Pomset bisimulation and unfolding for reset Petri nets

Thomas Chatain, Maurice Comlan, David Delfieu, Loïg Jezequel, Olivier Henri Roux

Pomset Bisimulation and Unfolding for Reset Petri Nets

Thomas Chatain[a], Maurice Comlan[b], David Delfieu[c,*], Loïg Jezequel[c], Olivier-Henri Roux[c,**]

[a] *ENS Cachan, LSV, France*
[b] *Université d'Abomey-Calavi, Bénin*
[c] *Université de Nantes and École Centrale de Nantes, LS2N UMR 6004, France*

Abstract

Reset Petri nets are a particular class of Petri nets where transition firings can remove all tokens from a place without checking if this place actually holds tokens or not. In this paper we look at partial order semantics of reset Petri nets. In particular, we propose a pomset bisimulation for comparing their concurrent behaviours. Building on this pomset bisimulation we then propose a generalization of the standard finite complete prefixes of unfolding for this class of Petri nets.

Keywords: Reset Petri nets, Pomset behaviour, Unfolding

1. Introduction

Petri nets are a well-suited formalism for specifying, modeling, and analyzing systems with conflicts, synchronization and concurrency. Many interesting properties of such systems (reachability, boundedness, liveness, deadlock, ...) are decidable for Petri nets. Over time, many extensions of Petri nets have been proposed in order to capture specific, possibly quite complex, behaviors in a more direct manner. These extensions offer more compact representations and/or increase expressive power. One can notice, in particular, a range of extensions adding new kinds of arcs to Petri nets: read arcs and inhibitor arcs [1, 2, 3] (allowing to read variables values without modifying them), and reset arcs [4] (allowing to modify the values of variables independently of their previous value).

These extensions do not only aim at constraining the executions of the models, but also at expressing other properties, like concurrency, that cannot simply be described in terms of recognized languages. For example, the commonly used

*Principal corresponding author
**Corresponding author
Email addresses: chatain@lsv.fr (Thomas Chatain), comlan@hotmail.fr (Maurice Comlan), david.delfieu@univ-nantes.fr (David Delfieu), loig.Jezequel@univ-nantes.fr (Loïg Jezequel), Olivier-h.roux@ec-nantes.fr (Olivier-Henri Roux)

read arcs (with their common semantics) do not add expressivity in terms of sequential semantics, but allow one to explictly model concurrent access to shared resources.

One interest of this explicit modeling of concurrency is that it enables dedicated analysis algorithms which exploit concurrency in order to avoid the state-space explosion due to interleavings of concurrent actions. These algorithms are partial order reduction techniques [5] and unfoldings [6]. Finite complete prefixes of unfoldings [7, 8] are sufficient to decide many verification properties and can be up to exponentially smaller than full state space exploration.

Petri nets unfolding has also gained interest of researchers beyond the verification community, for instance in planning [9] and in diagnosis [10], where the explicit distinction between concurrency and causality is a key to track the origin of observed faults. At last, in [11], the authors produce behavioral relations between workflow nets by analyzing the event relations of their respective branching processes.

Our contribution: The necessary preliminary for applying these rich techniques is to define and study the partial-order semantics of the models. This was done for Petri nets in [12, 13, 14], and for contextual Petri nets in [15, 16, 17, 18]. The aim of this paper is to do it for bounded reset Petri nets.

For that, we characterise the concurrent behaviour of reset Petri nets by defining a notion of pomset bisimulation. This has been inspired by several works on pomset behaviour of concurrent systems [19, 20, 21].

Based on true concurrency, pomset bissimulation makes fine behavioral distinctions and thus the proposed unfolding process preserves the concurrent behaviour of a reset Petri net.

We show that it is not possible to remove reset arcs from safe reset Petri nets while preserving their behaviours with respect to this pomset bisimulation. Then we propose a notion of finite complete prefixes of unfolding of safe reset Petri nets that allows for reachability analysis while preserving pomset behaviour. As a consequence of the two other contributions, these finite complete prefixes do have reset arcs.

After a questionning and a reflection on the computation of a prefix for the general class of unbounded reset Petri nets, we propose to extend the computing of a finite complete prefix to the class of (unsafe) bounded reset Petri nets. We conjecture that similar propositions and algorithms could be established for general weighted reset Petri nets, at the expense of combinatorial explosion.

This paper is organized as follows: We first give basic definitions and notations for bounded (safe) reset Petri nets. Then, in Section 3, we propose the definition of a pomset bisimulation for reset Petri nets. In Section 4, we show that, in general, there is no Petri net without reset arcs which is pomset bisimilar to a given reset Petri net. In Section 5 – building on the results of Section 4 – we propose a finite complete prefix construction for reset Petri nets. Finally, in Section 6, we extend our results to the case of unsafe reset Petri nets.

2. Reset Petri nets

Definition 1 (structure)**.** *A* reset Petri net structure *is a tuple* (P, T, F, R) *where* P *and* T *are disjoint sets of* places *and* transitions*,* $F \subseteq (P \times T) \cup (T \times P)$ *is a set of* arcs*, and* $R \subseteq P \times T$ *is a set of* reset arcs*.*

An element $x \in P \cup T$ is called a *node* and has a *preset* ${}^\bullet x = \{y \in P \cup T \ : \ (y, x) \in F\}$ and a *postset* $x^\bullet = \{y \in P \cup T \ : \ (x, y) \in F\}$. If, moreover, x is a transition, it has a set of resets ${}^\diamond x = \{y \in P \ : \ (y, x) \in R\}$.

For two nodes $x, y \in P \cup T$, we say that: x is a *causal predecessor* of y, noted $x \prec y$, if there exists a sequence of nodes $x_1 \dots x_n$ with $n \geq 2$ such that $\forall i \in [1..n-1], (x_i, x_{i+1}) \in F$, $x_1 = x$, and $x_n = y$. If $x \prec y$ or $y \prec x$ we say that x and y are *in causal relation*. The nodes x and y are *in conflict*, noted $x \# y$, if there exists two sequences of nodes $x_1 \dots x_n$ with $n \geq 2$ and $\forall i \in [1..n-1], (x_i, x_{i+1}) \in F$, and $y_1 \dots y_m$ with $m \geq 2$ and $\forall i \in [1..m-1], (y_i, y_{i+1}) \in F$, such that $x_1 = y_1$ is a place, $x_2 \neq y_2$, $x_n = x$, and $y_m = y$. It is important to note that this does not exactly define what one would intuitively called a conflict (which would be difficult to define in a general Petri net as this is a notion which depends on the order in which events occur). This definition assets that for two nodes, it exists a common ancestor that produces a divergence between some paths to which theses nodes belong, thus the conflict is potential. However, this definition is necessary and sufficient in the remainder of this section and of the paper. Notice that, in the particular nets called occurrence nets, defined later, this definition really reflects the intuitive notion of conflict. At last, nodes are said to be *concurrent* when they are not in causal relation nor in conflict.

A *marking* is a set $M \subseteq P$ of places. An occurence of a place p in a marking M is called a token. $|M|_p \in \{0, 1\}$ is the number of occurences of a place p in the marking M, i.e., the number of tokens in the place p for the marking M. A marking *enables* a transition $t \in T$ if $\forall p \in {}^\bullet t, p \in M$. In this case, t can be *fired* from M, leading to the new marking $M' = (M \setminus ({}^\bullet t \cup {}^\diamond t)) \cup t^\bullet$. The fact that M enables t and that firing t leads to M' is denoted by $M[t\rangle M'$.

Definition 2 (reset Petri net)**.** *A* reset Petri net *is a tuple* $\mathcal{N}_R = (P, T, F, R, M_0)$ *where* (P, T, F, R) *is a reset Petri net structure and* M_0 *is a marking called the* initial marking*.*

Figure 1 (left) is a graphical representation of a reset Petri net. It has five places (circles) and three transitions (squares). Its set of arcs contains seven elements (arrows) and there is one reset arc (line with a diamond).

A marking M is said to be *reachable* in a reset Petri net if there exists a sequence $M_1 \dots M_n$ of markings such that: $\forall i \in [1..n-1], \exists t \in T, M_i[t\rangle M_{i+1}$ (each marking enables a transition that leads to the next marking in the sequence), $M_1 = M_0$ (the sequence starts from the initial marking), and $M_n = M$ (the sequence leads to M). The set of all markings reachable in a reset Petri net $\mathcal{N}_R$ is denoted by $[\mathcal{N}_R\rangle$.

As an example, consider the reset Petri net of Figure 1 (left). From the marking $\{p_1, p_3\}$ (represented on the figure), the transition t_2 is enabled. Firing

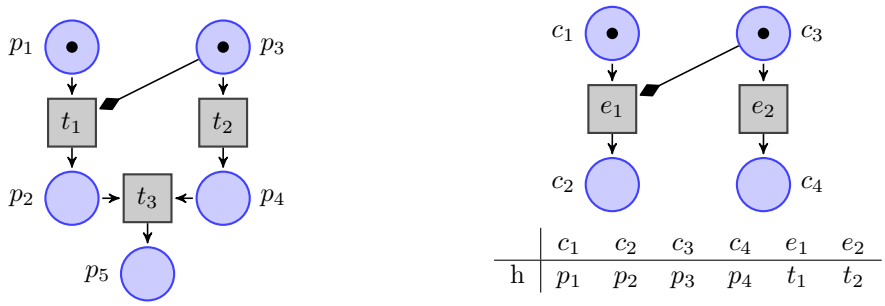


Figure 1: A reset Petri net (left) and one of its processes (right)

it leads to the new marking $\{p_1, p_4\}$. From there, t_1 is enabled, and firing it leads to $\{p_2, p_4\}$. Then, t_3 is enabled, and firing it leads to $\{p_5\}$. Hence, $\{p_5\}$ is reachable in this reset Petri net.

A reset Petri net with an empty set of reset arcs is simply called a *Petri net*.

Definition 3 (underlying Petri net)**.** *Given $\mathcal{N}_R = (P, T, F, R, M_0)$ a reset Petri net, we call its* underlying Petri net *the Petri net $\mathcal{N} = (P, T, F, \emptyset, M_0)$.*

The above formalism is in fact a simplified version of the general formalism of reset Petri nets: arcs have no multiplicity and markings are sets of places rather than multisets of places. We use it because it suffices for representing *safe* nets.

Definition 4 (safe reset Petri net)**.** *A reset Petri net (P, T, F, R, M_0) is said to be* safe *if for any reachable marking M and any transition $t \in T$, if M enables t then $(t^\bullet \setminus ({}^\bullet t \cup {}^\diamond t)) \cap M = \emptyset$.*

Informally a reset Petri net is said to be *safe*, if from any reachable marking M, no enabled transition t can produce at least one token already present in M without removing this token before.

The reader familiar with Petri nets may notice that our results generalize to larger classes of nets: unbounded reset Petri nets (where places can contain an unbounded number of tokens) for our pomset bisimulation (Section 3), and bounded reset Petri nets (where places can contain at most $k \in \mathbb{N}$ tokens) for our prefix construction (Section 5).

In the rest of the paper, unless the converse is specified, we consider reset Petri nets such that the preset of each transition t is non-empty: ${}^\bullet t \neq \emptyset$. Notice that this is not a restriction to our model: one can equip any transition t of a reset Petri net with a place p_t such that p_t is in the initial marking and ${}^\bullet p_t = p_t^\bullet = \{t\}$.

One may need to express that two (reset) Petri nets have the same behaviour. This is useful in particular for building minimal (or at least small, that is with few places and transitions) representatives of a net; or for building simple (such as loop-free) representatives of a net. A standard way to do so is to define a

bisimulation between (reset) Petri nets, and state that two nets have the same behaviour if they are bisimilar.

The behaviour of a net is an observation of its transition firing, this observation being defined thanks to a labelling of nets associating to each transition an observable label or the special unobservable label ε.

Definition 5 (labelled reset Petri net)**.** *A* labelled reset Petri net *is a tuple* $(\mathcal{N}_R, \Sigma, \lambda)$ *such that:* $\mathcal{N}_R = (P, T, F, R, M_0)$ *is a reset Petri net,* Σ *is a set of* transition labels*, and* $\lambda : T \to \Sigma \cup \{\varepsilon\}$ *is a* labelling function*.*

On every further figure of this paper, if the label matches with a node (transition or place) it will be noted only once. For example in Figure 2, the label t_2 matches to the transition t_2. If a label and a node are different, the node will be noted in parenthesis beside the label. For example, in the net $\mathcal{N}_{R,4}$ of Figure 5 the transitions e_1 and e_2 have the same label $\lambda(e_1) = \lambda(e_2) = t_2$.

In such a labelled net we extend the labelling function λ to sequences of transitions in the following way: given a sequence $t_1 \dots t_n$ (with $n \geq 2$) of transitions, if $\lambda(t_1) \in \Sigma$ then $\lambda(t_1 \dots t_n) = \lambda(t_1)\lambda(t_2 \dots t_n)$, else (that is if $\lambda(t_1) = \varepsilon$) $\lambda(t_1 \dots t_n) = \lambda(t_2 \dots t_n)$. From that, one can define bisimulation as follows.

Definition 6 (bisimulation)**.** *Let* $(\mathcal{N}_{R,1}, \Sigma_1, \lambda_1)$ *and* $(\mathcal{N}_{R,2}, \Sigma_2, \lambda_2)$ *be two labelled reset Petri nets with* $\mathcal{N}_{R,i} = (P_i, T_i, F_i, R_i, M_{0,i})$*. They are* bisimilar *if there exists a relation* $\rho \subseteq [\mathcal{N}_{R,1}\rangle \times [\mathcal{N}_{R,2}\rangle$ *(a* bisimulation*) such that:*

1. $(M_{0,1}, M_{0,2}) \in \rho$*,*
2. *if* $(M_1, M_2) \in \rho$*, then*
   (a) *for every transition* $t \in T_1$ *such that* $M_1[t\rangle M_{1,n}$ *there exists a sequence* $t_1 \dots t_n$ *of transitions from* T_2 *and a sequence* $M_{2,1} \dots M_{2,n}$ *of markings of* $\mathcal{N}_{R,2}$ *such that:* $M_2[t_1\rangle M_{2,1}[t_2\rangle \dots [t_n\rangle M_{2,n}$*,* $\lambda_2(t_1 \dots t_n) = \lambda_1(t)$*, and* $(M_{1,n}, M_{2,n}) \in \rho$
   (b) *the other way around (for every transition* $t \in T_2 \dots$*)*

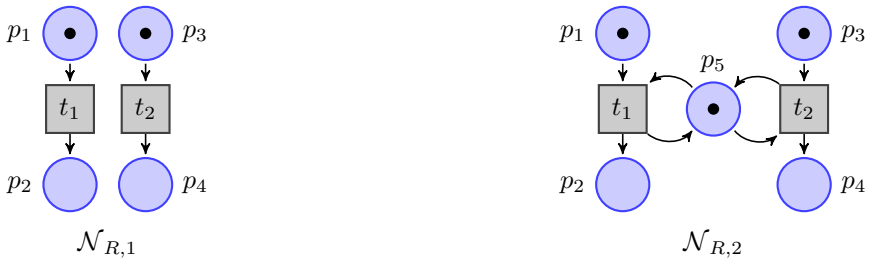


Figure 2: Two bisimilar nets

This bisimulation however abstracts from an important part of the behaviours of (reset) Petri nets: For example, consider Figure 2 where $\mathcal{N}_{R,1}$ and $\mathcal{N}_{R,2}$ are bisimilar (we identify transition names and labels). In $\mathcal{N}_{R,1}$, t_1 and

t_2 are concurrent while in $\mathcal{N}_{R,2}$ they are in causal relation. It is interesting to note that $\mathcal{N}_{R,1}$ modelizes the parallel execution of two processes ending by occurrences of t_1 and t_2, while $\mathcal{N}_{R,2}$ modelizes the mutual exclusion of t_1 and t_2.

To avoid this loss of information, a standard approach is to define bisimulations based on partially ordered sets of transitions rather than totally ordered sets of transitions (the transition sequences used in the above definition). Such bisimulations are usually called pomset bisimulations.

3. Pomset bisimulation for reset Petri nets

In this section, we propose a definition of pomset bisimulation for reset Petri nets. It is based on an ad hoc notion of processes (representations of the executions of a Petri net, concurrent counterpart of paths in automata).

3.1. Processes of reset Petri nets

We recall a standard notion of processes of Petri nets and show how it can be extended to reset Petri nets. As a first step, we define *occurrence nets* which are basically Petri nets without loops and without backward conflicts.

Definition 7 (occurrence net)**.** *An* occurrence net *is a (reset) Petri net* $(B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, M_0^{\mathcal{O}})$ *such that,* $\forall b \in B$, $\forall x \in B \cup E$*: (1)* $|{}^\bullet b| \leq 1$*, (2)* x *is not in causal relation with itself, (3)* x *is not in conflict with itself, (4)* $\{y \in B \cup E \ : \ y \prec x\}$ *is finite, (5)* $b \in M_0^{\mathcal{O}}$ *if* ${}^\bullet b = \emptyset$*.*

The places of an occurrence net are usually referred to as *conditions* and the transitions as *events*. In an occurrence net, if two nodes $x, y \in B \cup E$ are such that $x \neq y$, are not in causal relation, and are not in conflict, they are said to be *concurrent*. Moreover, in an occurrence net, the causal relation is a partial order.

There is a price to pay for having reset arcs in occurrence nets. With no reset arcs, checking if a set E of events together form a feasible execution (i.e., checking that the events from E can all be ordered such that they can be fired in this order starting from the initial marking) is linear in the size of the occurrence net (it suffices to check that E is causally closed and conflict free). With reset arcs the same task is NP-complete as stated in the below proposition.

Proposition 1. *The problem of deciding if a set* E *of events of an occurrence net with resets forms a feasible execution is NP-complete.*

Proof sketch. The problem is clearly in NP: In order to check that E is a feasible execution, it suffices to guess a corresponding firing sequence (of length $|E|$).

For NP-hardness, we reduce the problem of graph 3-coloring to executability of an occurrence net. Our construction is illustrated in Figure 3 for the graph composed of two vertices v_1 and v_2 connected by an edge.

The idea for the reduction is to build an occurrence net where each vertex v_i of the graph is represented by an event e_i, all the e_i being concurrent. We

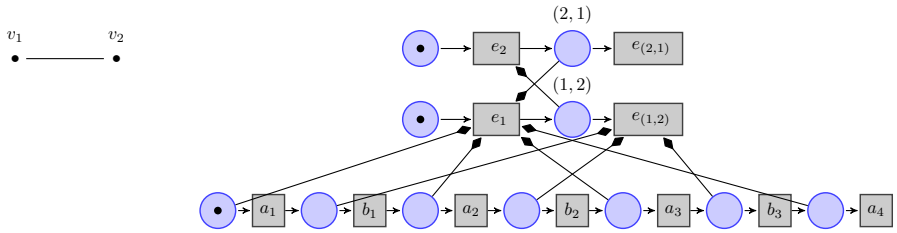


Figure 3: Reduction from 3-coloring to executability of an occurrence net. In the construction, events e_2 and $e_{(2,1)}$ reset the places of the sequence at bottom, like e_1 and $e_{(1,2)}$. For readability, these reset arcs are not represented.

add a sequence of fresh events $a_1 \prec b_1 \prec a_2 \prec b_2 \prec a_3 \prec b_3 \prec a_4$ and, using reset arcs from the places before a_1, a_2 and a_3, we enforce that, in every firing sequence containing all the events of the constructed occurrence net, the e_i fire in three separate slots: between a_1 and b_1, between a_2 and b_2 or between a_3 and b_3. Indeed, if any e_i fires outside these slots, its resets arcs consume the token in the sequence at the bottom of Figure 3 and prevents remaining events of the sequence from firing later.

These three slots represent the three colors of the coloring. Every feasible execution of the occurrence net assigns every e_i to a slot, like the color of the corresponding vertex.

It remains to represent the edges of the graph, i.e., for every edge (v_i, v_j), force e_i and e_j to fire in distinct slots. This is done using two conditions (i,j) and (j,i) and two events $e_{(i,j)}$ and $e_{(j,i)}$, with $(i,j) \in e_i^\bullet$ and ${}^\bullet e_{(i,j)} = \{(i,j)\}$ (and symmetrically $(j,i) \in e_j^\bullet$ and ${}^\bullet e_{(j,i)} = \{(j,i)\}$). Moreover $e_{(i,j)}$ and $e_{(j,i)}$ have reset arcs to conditions in the sequence at bottom, enforcing them to occur outside of the three slots allowed for the e_i. In an execution where all these events fire, assume without loss of generality that e_i fires before e_j, then $e_{(i,j)}$ must fire between e_i and e_j (while the token in (i,j) is present). This is only possible if e_i and e_j fire in different time slots.

Everything is now ready to show that the graph has a 3-coloring iff the constructed occurrence net is executable:

- Assume there exists a 3-coloring. It partitions the vertices of the graph in three sets V_1, V_2 and V_3. It induces the following firing sequence of our occurrence net: a_1 fires first, then all the e_i corresponding to the $v_i \in V_1$, then b_1, then all the $e_{(i,j)}$ for $v_i \in V_1$, then a_2, then all the e_i corresponding to the $v_i \in V_2$, then b_2, then all the $e_{(i,j)}$ for $v_i \in V_2$, then a_3, then all the e_i corresponding to the $v_i \in V_3$, then b_3, then all the $e_{(i,j)}$ for $v_i \in V_3$, then a_4. Notice that no reset arc consumes tokens in this firing sequence.

- Conversely, assume there exists a firing sequence which executes all our occurrence net. The e_i must fire in the three separate slots as we announced earlier. Because of the $e_{(i,j)}$, if v_i and v_j are connected in the

graph, then e_i and e_j have to fire in separate slots. Therefore the partition in slots induces a valid 3-coloring for the graph.

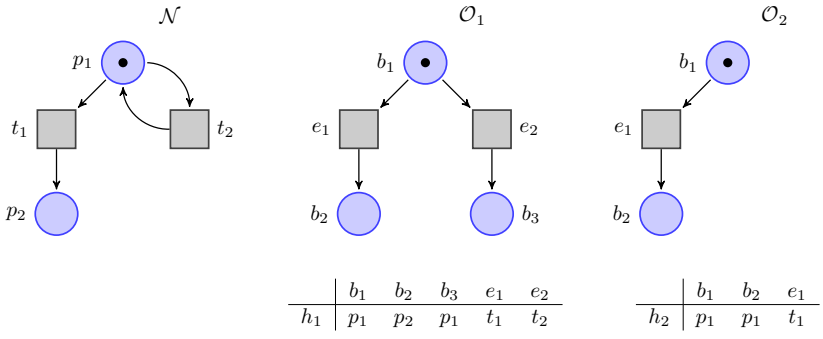
Back to the example of Figure 3, let's build a sequence where all events are present: $e_1, e_2, e_{(1,2)}, e_{(2,1)}, a_1, b_1, \ldots, a_4$. First, a_1 must come before e_1 and e_2. Let's consider, after a_1, the occurrence of e_1, then b_1 must follow: if e_2 had occurred instead of b_1, it would have prevented the occurrence of $e_{(1,2)}$; also the occurrence $e_{(1,2)}$ would have prevented the one of b_1. Hence b_1 imposes that e_1 belongs to the first slot. After b_1, two scenarios are possible: e_2 in the slot a_2, b_2 or in the slot a_3, b_3. In the first case $e_{(1,2)}$ must come before a_2, in the second case $e_{(1,2)}$ must come between b_2 and a_3. Finally, the complete guess of a feasible sequence containing every event of this net assigns to e_1 and e_2 a different slot. □

The branching processes of a Petri net are then defined as particular occurrence nets linked to the original net by *homomorphisms*.

Definition 8 (homomorphism of nets)**.** *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two Petri nets such that $\mathcal{N}_i = (P_i, T_i, F_i, \emptyset, M_{0,i})$. A mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ is a homomorphism of nets from $\mathcal{N}_1$ to $\mathcal{N}_2$ if $\forall p_1 \in P_1, \forall p_2 \in P_2, \forall t \in T_1$: (1) $h(p_1) \in P_2$, (2) $h(t) \in T_2$, (3) $p_2 \in {}^\bullet h(t) \Leftrightarrow \exists p_1' \in {}^\bullet t, h(p_1') = p_2$, (4) $p_2 \in h(t)^\bullet \Leftrightarrow \exists p_1' \in t^\bullet, h(p_1') = p_2$, (5) $p_2 \in M_{0,2} \Leftrightarrow \exists p_1' \in M_{0,1}, h(p_1') = p_2$.*

Definition 9 (processes of a Petri net)**.** *Let $\mathcal{N} = (P, T, F, \emptyset, M_0)$ be a Petri net, $\mathcal{O} = (B, E, F^{\mathcal{O}}, \emptyset, M_0^{\mathcal{O}})$ be an occurrence net, and h be a homomorphism of nets from $\mathcal{O}$ to $\mathcal{N}$. Then $(\mathcal{O}, h)$ is a* branching process *of $\mathcal{N}$ if $\forall e_1, e_2 \in E, ({}^\bullet e_1 = {}^\bullet e_2 \wedge h(e_1) = h(e_2)) \Rightarrow e_1 = e_2$. If, moreover, $\forall b \in B, |b^\bullet| \leq 1$, then $(\mathcal{O}, h)$ is a* process *of $\mathcal{N}$.*

Consider for example the Petri net $\mathcal{N}$ of Figure 4 (left). Figure 4 (middle) is one branching process of $\mathcal{N}$ (the occurrence net is represented with the homomorphism h below) and Figure 4 (right) is a process of $\mathcal{N}$.



	b_1	b_2	b_3	e_1	e_2
h_1	p_1	p_2	p_1	t_1	t_2

	b_1	b_2	e_1
h_2	p_1	p_1	t_1

Figure 4: A Petri net $\mathcal{N}$ (left), one branching process of $\mathcal{N}$ (middle, constituted of the occurrence net $\mathcal{O}_1$ and the homomorphism h_1), and one process of $\mathcal{N}$ (right, constituted of the occurrence net $\mathcal{O}_2$ and the homomorphism h_2).

Finally, a process of a reset Petri net is obtained by adding reset arcs to a process of the underlying Petri net (leading to what we call below a potential process) and checking that all its events can still be enabled and fired in some order.

We consider safe reset Petri nets whose underlying Petri net is also safe. This restriction will be released at the end of the paper.

Definition 10 (potential processes of a reset Petri net)**.**
Let $\mathcal{N}_R = (P, T, F, R, M_0)$ be a reset Petri net and $\mathcal{N}$ be its underlying Petri net, let $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, M_0^{\mathcal{O}})$ be an occurrence net, and h be a homomorphism of nets from $\mathcal{O}$ to $\mathcal{N}_R$. Then $(\mathcal{O}, h)$ is a potential process of $\mathcal{N}_R$ if (1) $(\mathcal{O}', h)$ is a process of $\mathcal{N}$ with $\mathcal{O}' = (B, E, F^{\mathcal{O}}, \emptyset, M_0^{\mathcal{O}})$, (2) $\forall b \in B, \forall e \in E, (b, e) \in R^{\mathcal{O}}$ if $(h(b), h(e)) \in R$.

Intuitively, it may be possible that no sequence of firings of $\mathcal{N}_R$ involves all the events of $(\mathcal{O}, h)$. Hence we define the processes of a reset Petri net as follows:

Definition 11 (processes of a reset Petri net)**.** *Let $\mathcal{N}_R = (P, T, F, R, M_0)$ be a reset Petri net, $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, M_0^{\mathcal{O}})$ be an occurrence net, and h be a homomorphism of nets from $\mathcal{O}$ to $\mathcal{N}_R$. Then $(\mathcal{O}, h)$ is a process of $\mathcal{N}_R$ if (1) $(\mathcal{O}, h)$ is a potential process of $\mathcal{N}_R$, and (2) if $E = \{e_1, \ldots, e_n\}$ then $\exists M_1, \ldots, M_n \subseteq B$ such that $M_0^{\mathcal{O}}[e_{k_1}\rangle M_1[e_{k_2}\rangle \ldots [e_{k_n}\rangle M_n$ with $\{k_1, \ldots, k_n\} = \{1, \ldots, n\}$.*

Notice that processes of reset Petri nets and processes of Petri nets do not exactly have the same properties. In particular, two properties are central in defining pomset bisimulation for Petri nets and do not hold for reset Petri nets.

Property 1. *In any process of a Petri net with set of events E, consider any sequence of events $e_1 e_2 \ldots e_n$ (1) that contains all the events in E, (2) such that $\forall i, j \in [1..n]$ if $e_i \prec e_j$ then $i < j$. Necessarily, there exist markings $M_1, \ldots, M_n$ such that $M_0^{\mathcal{O}}[e_1\rangle M_1[e_2\rangle \ldots [e_n\rangle M_n$.*

This property (which, intuitively, expresses that processes are partially ordered paths) is no longer true for reset Petri nets. Consider for example the reset Petri net of Figure 1 (left). Figure 1 (right) is one of its processes (the occurrence net with the homomorphism h below). As not $e_2 \prec e_1$, there should exist markings M_1, M_2 such that $M_0[e_1\rangle M_1[e_2\rangle M_2$. However, $M_0 = \{c_1, c_3\}$ indeed enables e_1, but the marking M_1 such that $M_0[e_1\rangle M_1$ is $\{c_2\}$, which does not enable e_2.

Property 2. *In a process of a Petri net all the sequences of events $e_1 e_2 \ldots e_n$ verifying (1) and (2) of Property 1 lead to the same marking (i.e., M_n is always the same), thus uniquely defining a notion of maximal marking of a process.*

This property defines the marking reached by a process. As a corollary of Property 1 not holding for reset Petri nets, there is no uniquely defined notion of maximal marking in their processes. Back to the example, $\{c_2\}$ is maximal (no event can be fired from it) as well as $\{c_2, c_4\}$.

To transpose the spirit of Properties 1 and 2 to processes of reset Petri nets, we define below a notion of maximal markings in such processes.

Definition 12 (maximal markings)**.** *Let $\mathcal{P} = (\mathcal{O}, h)$ be a process with set of events $E = \{e_1, \ldots, e_n\}$ and initial marking $M_0^{\mathcal{O}}$ of a reset Petri net. The set $M_{max}(\mathcal{P})$ of maximal markings of $\mathcal{P}$ contains exactly the markings M such that $\exists M_1, \ldots, M_{n-1}$, verifying $M_0^{\mathcal{O}}[e_{k_1}\rangle M_1[e_{k_2}\rangle \ldots M_{n-1}[e_{k_n}\rangle M$ for some $\{k_1, \ldots, k_n\} = \{1, \ldots, n\}$.*

In other words, the maximal markings of a process are all the markings that are reachable in it using all its events. This, in particular, excludes $\{c_2\}$ in the above example.

3.2. Abstracting processes

We show how processes of labelled reset Petri nets can be abstracted as partially ordered multisets (pomsets) of labels.

Definition 13 (pomset abstraction of processes)**.** *Let $(\mathcal{N}_R, \Sigma, \lambda)$ be a labelled reset Petri net and $(\mathcal{O}, h)$ be a process of $\mathcal{N}_R$ with $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, M_0^{\mathcal{O}})$. Define $E' = \{e \in E \ : \ \lambda(h(e)) \neq \varepsilon\}$. Define $\lambda' : E' \to \Sigma$ as the function such that $\forall e \in E', \lambda'(e) = \lambda(h(e))$. Define moreover $< \ \subseteq E' \times E'$ as the relation such that $e_1 < e_2$ if $e_1 \prec e_2$ (e_1 is a causal predecessor of e_2 in $\mathcal{O}$). Then, $(E', <, \lambda')$ is the* pomset abstraction *of $(\mathcal{O}, h)$.*

This abstraction $(E, <, \lambda')$ of a process is called its pomset abstraction because it can be seen as a multiset of labels (several events may have the same associated label by λ') that are partially ordered by the $<$ relation. In order to compare processes with respect to their pomset abstractions, we also define the following equivalence relation.

Definition 14 (pomset equivalence)**.** *Let $(E, <, \lambda)$ and $(E', <', \lambda')$ be the pomset abstractions of two processes $\mathcal{P}$ and $\mathcal{P}'$. These processes are* pomset equivalent*, noted $\mathcal{P} \equiv \mathcal{P}'$ if there exists a bijection $f : E \to E'$ such that $\forall e_1, e_2 \in E$: (1) $\lambda(e_1) = \lambda'(f(e_1))$, and (2) $e_1 < e_2$ if $f(e_1) <' f(e_2)$.*

Intuitively, two processes are pomset equivalent if their pomset abstractions define the same pomset: same multisets of labels with same partial orderings. Finally, we also need to be able to abstract processes as sequences of labels.

Definition 15 (linear abstraction)**.** *Let $(\mathcal{N}_R, \Sigma, \lambda)$ be a labelled reset Petri net, let $\mathcal{P} = (\mathcal{O}, h)$ be a process of $\mathcal{N}_R$ with $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, M_0^{\mathcal{O}})$, and let M be a reachable marking in $\mathcal{O}$. Define $\lambda' : E \to \Sigma$ as the function such that $\forall e \in E, \lambda'(e) = \lambda(h(e))$. The* linear abstraction *of $\mathcal{P}$ with respect to M is the set $lin(M, \mathcal{P})$ such that a sequence of labels ω is in $lin(M, \mathcal{P})$ if in $\mathcal{O}$ there exist markings $M_1, \ldots, M_{n-1}$ and events $e_1, \ldots, e_n$ such that $M_0^{\mathcal{O}}[e_1\rangle M_1[e_2\rangle \ldots M_{n-1}[e_n\rangle M$ and $\lambda'(e_1 \ldots e_n) = \omega$.*

3.3. Pomset bisimulation

We now define a notion of pomset bisimulation between reset Petri nets, inspired by [19, 20, 21]. Intuitively, two reset Petri nets are pomset bisimilar if there exists a relation between their reachable markings such that the markings that can be reached by pomset equivalent processes from two markings in relation are them selves in relation. This is formalized by the Definition 16.

Definition 16 (pomset bisimulation for reset nets)**.** *Let* $(\mathcal{N}_{R,1}, \Sigma_1, \lambda_1)$ *and* $(\mathcal{N}_{R,2}, \Sigma_2, \lambda_2)$ *be two labelled reset Petri nets with* $\mathcal{N}_{R,i} = (P_i, T_i, F_i, R_i, M_{0,i})$*. They are* pomset bisimilar *if there exists a relation* $\rho \subseteq [\mathcal{N}_{R,1}\rangle \times [\mathcal{N}_{R,2}\rangle$ *(called a* pomset bisimulation*) such that:*

1. $(M_{0,1}, M_{0,2}) \in \rho$,
2. *if* $(M_1, M_2) \in \rho$*, then*
   (a) *for every process* $\mathcal{P}_1$ *of* $(P_1, T_1, F_1, R_1, M_1)$ *there exists a process* $\mathcal{P}_2$ *of* $(P_2, T_2, F_2, R_2, M_2)$ *such that* $\mathcal{P}_1 \equiv \mathcal{P}_2$ *and*
   - $\forall M'_1 \in M_{max}(\mathcal{P}_1), \exists M'_2 \in M_{max}(\mathcal{P}_2)$ *such that* $(M'_1, M'_2) \in \rho$,
   - $\forall M'_1 \in M_{max}(\mathcal{P}_1), \forall M'_2 \in M_{max}(\mathcal{P}_2)$, $(M'_1, M'_2) \in \rho \Rightarrow lin(M'_1, \mathcal{P}_1) = lin(M'_2, \mathcal{P}_2)$.

   (b) *the other way around (for every process* $\mathcal{P}_2 \ldots$*)*

Remark that pomset bisimulation implies bisimulation, as expressed by the following proposition. The converse is obviously not true.

For example, in the Figure 2, $\mathcal{N}_{R,1}$ and $\mathcal{N}_{R,2}$ are bisimilar but not pomset-bisimilar. It can also be noted that while every Petri net is bisimilar to its marking graph, it is not necessarily pomset bisimilar to it. In the Figure 5, the reset Petri nets $\mathcal{N}_{R,3}$ and $\mathcal{N}_{R,4}$ (with $\lambda(e_1) = \lambda(e_2) = t_2$) are pomset bisimilar to $\mathcal{N}_{R,1}$ of Figure 2.

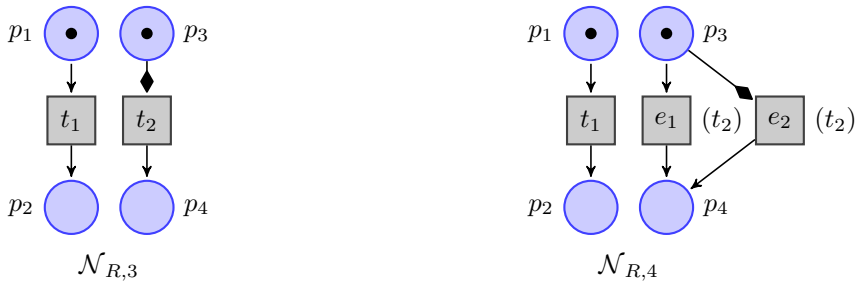


Figure 5: Pomset bisimilar reset Petri nets

Proposition 2. *Let* $(\mathcal{N}_{R,1}, \Sigma_1, \lambda_1)$ *and* $(\mathcal{N}_{R,2}, \Sigma_2, \lambda_2)$ *be two pomset bisimilar labelled reset Petri nets, then* $(\mathcal{N}_{R,1}, \Sigma_1, \lambda_1)$ *and* $(\mathcal{N}_{R,2}, \Sigma_2, \lambda_2)$ *are bisimilar.*

Proof. It suffices to notice that Definition 6 can be obtained from Definition 16 by restricting the processes considered, taking only those with exactly one transition whose label is different from ε. □

4. Reset arcs removal and pomset bisimulation

From now on, we consider finite (reset) Petri nets, i.e., their sets of places and transitions are finite.

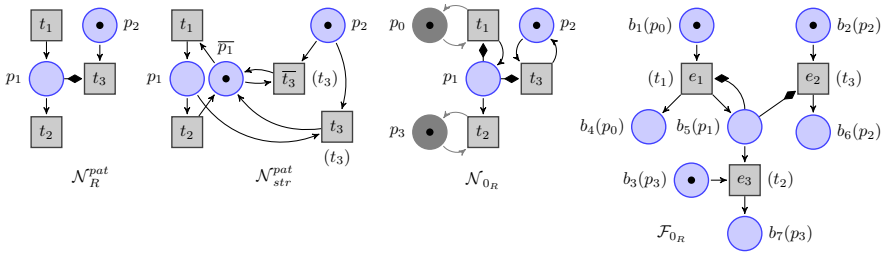


Figure 6: A remarkable pattern $\mathcal{N}_R^{pat}$ and its structural transformation $\mathcal{N}_{str}^{pat}$, a labelled reset Petri net $\mathcal{N}_{0_R}$ including the pattern $\mathcal{N}_R$, and a finite complete prefix $\mathcal{F}_{0_R}$ of $\mathcal{N}_{0_R}$. Transition labels are given on transitions.

In this section, we prove that it is, in general, not possible to remove reset arcs from safe reset Petri nets while preserving their behaviours up to pomset bisimulation. More precisely, we prove that it is not possible to build a safe labelled Petri net without reset arcs which is pomset bisimilar to a given safe labelled reset Petri net (while this is out of the scope of this paper, the reader familiar with Petri nets may notice that this is the case for bounded labelled Petri net). For that, we exhibit a particular pattern – Figure 6 (left) – and show that a reset Petri net including this pattern cannot be pomset bisimilar to a Petri net without reset arcs.

As a first intuition of this fact, let us consider the following structural transformation that removes reset arcs from a reset Petri net.

Definition 17 (Structural transformation)**.** *Let $(\mathcal{N}_R, \Sigma, \lambda)$ be a labelled reset Petri net such that $\mathcal{N}_R = (P, T, F, R, M_0)$, its structural transformation is the labelled Petri net $(\mathcal{N}_{R,str}, \Sigma_{str}, \lambda_{str})$ where $\mathcal{N}_{R,str} = (P_{str}, T_{str}, F_{str}, \emptyset, M_{0,str})$ such that:*

$$
\begin{aligned}
P_{str} &= P \cup \overline{P} \text{ with } \overline{P} = \{\overline{p} \ : \ p \in P \wedge \exists t \in T, (p,t) \in R\},\\
T_{str} &= T \cup \overline{T} \text{ with } \overline{T} = \{\overline{t} \ : \ t \in T \wedge {}^\diamond t \neq \emptyset\},\\
F_{str} &= F \cup \{(p,\overline{t}) \ : \ \overline{t} \in \overline{T}, (p,t) \in F\} \cup \{(\overline{t},p) \ : \ \overline{t} \in \overline{T}, (t,p) \in F\} && (1)\\
&\quad \cup \{(\overline{p},t) \ : \ \overline{p} \in \overline{P}, (t,p) \in F\} \cup \{(t,\overline{p}) \ : \ \overline{p} \in \overline{P}, (p,t) \in F\} && (2)\\
&\quad \cup \{(\overline{p},\overline{t}) \in \overline{P} \times \overline{T} \ : \ (t,p) \in F\} \cup \{(\overline{t},\overline{p}) \in \overline{T} \times \overline{P} \ : \ (p,t) \in F\} && (3)\\
&\quad \cup \{(p,t),(t,\overline{p}),(\overline{p},\overline{t}),(\overline{t},\overline{p}) \ : \ (p,t) \in R\}, && (4)\\
M_{0,str} &= M_0 \cup \{\overline{p} \in \overline{P} \ : \ p \notin M_0\},
\end{aligned}
$$

and moreover, $\Sigma_{str} = \Sigma$, $\forall t \in T, \lambda_{str}(t) = \lambda(t)$, and $\forall \overline{t} \in \overline{T}, \lambda_{str}(\overline{t}) = \lambda(t)$.

Intuitively, in this transformation, for each reset arc (p, t), a copy $\overline{p}$ of p and a copy $\overline{t}$ of t are created. The two places are such that p is marked if and only if $\overline{p}$ is not marked, the transition t will perform the reset when p is marked and $\overline{t}$ will perform it when p is not marked (i.e., when $\overline{p}$ is marked). For that, new arcs are added to F such that: $\overline{t}$ mimics t (1), the link between p and $\overline{p}$ is enforced (2, 3), and the resets are either performed by t or $\overline{t}$ depending of the markings of p and $\overline{p}$ (4). This is examplified in Figure 6 (left and middle left).

Lemma 1. *A labelled reset Petri net $(\mathcal{N}_R, \Sigma, \lambda)$ and its structural transformation $(\mathcal{N}_{R,str}, \Sigma_{str}, \lambda_{str})$ as defined in Definition 17 are bisimilar.*

Proof. The bisimulation relation is $\rho \subseteq [\mathcal{N}_R\rangle \times [\mathcal{N}_{R,str}\rangle$ defined by $(M, M_{struct}) \in \rho$ iff $\forall p \in P, |M|_p = |M_{struct}|_p$ and $\forall p \in P$ such that $\overline{p} \in \overline{P}$, we have $|M_{struct}|_p + |M_{struct}|_{\overline{p}} = 1$. See proof of lemma 3 for k-bounded nets. □

For the transformation of Definition 17, a reset Petri net and its transformation are bisimilar but not always pomset bisimilar. This can be remarked on any safe reset Petri net including the pattern $\mathcal{N}_R^{pat}$ of Figure 6. Indeed, this transformation adds in $\mathcal{N}_{str}^{pat}$ a causality relation between the transition labelled by t_1 and each of the two transitions labelled by t_3. From the initial marking of $\mathcal{N}_{str}^{pat}$, for any process whose pomset abstraction includes both t_1 and t_3, these two labels are causally ordered. While, from the initial marking of $\mathcal{N}_R^{pat}$ there is a process whose pomset abstraction includes both t_1 and t_3 but does not order them. We now generalize this result.

Let us consider the labelled reset Petri Net $\mathcal{N}_{0_R}$ of Figure 6 (middle right). It uses the pattern $\mathcal{N}_R^{pat}$ of Figure 6 in which t_1 and t_3 can be fired in different order infinitely often. In this net, the transitions with labels t_1 and t_3 are not in causal relation.

Notice that the firing of t_3 prevents the firing of t_2; then t_3 and t_2 are in conflict and share an input place which has to be marked again after the firing of t_1. This place generates a causality between t_1 and t_3. This is formalized and proved in the following lemma.

Lemma 2. *Any safe labelled Petri net with no reset arcs which is bisimilar (see definition 6) to $\mathcal{N}_{0_R}$ has a causal relation between two transitions labelled by t_1 and t_3 respectively.*

Proof. Assume there exists a safe labelled Petri net $N_0 = (\mathcal{N}_0, \Sigma_0, \lambda_0)$ bisimilar to the labelled reset Petri net $\mathcal{N}_{0_R}$ without any transitions $t \prec t'$ such that $\lambda_0(t) = t_1$ and $\lambda_0(t') = t_3$.

Let us consider in $\mathcal{N}_{0_R}$ a marking M_R such that p_1 and p_2 are marked. Both t_2 and t_3 are fireable (in $\mathcal{N}_{0_R}$ we identify transitions with their labelling as this is not ambiguous).

By definition of the bisimulation, in N_0 there exists a marking M bisimilar to M_R and from which two sequences $\tau_1 \dots \tau_{k_\tau} t''$ and $\varepsilon_1 \dots \varepsilon_{k_\varepsilon} t'$, with $\lambda_0(\tau_1 \dots \tau_{k_\tau} t'') = \lambda_0(t'') = t_2$ and $\lambda_0(\varepsilon_1 \dots \varepsilon_{k_\varepsilon} t') = \lambda_0(t') = t_3$, are fireable in the orders given by the sequences. Note that the set of transitions in $\varepsilon_1 \dots \varepsilon_{k_\varepsilon}$

and the set of transitions in $\tau_1 \dots \tau_{k_\tau}$ are not necessarily disjoint. Without loss of generality, we take M, $\tau_1 \dots \tau_{k_\tau}$, and t'' such that the sequence $\tau_1 \dots \tau_{k_\tau} t''$ can be fired infinitely often, that is, such that there exists a sequence of firing of transitions from the initial marking of N_0 in which the subsequence $\tau_1 \dots \tau_{k_\tau} t''$ appears infinitely many times (this is possible due to the finiteness of the set of transitions of N_0 and the fact that t_2 can be fired infinitely often in $\mathcal{N}_{0_R}$).

When t' occurs from M, the firing of t'' becomes impossible. Otherwise a sequence of transitions w such that $\lambda_0(w) = t_3 t_2$ would be possible in N_0, which contradicts bisimilarity with $\mathcal{N}_{0_R}$ where firing t_2 immediately after t_3 (i.e., with no firing of t_1 in the meantime) is not possible.

From that, one can deduce that from M the two sequences $\tau_1 \dots \tau_{k_\tau} t''$ and $\varepsilon_1 \dots \varepsilon_{k_\varepsilon} t'$ necessarily have a direct conflict: there are a transition τ in $\tau_1 \dots \tau_{k_\tau} t''$ and a transition ε in $\varepsilon_1 \dots \varepsilon_{k_\varepsilon} t'$ whose presets share at least one place. We call p'_1 such a place both in ${}^\bullet\tau$ and in ${}^\bullet\varepsilon$. We have $p'_1 \prec t''$ (recall that $\lambda_0(t'') = t_2$) and $p'_1 \prec t'$ (recall that $\lambda_0(t') = t_3$).

Notice that it can exist a non-empty set S_{P_1} of such places p'_1, because $|{}^\bullet\tau \cap {}^\bullet\varepsilon| \geq 1$. Since t_2 and t_3 can occur infinitely often in $\mathcal{N}_{0_R}$, and since the number of places in N_0 is finite, one could have chosen t' and thus ε such that all the places in S_{P_1} is marked infinitely often in some infinite sequence of transitions firing in N_0 in which $\tau_1 \dots \tau_{k_\tau} t''$ appears infinitely often and which reaches M infinitely often (recall that we chose t'' to have such a sequence). Assume that we chose p'_1 in such a S_{P_1}.

As a last step before concluding our proof, we show that $t'' \prec p'_1$. Assume this is not the case. Remark that firing $\tau_1 \dots \tau_{k_\tau} t''$ removes, at some point in the sequence of firings, p'_1 from the marking, by firing τ. However, p'_1 is marked infinitely often in some infinite sequence w of transitions firing in N_0 in which $\tau_1 \dots \tau_{k_\tau} t''$ appears infinitely often and which reaches M infinitely often. If $t'' \prec p'_1$ is false, it means that, at some point in w, p'_1 is marked thanks to a transition that is not a causal successor of t'' (i.e., t'' is not a causal predecessor of this transition), all the places in the preset of this transition must neither be causal successors of t''. By induction, one can find an infinite subsequence of w of transitions firing which infinitely often marks p'_1 while it is already marked. This is in contradiction with the assumption that N_0 is safe (for the reader familiar with Petri nets, this would work exactly the same for bounded Petri nets in a more general setting). Thus, $t'' \prec p'_1$.

We have shown that $t'' \prec p'_1$ and $p'_1 \prec t'$. Moreover, by definition of bisimilarity we know that there exists t in N_0 such that $t \prec t''$ and $\lambda_0(t) = t_1$ (because $t_1 \prec t_2$ in $\mathcal{N}_{0_R}$). Hence, by transitivity, we get $t \prec t'$ with $\lambda_0(t) = t_1$ and $\lambda_0(t') = t_3$, which concludes the proof. □

Proposition 3. *There is no finite safe labelled Petri net (i.e., without reset arcs) that is pomset bisimilar to the labelled reset Petri net* $\mathcal{N}_{0_R}$.

Proof. Any finite safe labelled Petri net with no reset arcs which is bisimilar to $\mathcal{N}_{0_R}$ has a causal relation between two transitions labelled by t_1 and t_3 respectively (Lemma 2). From that, by Proposition 2, we get that any such

labelled Petri net $\mathcal{N}$ which would be pomset bisimilar to $\mathcal{N}_{0_R}$ would have a process from its initial marking whose pomset abstraction is such that some occurrence of t_1 and some occurrence of t_3 are ordered, while this is never the case in the processes of $\mathcal{N}_{0_R}$. This prevents $\mathcal{N}$ from being pomset bisimilar to $\mathcal{N}_{0_R}$, and thus leads to a contradiction, proving the proposition. □

5. Finite complete prefixes of unfolding of safe reset Petri nets

In this section, we propose a notion of finite complete prefixes of unfolding of safe reset Petri nets that allows reachability analysis while preserving pomset behaviour.

The results of the previous section imply that these finite complete prefixes do have reset arcs.

The unfolding of a Petri net is a particular branching process (generally infinite) representing all its reachable markings and ways to reach them. It also preserves concurrency.

Definition 18 (Unfolding of a Petri net)**.** *The unfolding of a net can be defined as the union of all its branching processes [6] or equivalently as its largest branching process (with respect to inclusion).*

In the context of reset Petri nets, no notion of unfolding has been defined yet. Accordingly to our notion of processes for reset Petri nets and because of Proposition 4 below we propose Definition 19. In it and the rest of the paper, nets and labelled nets are identified (each transition is labelled by itself) and labellings of branching processes are induced by homomorphisms (as for pomset abstraction).

Definition 19 (Unfolding of a reset Petri net)**.** *Let $\mathcal{N}_R = (P, T, F, R, M_0)$ be a safe reset Petri net and $\mathcal{N} = (P, T, F, \emptyset, M_0)$ be its underlying Petri net (that we assume to be safe). Let $\mathcal{U} = (B, E, F^U, \emptyset, M_0^U)$ be the unfolding of $\mathcal{N}$. The unfolding $\mathcal{U}_R = (B, E, F^U, R^U, M_0^U)$ of $\mathcal{N}_R$ is obtained by adding reset arcs to $\mathcal{U}$ as follows : $\forall b \in B, \forall e \in E, (b, e) \in R^U$ if $(h(b), h(e)) \in R$.*

Proposition 4. *Any safe (labelled) reset Petri net $\mathcal{N}_R$ and its unfolding $\mathcal{U}_R$ are pomset bisimilar.*

Proof. (Sketch) This extends a result of [22], stating that two Petri nets having the same unfolding (up to isomorphism) are pomset bisimilar (for a notion of bisimulation coping with our in absence of resets). Clearly, a Petri net $\mathcal{N}$ and its unfolding $\mathcal{U}$ have the same unfolding, $\mathcal{U}$ itself. Thus, $\mathcal{N}$ and $\mathcal{U}$ are pomset bisimilar.

Moreover, from Definition 11, one gets that the processes of $\mathcal{N}_R$ are a subset of the processes of $\mathcal{N}$ (to which reset arcs are added). Similarly, the processes of $\mathcal{U}_R$ are a subset of the processes of $\mathcal{U}$ (to which reset arcs are added). Because $\mathcal{N}$ and $\mathcal{U}$ are pomset bisimilar, they have the same processes (up to pomset abstraction). Thus, $\mathcal{N}_R$ and $\mathcal{U}_R$ have the same processes (up to pomset abstraction) as well.

Finally, by definition of pomset bisimulation, in $\mathcal{N}$ and $\mathcal{U}$ two processes taken from markings in bisimulation and with the same pomset abstraction, must also reach markings in bisimulation. Because the addition of reset arcs mimics the resets arcs of $\mathcal{N}_R$ in $\mathcal{U}_R$ (i.e., adding reset arcs to the processes of $\mathcal{N}$ or to the processes of $\mathcal{U}$ is done exactly in the same way) and because their processes are the same, we get the same property about bisimulation between markings in $\mathcal{N}_R$ and $\mathcal{U}_R$ than in $\mathcal{N}$ and $\mathcal{U}$. □

Petri nets unfolding is however unpractical for studying Petri nets behaviour as it is generally an infinite object. In practice, finite complete prefixes of it are preferred [8, 7].

Definition 20 (finite complete prefix, reachable marking preservation)**.** *A* finite complete prefix *of the unfolding of a safe Petri net $\mathcal{N}$ is a finite branching processes $(\mathcal{O}, h)$ of $\mathcal{N}$ verifying the following property of* reachable marking preservation*: a marking M is reachable in $\mathcal{N}$ if there exists a reachable marking M' in $\mathcal{O}$ such that $M = \{h(b) \ : \ b \in M'\}$.*

In this section, we propose an algorithm for the construction of finite complete prefixes for safe reset Petri nets. For that, we assume the existence of an algorithm for building finite complete prefixes of safe Petri nets (without reset arcs). Notice that such algorithms indeed do exist [8, 7].

Because of Proposition 3, we know that such finite prefixes should have reset arcs to preserve pomset behaviour. We first remark that directly adding reset arcs to finite complete prefixes of underlying nets would not work.

Proposition 5. *Let $\mathcal{U}$ be the unfolding of the underlying Petri Net $\mathcal{N}$ of a safe reset Petri net $\mathcal{N}_R$, let $\mathcal{F}$ be one of its finite and complete prefixes. Let $\mathcal{F}'$ be the object obtained by adding reset arcs to $\mathcal{F}$ according to (2) in Definition 10. The reachable marking preservation is in general not verified by $\mathcal{F}'$ (with respect to $\mathcal{N}_R$).*

Proof. Let us consider the reset Petri net $\mathcal{N}_R$ of Figure 7 (left).

Applying the prefix construction procedure of [8] or [7] on the unfolding $\mathcal{U}$ of its underlying Petri net leads to the finite prefix $\mathcal{F}$ of Figure 7 (middle). The object $\mathcal{F}'$ obtained by adding reset arcs to $\mathcal{F}$ is represented in Figure 7 (right). It does not verify the reachable marking preservation property. Indeed, in $\mathcal{N}_R$, the sequence of transition firings $t_1t_3t_2t_4t_5$ allows to reach the marking $\{p_6\}$, while in $\mathcal{F}'$ no sequence of transition firings permits to reach the marking $\{b_6\}$ (which is the only one which could correspond to $\{p_6\}$). □

The proof of this proposition relies on the fact that some reachable markings of $\mathcal{N}_R$ are not represented in $\mathcal{F}'$. This suggests that this prefix is not big enough. We however know an object that contains, for sure, every reachable marking of $\mathcal{N}_R$ along with a way to reach each of them: its structural transformation $\mathcal{N}_{R,str}$ (Definition 17). We thus propose in Algorithm 1 to compute finite prefixes of reset Petri nets from their structural transformations: in this algorithm, $\mathcal{F}_{str}$

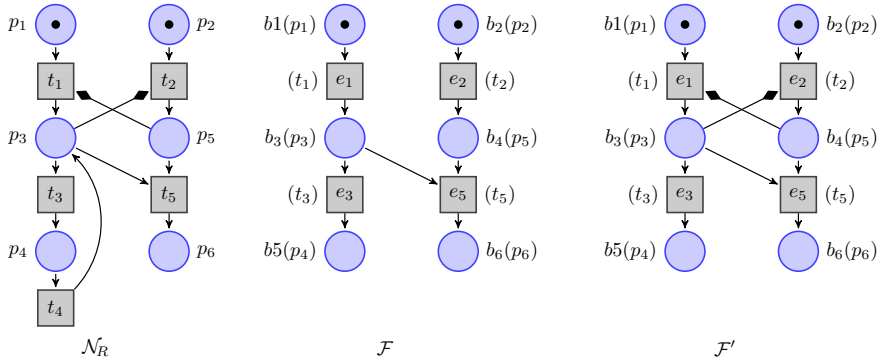


Figure 7: A reset Petri net $\mathcal{N}_R$ (left), a finite complete prefix $\mathcal{F}$ of its underlying Petri net (middle), and the same prefix after addition of reset arcs $\mathcal{F}'$ (right).

is used to determine the deepness of the prefix (i.e., the length of the longest chain of causally ordered transitions).

Algorithm 1: Finite complete prefix for safe reset Petri nets

Data: a safe reset Petri net $\mathcal{N}_R$
Result: a finite complete prefix $\mathcal{F}_R$ for $\mathcal{N}_R$
Step 1: compute the structural transformation $\mathcal{N}_{R,str}$ of $\mathcal{N}_R$
Step 2: compute a finite complete prefix $\mathcal{F}_{str}$ of $\mathcal{N}_{R,str}$
Step 3: compute a finite prefix $\mathcal{F}$ of $\mathcal{U}$ (the unfolding of the underlying net $\mathcal{N}$) that simulates $\mathcal{F}_{str}$ (a labelled net $\mathcal{N}_2$ simulates a labelled net $\mathcal{N}_1$ if they verify Definition 6 except for condition 2.b.)
Step 4: compute $\mathcal{F}_R$ by adding reset arcs from $\mathcal{N}_R$ to $\mathcal{F}$ according to (2) in Definition 10

Applying Algorithm 1 to the net $\mathcal{N}_{0_R}$ of Figure 6 (middle right) – using the algorithm from [7] at step 2 – leads to the reset Petri net $\mathcal{F}_{0_R}$ of Figure 6 (right).

Notice that the computation of $\mathcal{F}_{str}$ – step 1 and 2 – can be done in exponential time and space with respect to the size of $\mathcal{N}_R$. The computation of $\mathcal{F}$ from $\mathcal{F}_{str}$ (step 3) is linear in the size of $\mathcal{F}$. And, the addition of reset arcs (step 4) is at most quadratic in the size of $\mathcal{F}$.

We conclude this section by showing that Algorithm 1 actually builds finite complete prefixes of reset Petri nets.

Proposition 6. *The object $\mathcal{F}_R$ obtained by Algorithm 1 from a safe reset Petri net $\mathcal{N}_R$ is a finite and complete prefix of the unfolding of $\mathcal{N}_R$.*

Proof. Notice that if $\mathcal{N}_R$ is safe, then $\mathcal{N}_{R,str}$ is safe as well. Thus $\mathcal{F}_{str}$ is finite by definition of finite complete prefixes of Petri nets (without reset arcs). $\mathcal{F}_{str}$ is finite and has no node in causal relation with itself (i.e., no cycle), hence any

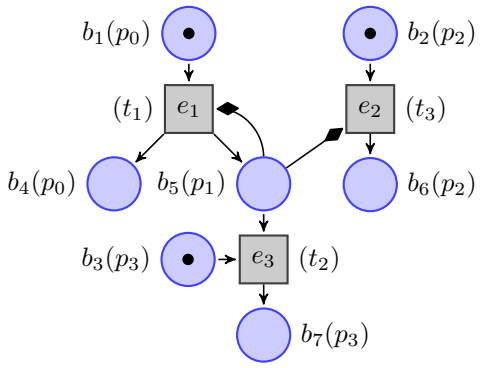


Figure 8: A finite complete prefix of the unfolding of the safe reset Petri net $\mathcal{N}_{0_R}$ of Figure 6.

net bisimilar with it is also finite, this is in particular the case of $\mathcal{F}$. Adding reset arcs to a finite object does not break its finiteness, so $\mathcal{F}_R$ is finite.

Moreover, $\mathcal{F}_{str}$ is complete by definition of finite complete prefixes of Petri nets (without reset arcs). As $\mathcal{F}$ simulates $\mathcal{F}_{str}$ it must also be complete (it can only do more). The reset arcs addition removes semantically to $\mathcal{F}$ only the unexpected sequences (i.e., the sequence which are possible in $\mathcal{F}$ but not in $\mathcal{F}_{str}$). Therefore, $\mathcal{F}_R$ is complete. □

6. Unsafe reset Petri nets

In this section we extend the above results to unsafe reset Petri nets. In other words, nets where it is possible to fire transitions when their postsets are marked. This implies to be able to represent the fact that a place is marked with more than one token. We achieve it by representing markings as multisets of places rather than sets of places.

6.1. Markings as multisets

Definition 21 (Multiset)**.** *A* multiset *is a couple $M = (S, f)$ where S is a set and $f : S \to \mathbb{N} \cup \{\infty\}$ is a function associating a multiplicity to each element in S.*

For simplicity and homogeneity with the above definitions (i.e., to avoid using the f function), we adopt the following notations: $|M| = \sum_{s \in S} f(s)$ is the cardinality of M, $|M|_s = f(s)$ is the number of occurrences of the element s in M, and we note $s \in M$ when $|M|_s >= 1$. We also need to be able to perform unions and differences of multisets in order to express transition firing. They can be defined as follows.

Definition 22 (Union of multisets)**.** *The* union *of two multisets $M_1 = (S_1, f_1)$ and $M_2 = (S_2, f_2)$ is the multiset $M_1 \oplus M_2 = (S_1 \cup S_2, f)$ where $\forall s_1 \in S_1 \setminus S_2, f(s_1) = f_1(s_1)$, $\forall s_2 \in S_2 \setminus S_1, f(s_2) = f_2(s_2)$, and $\forall s \in S_1 \cap S_2, f(s) = f_1(s) + f_2(s)$.*

Definition 23 (Difference of multisets)**.** *The* difference *of two multisets* $M_1 = (S_1, f_1)$ *and* $M_2 = (S_2, f_2)$ *is the multiset* $M_1 \ominus M_2 = (S, f)$ *where* $S = (S_1 \setminus S_2) \cup S'$ *with* $S' = \{s \in S_1 \cap S_2 \ : \ f_1(s) > f_2(s)\}$, $\forall s_1 \in S_1 \setminus S_2, f(s_1) = f_1(s_1)$, *and* $\forall s \in S', f(s) = f_1(s) - f_2(s)$.

From that, Petri nets are defined exactly as before. The only difference being that markings are multisets and that the firing rules are defined using multisets operations. For a node x, we simply define ${}^\bullet x$ and $x^\bullet$ as the same sets as above, with multiplicity 1 for each element. For a transition t, we define ${}^\diamond t$ as the same set as above, with multiplicity ∞ for each element. A marking M enables a transition t if $\forall p \in {}^\bullet t, |M|_p > 0$. In this case, firing t from M leads to the new marking $M' = (M \ominus ({}^\bullet t \oplus {}^\diamond t)) \oplus t^\bullet$.

Among unsafe nets, we distinguish two classes, based on the markings that are reachable. *Unbounded* nets are those nets where there exists at least one place p such that there is no bound $k \in \mathbb{N}$ verifying $k \geq |M|_p$ for all reachable M. In other words, they are the nets with infinite number of reachable markings. *Bounded* nets are the other nets: those where such a bound k can be found.

In the rest of this section we extend to unbounded reset Petri nets the notions of Pomset bisimulation and unfolding. Then, we extend our other results to bounded reset Petri nets.

6.2. Unbounded reset Petri nets

6.2.1. Pomset bisimulation for unbounded reset Petri nets

We first extend the definition of pomset bisimulation for reset Petri nets to unbounded reset Petri nets. As before, this implies to define a notion of process for unbounded reset Petri nets. This notion is in fact closely related to the one defined in Section 3.

The first step to define processes is to define occurrence nets. As this notion does not depend on the markings, one can keep it as is (notice that our occurence nets thus remain safe nets: markings are sets). Then, one needs a notion of homomorphism, to link Petri nets and occurrence nets representing the processes. The notion of homomorphism needs to be modified a little bit, as it does not yet take into account the possibility for an initial marking to be a multiset. The idea is to enforce building a copy of each initially marked place for each occurrence of it in the multiset representing the initial marking. This gives the following new definition

Definition 24 (updated homomorphism of nets)**.** *Let* $\mathcal{N}_1$ *and* $\mathcal{N}_2$ *be two Petri nets such that* $\mathcal{N}_i = (P_i, T_i, F_i, \emptyset, M_{0,i})$. *A mapping* $h : P_1 \cup T_1 \to P_2 \cup T_2$ *is an homomorphism of nets from* $\mathcal{N}_1$ *to* $\mathcal{N}_2$ *if* $\forall p_1 \in P_1, \forall p_2 \in P_2, \forall t \in T_1$*: (1)* $h(p_1) \in P_2$*, (2)* $h(t) \in T_2$*, (3)* $p_2 \in {}^\bullet h(t) \Leftrightarrow \exists p_1' \in {}^\bullet t, h(p_1') = p_2$*, (4)* $p_2 \in h(t)^\bullet \Leftrightarrow \exists p_1' \in t^\bullet, h(p_1') = p_2$*, (5)* $|M_{0,2}|_{p_2} = k \Leftrightarrow |\{p_1' \in M_{0,1} \ : \ h(p_1') = p_2\}| = k$.

From that, the notion of processes of a Petri net (without reset arcs) is defined exactly as before, simply replacing the previous notion of homomorphism

with the new one. The notion of potential processes of a reset Petri net as well as the notion of processes of a reset Petri net are then similarly derived from it.

Finally, the notions of pomset abstraction and pomset bisimulation of unbounded reset Petri nets have the same definition as before. Simply notice that the new definition of homomorphism is used, in order to deal with markings that are not safe (this is needed, in particular, in (2) of Definition 16).

6.2.2. Unfolding of unbounded reset Petri nets

As above, the unfolding can be defined from the branching processes of a reset Petri net in the unbounded case. We illustrate it on an example (the formal definition remains the same). For that, consider the net of Figure 9. It has been obtained by adding a token generator to the example of Figure 7. It is clearly unbounded, t_6 acting as a perpetual token provider for the place p_5.

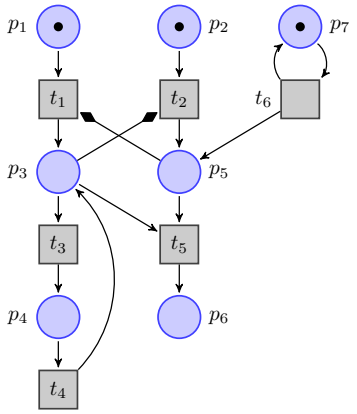


Figure 9: An unbounded reset Petri net

The unfolding of the underlying net is represented in Figure 10. This unfolding is infinite, so only two occurrences of t_6 and two occurrences of t_3 are represented. The token provider appears in the rigth part of the figure. Dashed arcs represent the control of all the occurrences of the transition t_5 by all the conditions b_i such as $\lambda(b_i) = p_3$.

The Figure 11 shows the adding of reset arcs.

6.3. Bounded reset Petri nets

We now extend to bounded reset Petri nets the structural translation preserving bisimulation given for safe reset Petri nets in definition 17 as well as our technique for computing finite complete prefixes.

In all this section we assume that the nets we consider are *k-bounded*, that is bounded with a known bound k.

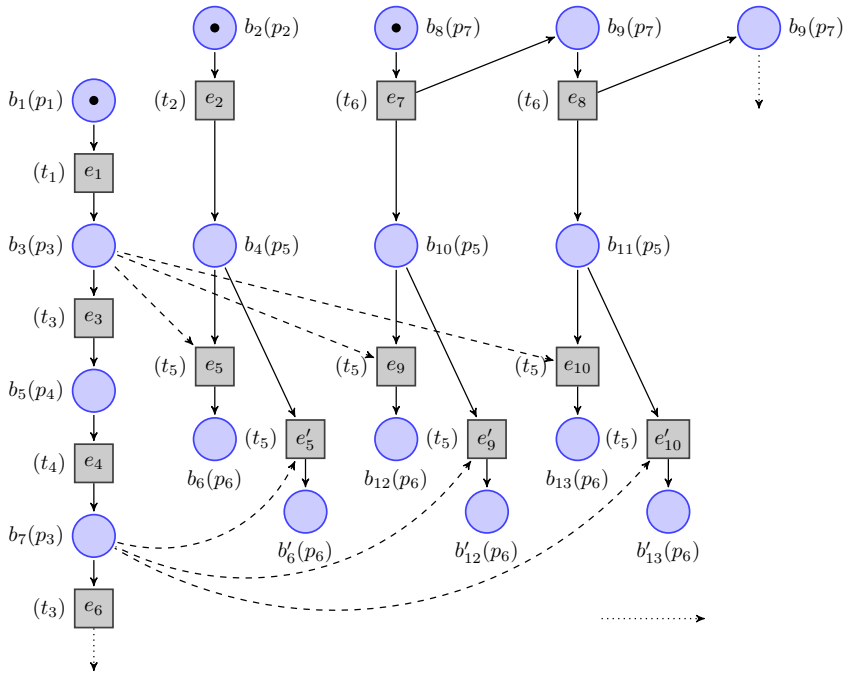


Figure 10: Infinite ufolding without reset arcs

6.3.1. Strutural transformation for a bounded reset Petri net

For each reset arc (p, t), a copy $\overline{p}$ of p is created and the transition t is translated into $k+1$ transitions $t_0 \dots t_k$. The two places are so that $M(p) + M(\overline{p}) = k$ (recall that the net is k-bounded) and the transition t_n will perform the reset when the number of token in p is n. For that, new arcs are added to F (as shown in figure 12) such that if t is fireable in the initial net, one and only one copy of t is firable in the translated net : if the number of tokens in p is n then the number of tokens in $\overline{p}$ is $k-n$ and only the transition t_n is fireable and its firing leads to a marking such that $M(p) = 0$ and $M(\overline{p}) = k$ i.e. the reset is performed.

This translation uses weighted arcs that are not allowed in our class of Petri net but it is well known that for every k-bounded Petri Net with weighted arcs, there is a safe Petri Net without weighted arcs which, if not pomset bisimilar, has the same interleaving behaviour and there exists well known straightforward approaches [23, 24] performing this translation.

For the sake of conciseness, we will keep the weighted arcs in our translation and, in the following paragraph, for a transition (p, t), resp. (t, p), with a weight n, we use the notation $(p, t)^n$, resp. $(t, p)^n$.

Definition 25 (Structural transformation)**.** *Let $(\mathcal{N}_R, \Sigma, \lambda)$ be a k-bounded labelled reset Petri net such that $\mathcal{N}_R = (P, T, F, R, M_0)$, its structural transfor-*

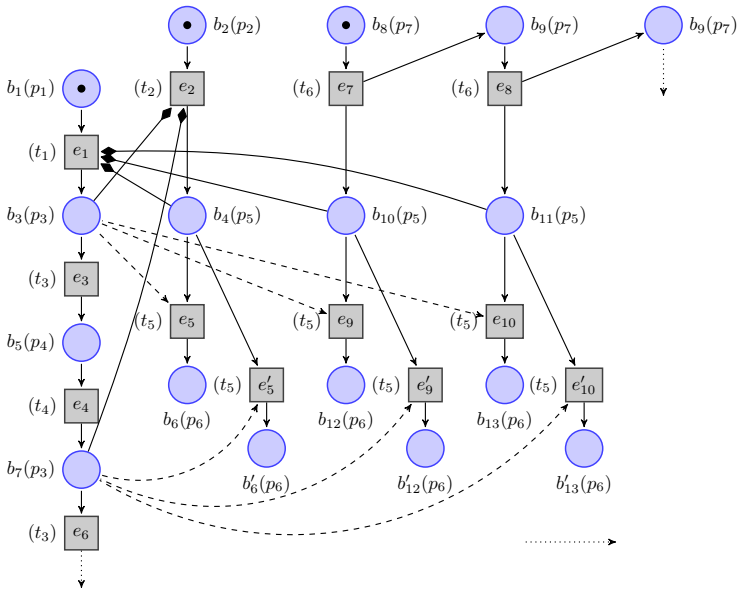


Figure 11: Addition of reset arcs to the infinite unfolding

mation is the labelled Petri net $(\mathcal{N}_{R,str}, \Sigma_{str}, \lambda_{str})$ *which is such that* $\mathcal{N}_{R,str} = (P_{str}, T_{str}, F_{str}, \emptyset, M_{0,str})$ *with:*

$$P_{str} = P \cup \overline{P} \text{ with } \overline{P} = \{\overline{p} \ : \ p \in P \wedge \exists t \in T, (p,t) \in R\},$$

$$T_{str} = T \cup \left(\bigcup_{n=1}^{k} \overline{T_n} \right) \quad \text{with } \overline{T_n} = \{\overline{t_n} \ : \ t \in T \wedge {}^\diamond t \neq \emptyset\},$$

$$F_{str} = F \cup \left(\bigcup_{n=1}^{k} \{(p, \overline{t_n}) : \overline{t_n} \in \overline{T_n}, (p,t) \in F\} \cup \{(\overline{t_n}, p) : \overline{t_n} \in \overline{T_n}, (t,p) \in F\} \right)$$

$$\cup \{(\overline{p}, t) \ : \ \overline{p} \in \overline{P}, (t,p) \in F\} \cup \{(t, \overline{p}) \ : \ \overline{p} \in \overline{P}, (p,t) \in F\}$$

$$\cup \left(\bigcup_{n=1}^{k} \{(\overline{p}, \overline{t_n}) \in \overline{P} \times \overline{T_n} \ : \ (t,p) \in F\} \cup \{(\overline{t_n}, \overline{p}) \in \overline{T_n} \times \overline{P} \ : \ (p,t) \in F\} \right)$$

$$\cup \{(p,t)^k, (t, \overline{p})^k, (\overline{p}, \overline{t_k})^k, (\overline{t_k}, \overline{p})^k \ : \ (p,t) \in R\}$$

$$\cup \left(\bigcup_{n=1}^{k-1} \{(p, \overline{t_n})^{k-n}, (\overline{p}, \overline{t_n})^n, (\overline{t_n}, \overline{p})^k\} \right)$$

$$M_{0,str} = M_0 \cup \{\overline{p} \in \overline{P} \ : \ p \notin M_0\},$$

and moreover, $\Sigma_{str} = \Sigma$, $\forall t \in T, \lambda_{str}(t) = \lambda(t)$, *and* $\forall \overline{t_n} \in \overline{T_n}, \lambda_{str}(\overline{t_n}) = \lambda(t)$.

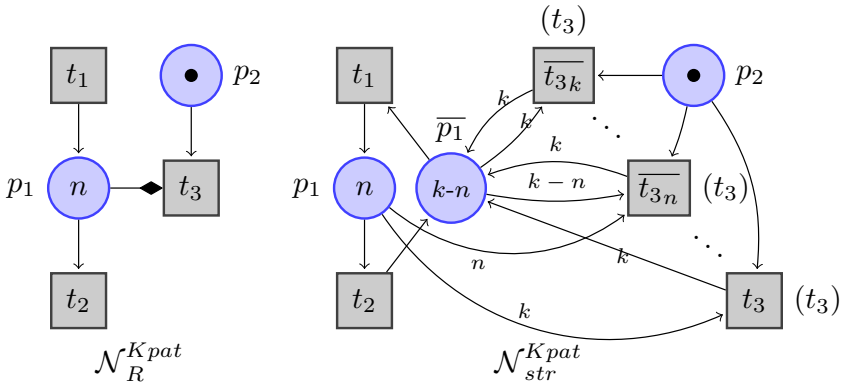


Figure 12: Translating reset arcs for bounded nets

Lemma 3. *A k-bounded labelled reset Petri net* $(\mathcal{N}_R, \Sigma, \lambda)$ *and its structural transformation* $(\mathcal{N}_{R,str}, \Sigma_{str}, \lambda_{str})$ *as defined in Definition 25 are bisimilar.*

Proof. (Sketch) The bisimulation relation is $\rho \subseteq [\mathcal{N}_R\rangle \times [\mathcal{N}_{R,str}\rangle$ defined by $(M, M_{struct}) \in \rho$ iff $\forall p \in P, |M|_p = |M_{struct}|_p$ and $\forall p \in P$ such that $\overline{p} \in \overline{P}$, we have $|M_{struct}|_p + |M_{struct}|_{\overline{p}} = k$. We give the sketch of the proof on the pattern $\mathcal{N}_R^{Kpat}$ of figure 12 assuming that the net that includes the pattern is k-bounded, making the pattern also k-bounded. The proof can easily be generalised to any k-bounded labelled reset Petri nets. Applying the translation to the reset Petri net $\mathcal{N}_R^{Kpat}$ on the left gives the Petri net $\mathcal{N}_{str}^{Kpat}$.

Assume that $(M^1, M^1_{struct}) \in \rho$. If $|M^1|_{p_1} = |M^1_{struct}|_{p_1} = n < k$ then $|M^1_{struct}|_{\overline{p_1}} = k - n$ and the firing of t_1 leads respectively to M^2 and M^2_{struct} such that $|M^2|_{p_1} = |M^2_{struct}|_{p_1} = n + 1$ and $|M^2_{struct}|_{\overline{p_1}} = k - (n + 1)$ and $(M^2, M^2_{struct}) \in \rho$. If $|M^1|_{p_1} = |M^1_{struct}|_{p_1} = n > 0$ then $M^1_{struct}(\overline{p_1}) = k - n$ and the firing of t_2 leads respectively to M^3 and M^3_{struct} such that $|M^3|_{p_1} = |M^3_{struct}|_{p_1} = n - 1$ and $|M^3_{struct}|_{\overline{p_1}} = k - (n - 1)$ and $(M^3, M^3_{struct}) \in \rho$.

Now assume $|M^1|_{p_2} = |M^1_{struct}|_{p_2} \geq 1$ and $|M^1|_{p_1} = |M^1_{struct}|_{p_1} = n$ then $M^1_{struct}(\overline{p_1}) = k - n$. The transition t_3 is fireable in the initial net and, in the translated net, the only fireable transition $\overline{t_i}$ (with $i \in \{0 \dots n\}$), such that $\lambda_{struct}(\overline{t_i}) = t_3$ is $\overline{t_n}$. The firing of t_3 (resp. $\overline{t_n}$) leads to M^4 (resp. M^4_{struct}) such that $|M^4|_{p_1} = |M^4_{struct}|_{p_1} = 0$ and $|M^4_{struct}|_{\overline{p_1}} = k$. Moreover $|M^4|_{p_2} = |M^4_{struct}|_{p_2} = |M^1|_{p_2} - 1$ then $(M^4, M^4_{struct}) \in \rho$ □

6.3.2. Finite complete prefixes of Bounded reset Petri nets

As the notion of unfolding, the notion of finite complete prefix naturally extends to bounded Petri nets. Using the same notations as for safe nets, our finite complete prefix construction can be extended to bounded reset Petri nets as follows. We consider k-bounded reset Petri nets whose underlying Petri nets are also k-bounded. This assumption can be lifted by using an on the fly

construction (merging step 3 and 4 of Algorithm 2) that will not be discussed in this paper.

Algorithm 2: Finite complete prefix for bounded reset Petri nets

Data: a k-bounded reset Petri net $\mathcal{N}_R^k$
Result: the finite complete prefix $\mathcal{F}_R^k$ for $\mathcal{N}_R^k$
Step 1: compute the structural transformation $\mathcal{N}_{R,str}^k$ of $\mathcal{N}_R^k$ (according to Definition 25)
Step 2: compute a finite complete prefix $\mathcal{F}_{str}^k$ of the unfolding $\mathcal{U}_{str}^k$ of $\mathcal{N}_{R,str}^k$
Step 3: compute a finite prefix $\mathcal{F}^k$ of $\mathcal{U}^k$ that simulates $\mathcal{F}_{str}^k$
Step 4: compute $\mathcal{F}_R^k$ by adding reset arcs to $\mathcal{F}^k$

The following proposition establishes the computability and the completeness of $\mathcal{F}_R^k$. The finiteness is granted by step 3 of the above algorithm.

Proposition 7. *The object $\mathcal{F}_R^k$ can be computed by the Algorithm 2 from a k-bounded reset Petri net $\mathcal{N}_R^k$ and is a complete prefix of the unfolding of $\mathcal{N}_R^k$.*

Proof. We first show that $\mathcal{F}_R^k$ is computable. The feasability of step 1 is granted by the definition of the structural transformation. Then, one can notice that, if a net is k-bounded, its transformation is bounded as well, so it exists at least one finite complete prefix of its unfolding (which is computable by standard techniques), thus step 2 is feasible as well. In step 3 we only require a simulation (not a bisimulation) which makes it straightforward to build $\mathcal{F}^k$ simply by a standard unfolding procedure that would stop as soon as a part of $\mathcal{U}^k$ that does not simulates $\mathcal{F}_{str}^k$ is reached (recall that $\mathcal{U}^k$ simulates $\mathcal{U}_{str}^k$). Finally step 4 simply constists in the addition of a finite number of reset arcs (because $\mathcal{F}^k$ is finite). This grants the computability of $\mathcal{F}_R^k$.

The completeness of $\mathcal{F}_R^k$ is directly obtained from the completeness of $\mathcal{F}_{str}^k$ and the fact that $\mathcal{F}_R^k$ simulates $\mathcal{F}_{str}^k$.

□

7. Conclusion

Our contribution in this paper is four-fold. First, we have proposed a notion of pomset bisimulation for reset Petri nets. This notion is, in particular, inspired from a similar notion that has been defined for Petri nets (without reset arcs) in [19]: it extends this notion, in the sens that, for Petri nets without reset arcs our notion of pomset bisimulation corresponds to the one of [19]. Second, we have shown that it is not possible to remove reset arcs from safe reset Petri nets while preserving their behaviours with respect to this pomset bisimulation. Third, we proposed a notion of finite complete prefixes of unfolding of safe reset Petri nets that allows for reachability analysis while preserving pomset

behaviour. As a consequence of the two other contributions, these finite complete prefixes do have reset arcs. Last, we have extended the previous results to unsafe reset Petri nets.

References

[1] P. Baldan, A. Corradini, U. Montanari, Contextual Petri nets, asymmetric event structures and processes, Information and Computation 171 (1) (2001) 1–49.

[2] U. Montanari, F. Rossi, Contextual nets, Acta Inf. 32 (6) (1995) 545–596.

[3] Y. Chen, Z. Li, K. Barkaoui, N. Wu, M. Zhou, Compact supervisory control of discrete event systems by petri nets with data inhibitor arcs, IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (2016) 1–16.

[4] T. Araki, T. Kasami, Some undecidable problems for Petri nets, Systems, Computers, Controls 7 (1) (1976) 20–28.

[5] P. Godefroid, Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem, Vol. 1032 of Lecture Notes in Computer Science, Springer, 1996.

[6] J. Esparza, K. Heljanko, Unfoldings – A Partial-Order Approach to Model Checking, Springer, 2008.

[7] J. Esparza, S. Römer, W. Vogler, An improvement of McMillan's unfolding algorithm, Formal Methods in System Design 20 (3) (2002) 285–310.

[8] K. L. McMillan, Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, in: CAV, 1993, pp. 164–177.

[9] S. Hickmott, J. Rintanen, S. Thiébaux, L. White, Planning via Petri net unfolding, in: IJCAI, 2007, pp. 1904–1911.

[10] A. Benveniste, E. Fabre, S. Haar, C. Jard, Diagnosis of asynchronous discrete-event systems: a net unfolding approach, IEEE TAC 48 (5) (2003) 714–727.

[11] M. Wang, G. Liu, P. Zhao, C. Yan, C. Jiang, Behavior consistency computation for workflow nets with unknown correspondence, IEEE/CAA Journal of Automatica Sinica 5 (1) (2018) 281–291.

[12] M. Nielsen, G. D. Plotkin, G. Winskel, Petri nets, event structures and domains, part I, Theoretical Computer Science 13 (1981) 85–108.

[13] U. Goltz, W. Reisig, The non-sequential behavior of Petri nets, Information and Control 57 (2/3) (1983) 125–147.

[14] E. Best, R. Devillers, Sequential and concurrent behaviour in Petri net theory, Theor. Comput. Sci. 55 (1) (1987) 87–136.

[15] P. Baldan, A. Corradini, U. Montanari, Contextual Petri nets, asymmetric event structures, and processes, Information and Computation 171 (1) (2001) 1–49.

[16] N. Busi, G. M. Pinna, Non sequential semantics for contextual P/T nets, in: Application and Theory of Petri Nets, Vol. 1091 of Lecture Notes in Computer Science, Springer, 1996, pp. 113–132.

[17] J. Winkowski, Processes of contextual nets and their characteristics, Fundamenta Informaticae 36 (1) (1998).

[18] W. Vogler, Partial order semantics and read arcs, Theoretical Computer Science 286 (1) (2002) 33–63.

[19] E. Best, R. R. Devillers, A. Kiehn, L. Pomello, Concurrent bisimulations in Petri nets, Acta Inf. 28 (3) (1991) 231–264.

[20] R. J. van Glabbeek, U. Goltz, Equivalence notions for concurrent systems and refinement of actions, in: MFCS, 1989, pp. 237–248.

[21] W. Vogler, Bisimulation and action refinement, Theor. Comput. Sci. 114 (1) (1993) 173–200.

[22] R. J. van Glabbeek, F. W. Vaandrager, Petri net models for algebraic theories of concurrency, in: PARLE, 1987, pp. 224–242.

[23] Best, Eike, In quest of a morphism, Petri Nets Newsletters 18 (1984) 14–18.

[24] Pelz, E., Normalization of place/transition-systems preserves net behaviour, RAIRO-Theor. Inf. Appl. 26 (1) (1992) 19–44.