ($a = b$)

Next we prove the converse simulation, namely that of pure names by means of replicated components. In the proof we will consider a total order defined in the set Var of variables. Therefore, for a set $\{x_1, \ldots, x_n\} \subseteq Var$ we will write $(x_1, \ldots, x_n)$ whenever $x_i \leq x_{i+1}$ for $i \in \{1, \ldots, n-1\}$. Moreover, for each transition t we will consider a different service label $s_t \in \mathcal{S}$ with arity $|Var(t)|$.

Proposition 6.2. Every ν-net system can be simulated by a g-RN system.

Proof:
Given a ν-net system N, we have to simulate it using a g-RN system N^*. Without loss of generality, thanks to Prop. 4.1 we can assume that N is in fact a ν^*-APN.

The key idea is to use a g-RN component for each different name, all of the same type. This component type has the same places as N, so that a black token in a place p corresponding to the component that is simulating the identifier a stands for a token a in the place p of the original net. Fig. 8 shows the simulation of the net in Fig. 7 when $a = b$. In this case, since in the marking there is a single identifier, we only need one component. Fig. 9 also simulates the net in Fig. 7, but now assuming $a \neq b$, so that we need two different components. The net in the left simulates the occurrence of a in Fig. 7, and the one in the right simulates the occurrence of b.

This is a straightforward way to represent the markings of N in N^*. However, the simulation of firings is quite more intricate. Given a transition of N, if every arc that is adjacent to it is labelled with the same variable, then the firing of that transition only involves one token name. If, for instance, that token is a then the g-RN component representing a can fire an autonomous transition that mimics that firing, moving tokens from the preconditions of t to its postconditions. However, if there is more than one variable adjacent to t then that is not possible. Consider the net in Fig. 7, with two different variables x and y next to its sole transition. It may still be the case that it is fired with x and y instantiated to the same value, so that we could simulate it by firing an autonomous transition. Instead, since we allow self-synchronization, we will simulate it with the synchronization of two transitions, as shown in Fig. 8. This simulation has the advantage that it also works when we fire t using two different tokens, say a and b, for x and y, respectively. In that case, to simulate the firing two net components, one representing a and one representing b, should interact by means of a pair of synchronizing transitions: The one representing a should remove a token from p_1 and put it in q (action $s_t(1)$ in Fig. 9), while the one representing b should just remove a token from its place p_2 (action $s_t(2)$ in Fig. 9).

Regarding name creation, we map it to component creation. Thus, every transition with outgoing arcs labelled by ν are simulated by replicating transitions. The initial marking of the new component is that with a token in every postcondition linked by a ν-arc (see Fig. 10). Notice that we have to remove in the

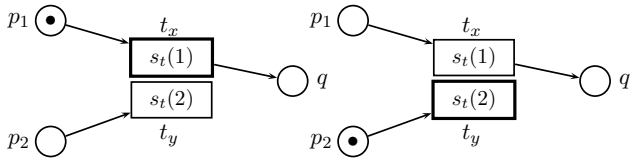

Figure 9. Simulation of the ν-net in Fig. 7 by means of g-RN systems ($a \neq b$)

simulation the arcs labelled by ν, since the replicating transition automatically marks those places when fired. Notice that this construction works because we are assuming that transitions that create names do not deal with more than two token names, since we are assuming that N is a ν^*-APN (otherwise, we would need replicating synchronizing transitions in g-RN components).

The previous construction can be generalized to transitions with an arbitrary number of variables. In the case of two variables, we have considered two transitions, one for each variable. In general, we have to consider a transition t_x for every variable $x \in Var(t) = (x_1, \ldots, x_n)$, and label it with $s_t(i)$ if x is the i-th variable in $Var(t)$. Thus, the firing of the compatible tuple $\overline{t} = (t_{x_1}, \ldots, t_{x_n})$ simulates a firing of t. Some components could fire several transitions in $\overline{t}$, simulating the fact that some variables could be instantiated to the same value.

Let us formalize the previous ideas. If $N = (P, T, F)$ we take the g-RN system N^*, that will have a single component, given by $N^* = (P, T^*, F^*, \lambda^*)$, where:

- $T^* = \{t_x \mid t \in T,\ \nu \neq x \in Var(t)\}$,
- $F^*(p, t_x) = F(p, t)(x)$ and $F^*(t_x, p) = F(t, p)(x)$,
- $\lambda^*(t_x) = \begin{cases} s_t(i) & \text{if } \nu \notin Var(t) = (x_1, \ldots, x_n) \text{ and } x = x_i \\ M_t^\nu & \text{otherwise, where } M_t^\nu(p) = F(t, p)(\nu). \end{cases}$

For a transition t, if $Var(t) = (x_1, \ldots, x_n)$ we will write $t^* = \begin{cases} (t_{x_1}, \ldots, t_{x_n}) & \text{if } \nu \notin Var(t), \\ t_x & \text{if } x, \nu \in Var(t). \end{cases}$

Given the marking M of N we define the marking of $\mathcal{N}^*$, $M^* = \{M_a \mid a \in S(M)\}$, where $M_a(p) = M(p)(a)$. Finally, for a mode $\sigma : Var(t) \to Id$ we define $\sigma^*(t_x)$ in the following way. We consider the partition $S(M) = A_1 \sqcup \ldots \sqcup A_k$ such that $M_a = M_b$ for every $a, b \in A_i$, for every $i \in \{1, \ldots, k\}$ and $M_a \neq M_b$ for every $a \in A_i$ and every $b \in A_j$, for all $i \neq j$. Let us consider an arbitrary order in each A_i, so that $A_i = \{a_i^1, \ldots, a_i^{k_i}\}$. Each A_i is composed of those identifiers that are represented by components with the same marking. If we denote $M^i = M_{a_i^j}$ for all $i \in \{1, \ldots, k\}$ (it does not matter which a_i^j we choose) then it holds that $M^*(M^i) = k_i$ and $S(M^*) = \{M^1, \ldots, M^k\}$. We can use these notations to define the mode σ^* for t^* induced by a mode σ for t. If $\sigma(x) = a_i^j$ we define $\sigma^*(t_x) = (M^i, j)$. Notice that thanks to Prop. 5.1 the ordering of the elements in each A_i considered is unsubstantial. Let us now see that $M \overset{t(\sigma)}{\to} M' \Leftrightarrow M^* \overset{t^*(\sigma^*)}{\longrightarrow} M'^*$.

In the first place, $\sigma(F(p,t))(a_i^j) = \sum_{\substack{x \in Var(t) \\ \sigma(x) = a_i^j}} F(p,t)(x) = \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} F^*(p, t_x) = \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x\,(p)$.

Analogously, $\sigma(F(t,p))(a_i^j) = \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} t_x^\bullet\,(p)$. Then,

$t(\sigma)$ is enabled in $M \Leftrightarrow$ for all $p \in P$, $\sigma(F(p,t)) \subseteq M(p) \Leftrightarrow$

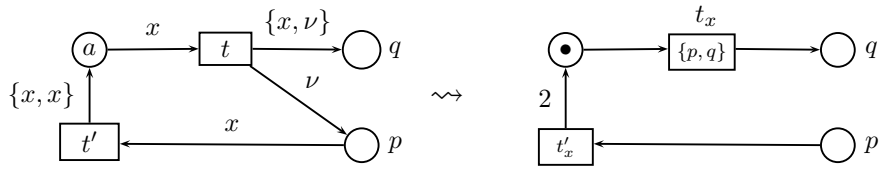

Figure 10. ν-net with name creation and its g-RN simulation

$\Leftrightarrow$ for all $p \in P$ and $a_i^j \in S(M)$, $\sigma(F(p,t))(a_i^j) \leq M(p)(a_i^j) \Leftrightarrow$

$\Leftrightarrow$ for all $p \in P$ and $a_i^j \in S(M)$, $\sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x\,(p) \leq M^i(p) \Leftrightarrow$

$\Leftrightarrow$ for all (M^i, j), $\sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x \subseteq M^i \Leftrightarrow t^*(\sigma^*)$ is enabled in M'^*.

Notice that this is true either when $\nu \in Var(t)$ or when $\nu \notin Var(t)$. If both $t(\sigma)$ and $t^*(\sigma^*)$ are enabled, reaching M' and $\mathcal{M}'$ respectively, let us see that $M'^* = \mathcal{M}'$.

On the one hand, $M'(p) = M(p) - \sigma(F(p,t)) + \sigma(F(t,p))$ and $M'^* = \{M'_a \mid a \in S(M')\}$ with $M'_a(p) = M'(p)(a)$. On the other hand, by definition of M^* we have that

$$\mathcal{M}' = \{M_a - \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x \quad + \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} t_x^\bullet \quad \mid a \in S(M)\} + \begin{cases} \{\lambda^*(t)\} & \text{if } \lambda^*(t) \in \mathcal{MS}(P), \\ \emptyset & \text{otherwise.} \end{cases}$$

Let us distinguish the possible cases for $a \in S(M) \cup S(M')$:

- If $a \in S(M') \cap S(M)$ then $a = a_i^j$ for some i and j. In this case, $M'_a(p) = M'(p)(a) = M(p)(a) - \sigma(F(p,t))(a) + \sigma(F(t,p))(a) = M_a(p) - \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x\,(p) + \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} t_x^\bullet\,(p)$ for every p and, therefore, $M'_a = M_a - \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x \quad + \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} t_x^\bullet$. This implies, by definition of $\mathcal{M}'$, that $M'_a \in \mathcal{M}'$ for every $a \in S(M) \cap S(M')$.

- If $a \in S(M') \setminus S(M)$ then $a = \sigma(\nu)$. In this case, $\lambda^*(t) \in \mathcal{MS}(P)$ and $M'_a(p) = F(t,p)(\nu) = \lambda^*(t)(p) \Rightarrow M'_a = \lambda^*(t) \in \mathcal{M}'$.

- To conclude, let us consider $a = a_i^j \in S(M) \setminus S(M')$. Then, a is an identifier that has disappeared due to the firing of $t(\sigma)$, which implies $\sigma(F(p,t))(a_i^j) = M(p)(a_i^j)$ and $\sigma(F(t,p))(a_i^j) = 0$. Then, $\sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x = M_{a_i^j}$ and $\sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} t_x^\bullet = \emptyset$, so that $M_{a_i^j} - \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} {}^\bullet t_x \quad + \sum_{\substack{t_x \in t^* \\ \sigma^*(t_x) = (M^i, j)}} t_x^\bullet = \emptyset$.

Summing up, $\mathcal{M}' = M'^* + \{\emptyset \mid a \in S(M) \setminus S(M')\} \equiv M'^*$, as we wanted to prove.

To conclude our proof, let us see that $M \sqsubseteq_\alpha M' \Leftrightarrow M^* \sqsubseteq M'^*$. If $\iota : S(M) \to S(M')$ is an injection then $h(M_a) = M'_{\iota(a)}$ is also an injection. Conversely, given h we can define the corresponding ι. Then, $M \sqsubseteq_\alpha M' \Leftrightarrow$ there is some ι such that for all a, p, $M(p)(a) \leq M'(p)(\iota(a)) \Leftrightarrow$ there is some ι such that for all a, p, $M_a(p) \leq M'_{\iota(a)}(p) \Leftrightarrow$ there is some ι such that for all a, $M_a \subseteq M'_{\iota(a)} \Leftrightarrow$ there is some h such that for all a, $M_a \subseteq h(M_a) \Leftrightarrow M^* \sqsubseteq M'^*$. □

Corollary 6.1. Reachability is undecidable and coverability is decidable for g-RN systems.

Proof:
Prop. 6.1 gives us the decidability of coverability, and Prop. 6.2 implies undecidability of reachability. □

7. Name creation and Replication

Now we consider a model in which we combine the two features in the two previous sections, that is, systems composed of nets that can both create fresh identifiers and replicate themselves. Formally, ν-RN systems are ν-net systems for which we allow an extra transition label type, so that we can label transitions with markings of components, as in the previous section, but now these markings are composed of named tokens. More precisely, a marking is a multiset of markings of its components, where a marking of a component $N = (P, T, F, \lambda)$ is a function $M : P \to \mathcal{MS}(Id)$. Therefore, a mode for a tuple of transitions has to specify both which are the components firing the transitions and what are the tokens taken from their preconditions.

Quite surprisingly, though the expressive powers obtained by adding either name creation or only replication are identical, as we have proved in the previous section, it turns out that they are somehow orthogonal, so that when combined we reach Turing-completeness, as we will prove next. Informally, we could say that both features generate bidimensional infinite state systems. In the case of ν-net components we may have an unbounded number of different tokens, each of which can appear in markings an unbounded number of times. Analogously, in the case of g-RN systems, we may have an unbounded number of components, each of them with a possibly unbounded number of tokens. However, these two dimensional spaces "are not the same", although somehow equivalent to each other, according to Prop. 6.1 and Prop. 6.2. But when we merge the four sources of infinity, only the common dimension overlaps, or following the geometric metaphor, we have two perpendicular planes, that no longer generate a bidimensional space, but the whole tridimensional space. We will see that this space is too rich, thus producing a Turing-complete formalism.

Proposition 7.1. Any Turing machine can be simulated by a ν-RN system.

Proof:
The proof is reminiscent of the simulation of Turing machines with unbounded security protocols in [12]. Given a Turing machine, we have to simulate it using a ν-RN system. The main problem to simulate Turing machines is the representation of the potentially one-way infinite set of cells of the tape. We represent the tape by means of a doubly-linked list. Each node of the list (or cell of the tape) will be represented by a different copy of the same process, depicted in Fig. 11. This process has a memory with two kinds of data: First, it must know whether its value is 0 or 1. For that purpose, it has two places named 0 and 1, that are marked in mutual exclusion. Second, and more important, the cell must hold information about which are the previous and next cell, whenever it becomes part of the tape of the machine, for what we will use identifiers. For that purpose, components that represent cells have three places (among others), me, $prev$ and $next$, that contain three different identifiers: the name of the cell in me, that of the previous cell in $prev$ and that of the next cell in $next$.

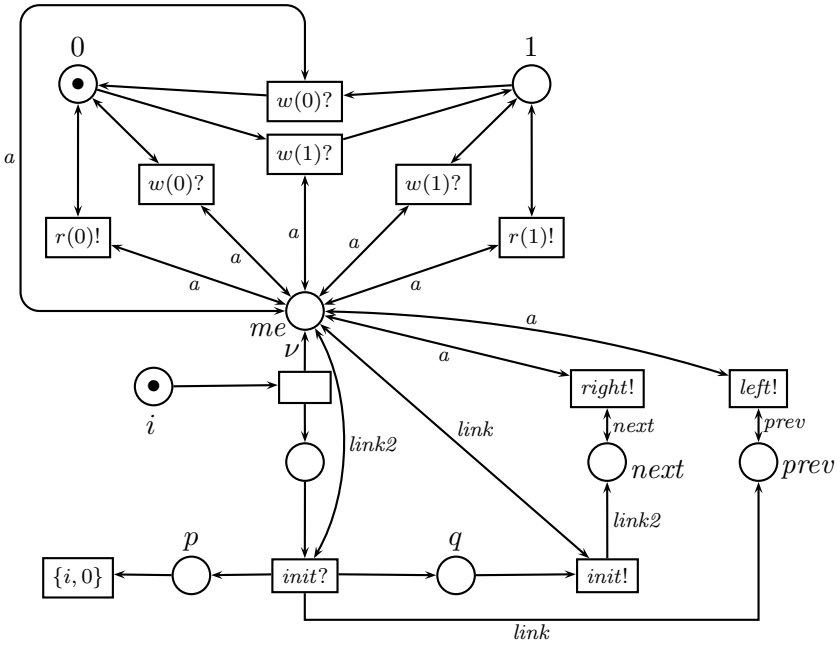

Figure 11. Turing cell

At any time, the tape has a final cell, that knows its own name and that of its previous cell (it has been initialized firing its $init?$ transition), but not that of the next cell, which in fact does not still exist. In order to expand the tape a new cell can be created by firing the replicating transition, labelled by $\{i, 0\}$. This cell can generate its own name, after which it is willing to synchronize with its parent net (the cell at the end of the tape). Then, the synchronization of the transitions $init!$ and $init?$ causes the exchange of the respective names of the cells, putting them in the corresponding $next$ and $prev$ places, so that the net in the tape now knows the name of its next cell, and the new cell becomes the last of the tape.

The cells should also have synchronizing transitions that output the name of the previous and the next cell (transitions $left!$ and $right!$, respectively), two others that output its current value (transitions $r(0)!$ and $r(1)!$) and finally four more that change it as indicated by the program of the machine (those labelled by $w(0)?$ and $w(1)?$). All these transitions are doubly-linked with the place me, by arrows labelled by the variable a.

The part of the construction described so far is the same for any Turing Machine. The rest of the simulation only needs an additional net component for the control, with a place now containing the identifier of the cell where the head is pointing. All the synchronizations with the program mentioned in the previous paragraph are done forcing the matching between the value in the place now of the program and the value in the place me of the cell, by means of the variable a, so that the head can only read and modify the proper cell. Of course, it should also have one place per state of the machine, marked in mutual exclusion, each of them with the corresponding actions attached. More precisely, and following the standard notations for Turing machines, if $(p, \eta_1, q, \eta_2, D) \in \delta$ is a transition of the machine, and for instance $D = \rightarrow$, we simulate that transition by means of the net in Fig. 12. As an example, Fig. 13 shows the simulation of the head of a Turing machine that computes the infinite sequence $0101\ldots$[5]

Finally, let us see what is the initial marking of the constructed ν-RN system. If we consider Turing

[5]The shown component is actually a simplification of the one that would be obtained by literally following the previous ideas, considering that, in this example, symbols written on the tape do not depend on the ones read.

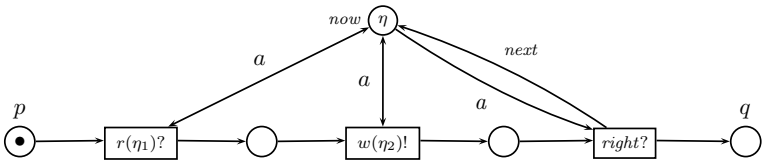

Figure 12. Simulation of $(p, \eta_1, q, \eta_2, \rightarrow) \in \delta$

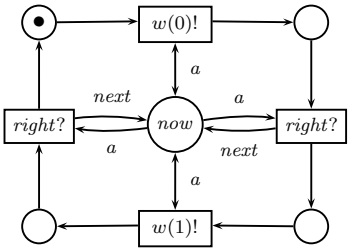

Figure 13. Turing machine that computes $0101\ldots$

machines as language generators (without any input) then it is enough to consider a single cell component marked by an ordinary token in $\{0, p, q\}$ and an arbitrary identifier η in place me, together with the control component marked by an ordinary token in the place that represents the intial state of the machine and the identifier η in its place now (meaning that the head is pointing at the first cell of the tape).

□

Notice that the simulation in the proof of the previous result does not make any use of garbage collection, so that Turing completeness is achieved even without it. In fact, we are implicitly considering a kind of garbage collection for identifiers, in the sense that they could disappear from markings after the firing of some transition (as happens for instance to identifier b in Fig. 7 after firing t), thus becoming reusable. We conjecture that if we avoided this automatic garbage collection for names, by including the full set of created names to the state, we would get a situation analogous to that in [11], so that reachability is also decidable. However, for our ν-RNs we can prove the following.

Corollary 7.1. Coverability and reachability are undecidable for ν-RN systems.

Proof:
It is clear that reachability remains undecidable since it also was undecidable in the less general model of ν-net systems. Coverability is undecidable for ν-RN systems, or we could use the previous simulation to decide the halting problem in Turing machines, just by asking whether the marking with a token in the place representing the final state can be covered. □

It is worth pointing out that though the natural orders for both ν-net systems and g-RN systems are wqos, as a consequence of the previous undecidability result, the natural order for ν-RN systems, that extends both, cannot be a wqo.

Definition 7.1. Given two markings $\mathcal{M}_1 = \{M_1, \ldots, M_n\}$ and $\mathcal{M}_2 = \{M'_1, \ldots, M'_m\}$ of a ν-RN system $\mathcal{N}$, we write $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$ if there are two injections $h : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$ and $\iota : Id \rightarrow Id$ such that for every $i \in \{1, \ldots, n\}$, $\iota(M_i) \sqsubseteq M'_{h(i)}$.

$\mathcal{M}^1$	$\mathcal{M}^2$	$\mathcal{M}^3$
1	1	1
1 2	2 2	2 2
	2 3 3	3 3 3
		3 4 4 4

Figure 14. The natural order of ν-RNs is not a wqo

Therefore, h has the role of mapping components of $\mathcal{M}_1$ to components of $\mathcal{M}_2$, while ι has the role of mapping identifiers in $\mathcal{M}_1$ to identifiers in $\mathcal{M}_2$. The key point is that ι must be the same for all components; otherwise, we would obtain the multiset order induced by $\sqsubseteq_\alpha$, which is a wqo, but each component would have its own name space, and they would not be allowed to interact.

To see that the defined order is not a wqo, let us consider the simple case of systems with a single net type, with just one place, so that every marking of such a system consists on a multiset of multisets of identifiers. Let us take $Id = \mathbb{N}$ and consider the sequence of markings depicted in Fig. 14, $\mathcal{M}^i = \{M^i_1, \ldots, M^i_i, M^i_{i+1}\}$, for $i = 1, 2, \ldots$, where $M^i_k = \{\overbrace{k, \ldots, k}^{i}\}$ for $k = 1, \ldots, i$ and $M^i_{i+1} = \{i, \overbrace{i+1, \ldots, i+1}^{i}\}$, for which there are no indices $i < j$ such that $\mathcal{M}^i \sqsubseteq \mathcal{M}^j$. This is so because for any $i < j$, due to the cardinals of each M^k_i and M^k_j, the only possible choice is to take $h(k) = k$ for $k = 1, \ldots, i$, and ι as the identity function, which does not work because $i \in M^i_{i+1}$, but $i \notin M^j_k$ for all $k \geq i + 1$.

8. Conclusions and Future Work

We have investigated the consequences of adding two different primitives to Petri Net Systems. The first one is a pure name creation primitive. The model that results from adding it was already studied in [16] and [24], where we proved that its expressive power lies in between that of ordinary Petri Nets and Turing machines. The second primitive, which was presented in [26], deals with component replication. We have proved that these two extensions can simulate each other and thus have the same power. However, when we simultaneously extend the basic model in the two directions, by incorporating both the name creation and the replication primitive, the obtained model turns out to be Turing complete.

There are several directions in which we plan to continue our research. First, we would like to know the relation between our g-RN systems (or equivalently, our ν-net systems) and some well established Petri net formalisms, whose expressive power is also in between Turing machines and Petri nets, such as Petri nets with transfer or reset arcs. We already know that both can be weakly (that is, preserving reachability) simulated using ν-nets, but we do not know if a more faithful simulation that preserves the full behaviour of the net is also possible.

Another interesting issue would be finding some restrictions to the use of the ν variable or that of replicating transitions, so that a model with constrained features, but including both primitives, would no longer be Turing complete. For instance, it is clear that this is the case if the number of times we can use the replication transitions is globally bounded, or equivalently, whenever we can only create a finite number of names. Certainly, the simulation of Turing machines we have presented would not work if only a bounded quantity of different names could appear at any reachable marking, even if an arbitrary number of them can be created, which suggests that coverability remains decidable in that case.

After Def. 7.1 we have suggested another restriction in which coverability would remain decidable, by avoiding arbitrary synchronizations between the different components.

It would also be desirable to introduce the models presented in this paper and others appearing in [23] in the hierarchy presented in [1], that include lossy channel systems [9], constrained multiset rewriting [10, 2] or data nets [17]. Since all these models are well structured (and therefore have decidable coverability) we need a finer mechanism in order to distinguish between them. In [1] this is done by considering the class of languages generated by each of the models, so that we are now studying what is the class of languages generated by ν-APNs (and equivalently, g-RN systems) and what is its relation to the other classes.

We are also interested in the practical implementation of the results presented in this paper. In [25] we presented a prototype of a tool for the verification of MSPN systems, which mainly corresponds to our ν-nets here. We plan to extend it to include the replication operator and coverability and reachability algorithms, covering the cases in which these properties are decidable.

References

[1] P.A. Abdulla, G. Delzanno, and L. Van Begin. *Comparing the Expressive Power of Well-Structured Transition Systems.* 21st Int. Workshop on Computer Science Logic, CSL'07, LNCS vol. 4646, pp. 99-114. Springer, 2007.

[2] P.A. Abdulla, and G. Delzanno. *On the coverability problem for constrained multiset rewriting.* In: AVIS 2006, an ETAPS Workshop, 2006.

[3] T. Ball, S. Chaki, and S.K. Rajamani. *Parameterized Verification of Multithreaded Software Libraries.* 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'01. LNCS vol. 2031, pp. 158–173. Springer, 2001.

[4] A. Bouajjani, J. Esparza, and T. Touili. *A generic approach to the static analysis of concurrent programs with procedures.* 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'03. ACM SIGPLAN 38(1):62-73. ACM, 2003.

[5] A. Bouajjani, J. Esparza, and T. Touili. *Reachability Analysis of Synchronized PA Systems.* 6th Int. Workshop on Verification of Infinite-State Systems, INFINITY'04. ENTCS vol. 138(2), pp. 153-178. Elsevier, 2005.

[6] N. Busi, and G. Zavattaro. *Deciding Reachability in Mobile Ambients.* 14th European Symposium on Programming Languages and Systems, ETAPS'05. LNCS vol. 3444, pp. 248-262. Springer, 2005.

[7] L. Cardelli, and A.D. Gordon. *Mobile Ambients.* 1st Int. Conf. on Foundations of Software Sciences and Computation Structures, FOSSACS'98. LNCS vol. 1387, pp. 140-155. Springer, 1998.

[8] L. Cardelli, G. Ghelli, and A.D. Gordon. *Types for the Ambient Calculus.* Information and Computation 177(2): 160-194 (2002).

[9] G. Cécé, A. Finkel, and S.P. Iyer. *Unreliable channels are easier to verify than perfect channels.* Information and Computation 124(1):20-31, 1996.

[10] G. Delzanno. *An overview of MSR(C): A CLP-based Framework for the Symbolic Verification of Parameterized Concurrent Systems.* 11th Int. Workshop on Functional and Logic Programming, WFLP'02. ENTCS vol. 76. Elsevier, 2002.

[11] G. Delzanno. *Constraint-based Automatic Verification of Abstract Models of Multitreaded Programs.* To appear in the Journal of Theory and Practice of Logic Programming, 2006.

[12] N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. *Undecidability of bounded security protocols.* Proc. Workshop on Formal Methods and Security Protocols (FMSP'99).

[13] A.Finkel, and P.Schnoebelen. *Well-Structured Transition Systems Everywhere!* Theoretical Computer Science 256(1-2):63-92 (2001).

[14] D. Frutos-Escrig, O. Marroquín-Alonso and F. Rosa-Velardo. *Ubiquitous Systems and Petri Nets*. Ubiquitous Web Systems and Intelligence, LNCS vol.3841. Springer, 2005.

[15] A. Gordon. *Notes on Nominal Calculi for Security and Mobility.* Foundations of Security Analysis and Design, FOSAD'00. LNCS vol. 2171, pp. 262-330. Springer, 2001.

[16] O. Kummer. *Undecidability in object-oriented Petri nets.* Petri Net Newsletter, 59:18-23, 2000.

[17] R. Lazic, T.C. Newcomb, J. Ouaknine, A.W. Roscoe and J. Worrell. *Nets with Tokens Which Carry Data*. 28th Int. Conf. on Applications and Theory of Petri Nets and other models of concurrency, ICATPN'07, LNCS vol. 4546, pp. 301-320. Springer, 2007.

[18] R. Milner, J. Parrow, and D. Walker. *A Calculus of Mobile Processes, I.* Information and Computation 100(1): 1-40 (1992).

[19] R.M. Needham. *Names.* Distributed systems, ed. S. Mullender, 2nd ed., Reading, MA: Addison-Wesley, 1993, 315–326 and 531– 541.

[20] F. Nielson, R.R. Hansen, and H.R. Nielson. *Abstract interpretation of mobile ambients.* Sci. Comput. Program. 47(2-3): 145-175 (2003).

[21] G. Ramalingam. *Context-sensitive synchronization-sensitive analysis is undecidable.* ACM Trans. Program. Lang. Syst. 22(2): 416-430 (2000).

[22] M. Rinard. *Analysis of multithreaded programs.* 8th Static Analysis Symposium, SAS'01. LNCS 2126, pp. 1–19. Springer, 2001.

[23] F. Rosa-Velardo, D. Frutos-Escrig, and O. Marroquín-Alonso. *Mobile Synchronizing Petri Nets: a choreographic approach for coordination in Ubiquitous Systems.* 1st Int. Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems, MTCoord'05. ENTCS vol.150(1). Elsevier, 2006.

[24] F. Rosa-Velardo, D. Frutos-Escrig, and O. Marroquín-Alonso. *On the expressiveness of Mobile Synchronizing Petri Nets.* 3rd Int. Workshop on Security Issues in Concurrency, SecCo'05. ENTCS vol. 180(1), pp. 77-94. Elsevier, 2007.

[25] F. Rosa-Velardo. *Coding Mobile Synchronizing Petri Nets into Rewriting Logic*. 7th Int. Workshop on Rule-based Programming, RULE'06. ENTCS vol. 174(1), pp. 83-98. Elsevier 2007.

[26] F. Rosa-Velardo, D. Frutos-Escrig, O. Marroquín-Alonso. *Replicated Ubiquitous Nets*. Ubiquitous Web Systems and Intelligence, LNCS vol. 3983. Springer, 2006.

[27] F. Rosa-Velardo, D. Frutos-Escrig. *Name creation vs. Replication in Petri Net Systems.* 28th int. Conf. on Applications and Theory of Petri Nets and other models of concurrency, ICATPN'07, LNCS vol. 4546, pp. 402-422. Springer, 2007.

[28] P. Zimmer. *On the Expressiveness of Pure Mobile Ambients.* 7th Int. Workshop on Expressiveness in Concurrency, EXPRESS'00, ENTCS vol. 39(1). Elsevier, 2003.

Appendix

Lemma A. Every ν-net system can be simulated by a ν-APN.

Proof:
We will proceed analogously as we did in the proof of Prop. 3.1. If $\mathcal{N}$ is the ν-net system we want to flatten, with set of places P, set of transitions T, arcs defined by F and transitions labelled by λ, then we consider $\mathcal{N}^* = (P, T^*, F^*)$. Notice that we disregard λ^*.

- $T^* = \{\overline{t} = (t_1, \ldots, t_n) \mid t_i \in T,\ \overline{t} \text{ compatible}\}$,
- $F^*(p, \overline{t}) = \sum_{t \in \overline{t}} F(p, t)$ and $F^*(\overline{t}, p) = \sum_{t \in \overline{t}} F(t, p)$.

Again, every marking of $\mathcal{N}$ represents also a marking of $\mathcal{N}^*$ and a mode of a tuple $\overline{t}$ of compatible transitions of $\mathcal{N}$ is also a mode of the transition $\overline{t}$ of $\mathcal{N}^*$. Then we can see that $M \xrightarrow{\overline{t}} M'$ is a firing of $\mathcal{N}$ if and only if it is a firing of $\mathcal{N}^*$. Using the notations we have seen before, $F^*(p, \overline{t}) = F(p, \overline{t})$ and $F^*(\overline{t}, p) = F(\overline{t}, p)$. The order is trivially preserved, and therefore the thesis follows straightforward. □

Lemma B. Every ν-APN can be simulated by a ν^*-APN

Proof:
Given the ν-APN $N = (P, T, F)$ we define the ν^*-APN $N^* = (P^*, T^*, F^*)$ following the standard procedure of adding control points to the net. This construction reduces the concurrency of the simulating net, though it simplifies the proof significantly. This simulation is not as the simulations defined in Sect. 2, but it preserves the properties we are interested in. The idea is to split every $t \in T$ in two different transitions t_1 and t_2, as illustrated in the example of Fig. 15. The transition t_1 represents the part of t that does not involve ν, while t_2 is the part of t that only involves ν. We add the mentioned control points to guarantee that once t_1 is fired, the only possible transition to be fired next is t_2. N^* is defined as follows:

- $P^* = P \cup \{red_t \mid t \in T\} \cup \{ok\}$,
- $T^* = \{t_1, t_2 \mid t \in T\}$,
- $F^* : (P^* \times T^*) \cup (T^* \times P^*) \to \mathcal{MS}(Var)$ is defined as follows:
  - If $p \in P$, $F^*(p, t_1) = F(p, t)$, $F^*(t_2, p)(x) = \begin{cases} F(t,p)(\nu) & \text{if } x = \nu \\ 0 & \text{otherwise,} \end{cases}$ and $F^*(t_1, p) = F(t, p) - F^*(t_2, p)$,
  - $F^*(t_1, red_t) = F^*(ok, t_1) = F^*(red_t, t_2) = F^*(t_2, ok) = \{\varepsilon\}$,
  - In any other case, $F^*(t, p) = \emptyset$ and $F^*(p, t) = \emptyset$.

Notice that N^* is a ν^*-APN, because for every $t \in T$, $\nu \notin Var(t_1)$ and $Var(t_2) = \{\nu, \varepsilon\}$. For a marking M of N we take $M^*(p) = \begin{cases} M(p) & \text{if } p \in P, \\ \{\bullet\} & \text{if } p = ok, \\ \emptyset & \text{otherwise.} \end{cases}$

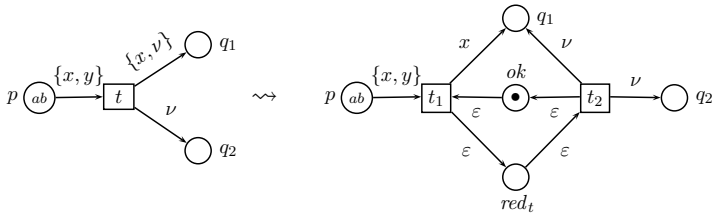

Figure 15. ν-APN to ν^*-APN

Finally, for a mode σ we consider $\sigma_1(x) = \begin{cases} \sigma(x) & \text{if } x \neq \varepsilon, \\ \bullet & \text{if } x = \varepsilon. \end{cases}$ and $\sigma_2(x) = \begin{cases} \sigma(\nu) & \text{if } x = \nu, \\ \bullet & \text{if } x = \varepsilon. \end{cases}$

With these notations, if $M \stackrel{t(\sigma)}{\rightarrow} M'$ then there is $\overline{M}$ such that $M^* \stackrel{t_1(\sigma_1)}{\longrightarrow} \overline{M} \stackrel{t_2(\sigma_2)}{\longrightarrow} M'^*$. Indeed, if $p \in P$ then $\sigma_1(F^*(p,t_1)) = \sigma_1(F(p,t)) = \sigma(F(p,t))$, this last equality holds because $\varepsilon \notin F(p,t)$. Since $t(\sigma)$ is enabled, $\sigma(F(p,t)) \in M(p) = M^*(p)$. Moreover, $\sigma_1(F^*(ok,t_1)) = \sigma_1(\{\varepsilon\}) = \{\bullet\} = M^*(p)$. Finally, $\sigma_1(F^*(red_t,t_1)) = \emptyset = M^*(red_t)$. Therefore, for all $p \in P^*$, $\sigma_1(F^*(p,t_1)) \subseteq M^*(p)$ and thus $t_1(\sigma_1)$ is enabled in M^*.

Let $\overline{M}(p) = M^*(p) - \sigma_1(F^*(p,t_1)) + \sigma_1(F^*(t_1,p))$. In order to see that $t_2(\sigma_2)$ is enabled in $\overline{M}$, since red_t is the only precondition of t_2, it is enough to see that $\sigma_2(F^*(red_t,t_2)) \subseteq \overline{M}(red_t)$. Indeed, $\overline{M}(red_t) = M^*(red_t) - \sigma_1(F^*(red_t,t_1)) + \sigma_1(F^*(t_1,red_t)) = \emptyset - \emptyset + \{\bullet\} = \{\bullet\} = \sigma_2(\{\varepsilon\}) = \sigma_2(F^*(red_t,t_2))$. Let us see that the reached state $\overline{M}'$ is M'^*.

By definition of $\overline{M}'$, $\overline{M}'(p) = \overline{M}(p) - \sigma_2(F^*(p,t_2)) + \sigma_2(F^*(t_2,p)) = M^*(p) - \sigma_1(F^*(p,t_1)) + \sigma_1(F^*(t_1,p)) - \sigma_2(F^*(p,t_2)) + \sigma_2(F^*(p,t_2))$.

If $p \in P$ then $\sigma_1(F^*(p,t_1)) = \sigma(F(p,t))$, $\sigma_2(F^*(p,t_2)) = \emptyset$ and $\sigma_1(F^*(t_1,p)) + \sigma_2(F^*(t_2,p)) = \sigma(F(t,p))$. Then, for all $p \in P$, $\overline{M}'(p) = M(p) - \sigma(F(p,t)) + \sigma(F(t,p)) = M'(p) = M'^*(p)$. Finally, $\overline{M}'(ok) = \{\bullet\}$ and $\overline{M}'(red_t) = \emptyset$, so that we can conclude that $\overline{M}' = M'^*$.

Analogously, we can see that if $M^* \stackrel{u}{\rightarrow} \overline{M}$ then there are t and σ such that $u = t_1(\sigma_1)$ and that, from $\overline{M}$, the only possible firing is that of $t_2(\sigma_2)$, reaching a marking of the form M'^* such that $M \stackrel{t(\sigma)}{\rightarrow} M'$.

Therefore, the simulation preserves reachability. Let us see that it also preserves coverability. Since M^* coincides with M in their common domain, it is clear that $M_1 \sqsubseteq_\alpha M_2$ if and only if $M_1^* \sqsubseteq_\alpha M_2^*$. Let us then see that M can be covered from M_0 if and only if M^* can be covered from M_0^*. If M can be covered then there is a reachable M' from M_0 such that $M \sqsubseteq_\alpha M'$. Then M'^* is reachable from M_0^*. Due to the previous remark, $M^* \sqsubseteq_\alpha M'^*$ and M^* can be covered from M_0^*. Conversely, if M^* can be covered then there is some $\overline{M}$ reachable from M_0^* such that $M^* \sqsubseteq_\alpha \overline{M}$. Since $M^*(ok) = \{\bullet\}$, by definition of N^* we have that $\overline{M}(ok) = \{\bullet\}$ and, consequently, $\overline{M}(red_t) = \emptyset$ for all $t \in T$. This implies that there is some M' such that $\overline{M} = M'^*$. Then it holds that M' is reachable from M_0 and $M \sqsubseteq_\alpha M'$, so that M can be covered from M_0 and the thesis follows.

□

Proposition 4.1. Every ν-net system can be simulated by a ν^*-APN.

Proof:
It is a corollary of Lemma A and Lemma B. □

Corollary 4.1. Reachability is undecidable, but coverability is decidable, for ν-net systems.

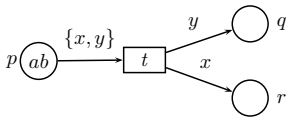

Figure 16. ν-APN with multisets in arcs

Proof:
Thanks to Prop. 4.1 we have to prove that the results hold for ν^*-APNs. Since reachability is undecidable and coverability is decidable for minimal OO-nets [24, 16], it is enough to simulate ν^*-APNs using minimal OO-nets, by means of a simulation that preserves reachability and coverability. Minimal OO-nets are ν-APNs with arcs labelled either by the empty multiset or by a singleton.

We simulate the effect of firing a transition t that can take several tokens from preconditions and add several tokens in postconditions, by means of a sequence of firings that take those tokens from preconditions, one by one, and then add them in preconditions, also one by one. Fig. 17 shows the simulation of the net in Fig. 16, where the firing of t is simulated by the firing of the sequence $t_p(x), t_p(y), t, t^r(x), t^q(y)$.

When taking tokens from preconditions one by one, it could be the case that those tokens do not enable the firing of the transitions, so that we would need to undo some steps. For that purpose we add transitions $\overline{t}_p(x)$. Notice that this construction introduces livelocks and reduces concurrency, although this is not important for our goal here.

For a variable t we denote by $\overline{pre}(t) = \sum_{p \in P} F(p,t)$ and analogously for $\overline{post}(t)$. Formally, for a ν-APN $N = (P, T, F)$ we take $N^* = (P^*, T^*, F^*)$, where

- $P^* = P \cup \{ok\} \cup \{p_t(x) \mid x \in F(p,t)\} \cup \{c_i(t), c'_i(t) \mid 1 \leq i \leq |\overline{pre}(t)|\} \cup$
  $\{p^t(x) \mid x \in F(t,p)\} \cup \{d_i(t) \mid 1 \leq i \leq |\overline{post}(t)| - 1\}$,
- $T^* = T \cup \{t_p(x), \overline{t}_p(x) \mid F(p,t) = x\} \cup$
  $\{t^p(x) \mid t \in T,\ F(t,p) = x\} \cup \{t_i \mid 1 \leq i \leq |\overline{pre}(t)|\}$,
- $F^*(p, t_p(x)) = F^*(t_p(x), p_t(x)) = x$ y $F^*(t^p(x), p) = F^*(p^t(x), t^p(x)) = x$,
- $F^*(p_t(x), t) = x$ y $F^*(t, p^t(x)) = x$,
- For each $t \in T$, if $\overline{pre}(t) = (x_1, \ldots, x_n)$ and $\overline{post}(t) = (y_1, \ldots, y_m)$ we take
  - $F^*(ok, t_p(x_1)) = \varepsilon$, $F^*(c_n(t), t^p(y_1)) = \varepsilon$, $F^*(t^p(y_m), ok) = \varepsilon$,
  - $F^*(c_i(t), t_p(x_{i+1})) = \varepsilon$ for $i = 1, \ldots, n-1$,
  - $F^*(t_p(x_i), c_i(t)) = \varepsilon$ for $i = 1, \ldots, n$,
  - $F^*(d_i(t), t^p(y_{i+1})) = \varepsilon$ for $i = 1, \ldots, m-1$,
  - $F^*(t^p(y_i), d_i(t)) = \varepsilon$ for $i = 1, \ldots, m-1$,
  - $F^*(c_i(t), t_i) = F^*(t_i, c'_i(t)) = \varepsilon$,
  - $F^*(c'_i(t), \overline{t}_p(x_i)) = \varepsilon$ for $i = 1, \ldots, n$,
  - $F^*(\overline{t}_p(x_i), c'_{i-1}(t)) = \varepsilon$ for $i = 2, \ldots, n$,
  - $F^*(\overline{t}_p(x_1), ok) = \varepsilon$,
  - $F^*(p_t(x_i), \overline{t}_p(x_i)) = F^*(\overline{t}_p(x_i), p) = x_i$ for $i = 1, \ldots, n$.

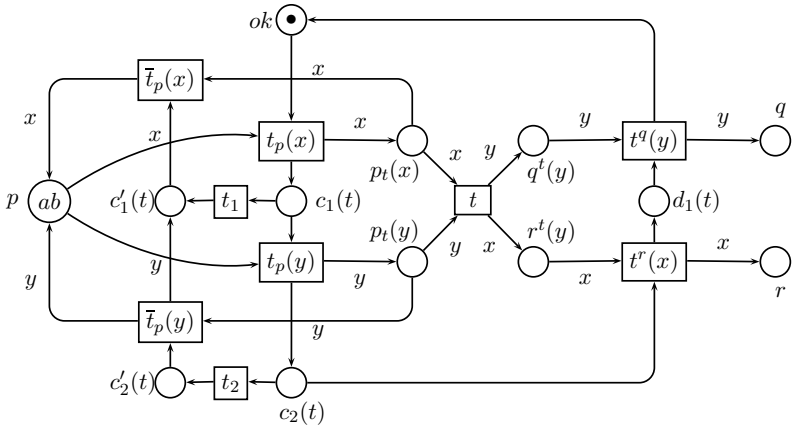

Figure 17. Simulation of the net in Fig. 16 without multisets in arcs

Moreover, for each marking M of N we take M^* that coincides with M in P, $M^*(ok) = \{\bullet\}$ and $M^*(q) = \emptyset$, otherwise.

In the first place, let us see that $M_1[t(\sigma)\rangle M_2$ implies $M_1^* \rightarrow^* M_2^*$. If $\overline{pre}(t) = (x_1, \ldots, x_n)$ and $\overline{post}(t) = (y_1, \ldots, y_m)$ we have that $\sigma(x_i) \in M(p)$ for all i and p, with $x_i \in F(p_i, t)$. Moreover, $M_1^*(ok) = 1$, so that we can fire the sequence $t_{p_{i_1}}(x_1)(\sigma_1), \ldots, t_{p_{i_n}}(x_n)(\sigma_n)$ with $\sigma_i(x_i) = \sigma(x_i)$, reaching a marking $\overline{M}$ such that $\overline{M}(p) = M_1(p) - \sigma(F(p,t))$, $\overline{M}(p_t(x_i)) = \{\sigma(x_i)\}$ and $\overline{M}(c_n(t)) = 1$. Then we can fire $t(\sigma)$ to obtain $\overline{M}_2$, that is equal to $\overline{M}_1$ in all places, except for $\overline{M}_2(p_t(x_i)) = \emptyset$ for all i, and $\overline{M}_2(p^t(y_i)) = \sigma(y_i)$ for all i. Under those conditions, we can fire the sequence $t^{p_1}(y_1)(\sigma_1'), \ldots, t^{p_m}(y_m)(\sigma_m')$ with $\sigma_i'(y_i) = \sigma(y_i)$. After that we obtain a making $\overline{M}'$ such that for all $p \in P$, $\overline{M}'(p) = \overline{M}_2(p) + \sigma(post(t)) = \overline{M}(p) + \sigma(post(t)) = M_1(p) - \sigma(pre(t)) + \sigma(post(t)) = M_2(p)$. Finally, $\sigma(ok) = 1$, $\sigma(c_i(t)) = \sigma(c_i'(t)) = 0$ for all i and t, so that we can conclude that $\overline{M}' = M_2^*$.

Conversely, let us suppose that the only marking in the image of $(\ )^*$ in $M_1^* \rightarrow^* \overline{M}$ is M_1^* and that there is no firing of $t \in T$ in that sequence. By definition of $(\ )^*$, there is a single marked place of type $c_l(t)$ y $c_l'(t)$. In the first case, we can fire t_l, which brings us to the second case. Otherwise, we can fire the sequence $\overline{t}_{p_{i_{l-1}}}(x_{l-1}), \ldots, \overline{t}_{p_{i_1}}(x_1)$ that yields marking M_1^* again, and then we trivially have $M_1 \rightarrow^* M_1$.

If there is exactly one firing of $t \in T$, for some transition $t \in T$, then we have fired the sequence $t_{p_1}(x_1)(\sigma_1), \ldots, t_{p_m}(x_m)(\sigma_m)$. In this case, there is a single i such that $\overline{M}(d_i(t)) = 1$. Then we can fire $t^{q_{i+1}}(y_{i+1}), \ldots, t^{q_m}(y_m)$ to obtain a marking of the form M_2^*. If we take $\sigma(x_i) = \sigma_i(x_i)$, let us see that $M_1[t(\sigma)\rangle M_2$. Since $\sigma_i(x_i) \in M_1^*(p)$ for all i and $pre(t) = (x_1, \ldots, x_n)$ we have that $\sigma(F(p,t)) \subseteq M_1^*(p)$ and the transition is enabled. Moreover, $M_2(p) = M_2^*(p) = M_1^*(p) - \sigma(F(p,t)) + \sigma(F(t,p)) = M_1(p) - \sigma(F(p,t)) + \sigma(F(t,p))$, by following the analogous reasoning that we followed in the previous implication. Finally, by the same reasoning followed in Lemma B, coverability is also preserved. □