Pomsets and Unfolding of Reset Petri Nets

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

Pomsets and Unfolding of Reset Petri Nets

Thomas Chatain[1], Maurice Comlan[2], David Delfieu[3], Loïg Jezequel[3], and Olivier H. Roux[3]

[1] LSV – ENS Cachan, France
[2] Université d'Abomey-Calavi, LETIA, Bénin
[3] 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 such 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 to the class of safe reset Petri nets.

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 [3, 11] (allowing to read variables values without modifying them), and reset arcs [1] (allowing to modify variables values independently of their previous value). Reset arcs increase the expressiveness of Petri nets, but they compromise analysis techniques. For example, boundedness [6] and reachability [1] are undecidable. For bounded reset Petri nets, more properties are decidable, as full state spaces can be computed.

Full state-space computations (i.e. using state graphs) do not preserve partial order semantics. To face this problem, Petri nets unfolding has been proposed and has gained the interest of researchers in verification [7], diagnosis [4], and planning [9]. This technique keeps the intrinsic parallelism and prevents the combinatorial interleaving of independent events. While the unfolding of a Petri net can be infinite, there exist algorithms for constructing finite prefixes of it [8, 10]. Unfolding have the strong interest of preserving more behavioral properties of Petri nets than state graphs. In particular they preserve concurrency and its counterpart: causality. Unfolding techniques have also been developed for extensions of Petri nets, and in particular Petri nets with read arcs [2].

Our contribution: Reachability analysis is known to be feasible on bounded reset Petri nets, however, as far as we know, no technique for computing finite

prefixes of unfolding exists yet, and so, no technique preserving concurrency and causality exists yet. This is the aim of this paper to propose one. 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 [5, 12, 14]. From this characterization we can then express what should be an unfolding preserving 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.

This paper is organized as follows: We first give basic definitions and notations for (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 resets which is pomset bisimilar to a given reset Petri net. Finally, in Section 5 – building on the results of Section 4 – we propose a finite complete prefix construction for 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 $n \geq 2$ so 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 $\forall i \in [1..m-1], (y_i, y_{i+1}) \in F$, so that $x_1 = y_1$ is a place, $x_2 \neq y_2$, $x_n = x$, and $y_m = y$.

A *marking* is a set $M \subseteq P$ of places. It *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* (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 so 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 $[\mathcal{N}_R\rangle$.

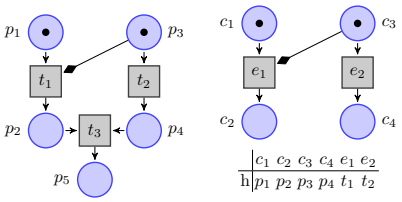

Fig. 1. A reset Petri net (left) and one of its processes (right)

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 $(t^\bullet \setminus ({}^\bullet t \cup {}^\diamond t)) \cap M = \emptyset$.*

The reader familiar with Petri nets will notice that our results generalize to larger classes of nets: unbounded reset Petri nets for our pomset bisimulation (Section 3), and bounded reset Petri nets for our prefix construction (Section 5).

In the rest of the paper, unless the converse is specified, we consider reset Petri nets so 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 so 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 will be 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 $(\mathcal{N}_R, \Sigma, \lambda)$ so 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*.*

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, $\lambda(t_1 \dots t_n) = \lambda(t_1)\lambda(t_2 \dots t_n)$ if $\lambda(t_1) \in \Sigma$ and $\lambda(t_1 \dots t_n) = \lambda(t_2 \dots t_n)$ else (that is if $\lambda(t_1) = \varepsilon$). 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 and only if there exists a relation* $\rho \subseteq [\mathcal{N}_{R,1}\rangle \times [\mathcal{N}_{R,2}\rangle$ *(a* bisimulation*) so that:*

1. $(M_{0,1}, M_{0,2}) \in \rho$,
2. *if* $(M_1, M_2) \in \rho$, *then*
   (a) *for every transition* $t \in T_1$ *so 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* $\mathcal{N}_{R,2}$ *so 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* $t \in T_2 \dots$*)*

This bisimulation however hides an important part of the behaviours of (reset) Petri nets: transition firings may be *concurrent* when transitions are not in causal relation nor in conflict. 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 not in causal relation while in $\mathcal{N}_{R,2}$ they are in causal relation.

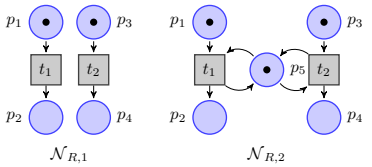

Fig. 2. Two bisimilar nets

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.

Definition 7 (occurrence net). *An* occurrence net *is a (reset) Petri net* $(B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, M_0^{\mathcal{O}})$ *so 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 and only if* ${}^\bullet b = \emptyset$*.*

Places of an occurrence net are usually referred to as *conditions* and transitions as *events*. In an occurrence net, if two nodes $x, y \in B \cup E$ are so that $x \neq y$, are

not in causal relation, and are not in conflict, they are said to be *concurrent*. Moreover, in 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 so 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) Graph 3-coloring reduces to executability of an occurrence net.

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 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) $p_2 \in M_{0,2} \Leftrightarrow \exists p'_1 \in M_{0,1}, h(p'_1) = p_2$.*

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 an 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 $\mathcal{N}$.*

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.

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 an 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 and only if $(h(b), h(e)) \in R$.*

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 an 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 $\exists M_1, \ldots, M_n \subseteq B$ so 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 $e_1 e_2 \dots e_n$ (1) that contains all the events in E and (2) such that $\forall i, j \in [1..n]$ if $e_i \prec e_j$ then $i < j$. Necessarily, there exist markings $M_1, \dots, M_n$ so that $M_0^{\mathcal{O}}[e_1\rangle M_1[e_2\rangle \dots [e_n\rangle M_n$.

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 $e_2 \prec e_1$, their should exist markings M_1, M_2 so 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 $e_1 e_2 \dots e_n$ 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 somehow 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 $E = \{e_1, \dots, e_n\}$ 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 so that $\exists M_1, \dots, M_{n-1}$, verifying $M_0^{\mathcal{O}}[e_{k_1}\rangle M_1[e_{k_2}\rangle \dots M_{n-1}[e_{k_n}\rangle M$ for some $\{k_1, \dots, k_n\} = \{1, \dots, n\}$.*

In other words, the maximal markings of a process are all the marking 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 so that $\forall e \in E', \lambda'(e) = \lambda(h(e))$. Define moreover $< \ \subseteq E' \times E'$ as the relation so that $e_1 < e_2$ if and only 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 and only if there exists a bijection $f : E \to E'$ so that $\forall e_1, e_2 \in E$: (1) $\lambda(e_1) = \lambda'(f(e_1))$, and (2) $e_1 < e_2$ if and only 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 so that $\forall e \in E, \lambda'(e) = \lambda(h(e))$. The* linear abstraction *of $\mathcal{P}$ with respect to M is the set $lin(M, \mathcal{P})$ so that a sequence of labels ω is in $lin(M, \mathcal{P})$ if and only if in $\mathcal{O}$ there exist markings $M_1, \ldots, M_{n-1}$ and events $e_1, \ldots, e_n$ so 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 [5, 12, 14]. Intuitively, two reset Petri nets are pomset bisimilar if there exists a relation between their reachable markings so that the markings that can be reached by pomset equivalent processes from two markings in relation are themselves in relation. This is formalized by the below definition.

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 and only if there exists a relation $\rho \subseteq [\mathcal{N}_{R,1}\rangle \times [\mathcal{N}_{R,2}\rangle$ (called a* pomset bisimulation*) so 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 $(P_2, T_2, F_2, R_2, M_2)$ so that $\mathcal{P}_1 \equiv \mathcal{P}_2$ and*
      - $\forall M_1' \in M_{max}(\mathcal{P}_1), \exists M_2' \in M_{max}(\mathcal{P}_2)$ *so 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 $\mathcal{P}_2 \ldots$)*

Notice that, in the above definition, taking the processes $\mathcal{P}_1$ and $\mathcal{P}_2$ bisimilar (using the standard bisimulation relation for Petri nets) rather than comparing $lin(M_1', \mathcal{P}_1)$ and $lin(M_2', \mathcal{P}_2)$ would lead to an equivalent definition.

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

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 that (reset) Petri nets are finite, i.e. their sets of places and transitions are finite.

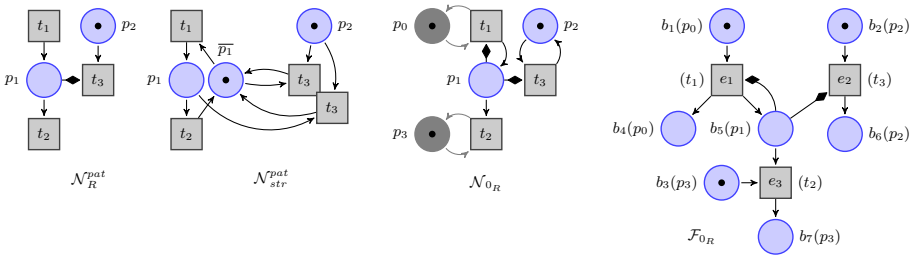

Fig. 3. A remarkable pattern $\mathcal{N}_R^{pat}$ and its structural transformation $\mathcal{N}_{str}^{pat}$, a labelled reset Petri net $\mathcal{N}_{0_R}$ including the pattern $\mathcal{N}_R$, and a finite complete prefix $\mathcal{F}_{0_R}$ of $\mathcal{N}_{0_R}$. Transition labels are given on 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 with respect to this pomset bisimulation. More precisely, we prove that it is not possible to build a safe labelled 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) without reset arcs which is pomset bisimilar to a given safe labelled reset Petri net. For that, we exhibit a particular pattern – Figure 3 (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 $\mathcal{N}_{R,str} = (P_{str}, T_{str}, F_{str}, \emptyset, M_{0,str})$ so that:*

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

$$T_{str} = T \cup \overline{T} \text{ with } \overline{T} = \{\overline{t} \ : \ t \in T \wedge {}^\diamond t \neq \emptyset\},$$

$$F_{str} = F \cup \{(p,\overline{t}) \ : \ \overline{t} \in \overline{T}, (p,t) \in F\} \cup \{(\overline{t},p) \ : \ \overline{t} \in \overline{T}, (t,p) \in F\} \quad (1)$$

$$\cup \{(\overline{p},t) \ : \ \overline{p} \in \overline{P}, (t,p) \in F\} \cup \{(t,\overline{p}) \ : \ \overline{p} \in \overline{P}, (p,t) \in F\} \quad (2)$$

$$\cup \{(\overline{p},\overline{t}) \in \overline{P} \times \overline{T} \ : \ (t,p) \in F\} \cup \{(\overline{t},\overline{p}) \in \overline{T} \times \overline{P} \ : \ (p,t) \in F\} \quad (3)$$

$$\cup \{(p,t),(\overline{p},\overline{t}),(t,\overline{p}),(\overline{t},\overline{p}) \ : \ (p,t) \in R\}, \quad (4)$$

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

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

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 so 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 so 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 3 (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. (Sketch) The bisimulation relation is $\rho \subseteq [\mathcal{N}_{R,1}\rangle \times [\mathcal{N}_{R,2}\rangle$ 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 $M_{struct}(p) + M_{struct}(\overline{p}) = 1$.

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 3. 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 which 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 3 (middle right). It uses the pattern $\mathcal{N}_R^{pat}$ of Figure 3 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.

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

Proof. We simply remark that 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.

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. (Sketch) 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.

5 Finite complete prefixes of unfolding of reset Petri nets

In this section, we propose a notion of finite complete prefixes of unfolding of safe reset Petri nets preserving reachability of markings and pomset behaviour. As a consequence of the previous section, 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 [7] or equivalently 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 $\mathcal{N}_R$ be a safe reset Petri net and $\mathcal{N}$ be its underlying Petri net. Let $\mathcal{U}$ be the unfolding of $\mathcal{N}$. The unfolding of $\mathcal{N}_R$ is $\mathcal{U}_R$, obtained by adding reset arcs to $\mathcal{U}$ according to (2) in Definition 10.*

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 [13], 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).

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 [10, 8].

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 processe $(\mathcal{O}, h)$ of $\mathcal{N}$ verifying the following property of* reachable marking preservation*: a marking M is reachable in $\mathcal{N}$ if and only if there exists a reachable marking M' in $\mathcal{O}$ so that $M = \{h(b) \ : \ b \in M'\}$.*

In this section, we propose an algorithm for construction of finite complete prefixes for safe reset Petri nets. For that, we assume the existence of a black-box algorithm for building finite complete prefixes of safe Petri nets (without reset arcs). Notice that such algorithms indeed do exist [10, 8].

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 $\mathcal{N}_R$).*

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 to compute finite prefixes of reset Petri nets from their structural transformations: in the below algorithm, $\mathcal{F}_{str}$ 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 construction for reset Petri nets) *Let $\mathcal{N}_R$ be a safe reset Petri net, (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. The output of the algorithm is $\mathcal{F}_R$.*

Applying this algorithm to the net $\mathcal{N}_{0_R}$ of Figure 3 (middle right) – using the algorithm from [8] at step 2 – leads to the reset Petri net $\mathcal{F}_{0_R}$ of Figure 3 (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 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 Conclusion

Our contribution in this paper is three-fold. First, we 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 [5]. 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. And, 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.

References

1. T. Araki and T. Kasami. Some decision problems related to the reachability problem for Petri nets. *Theor. Comput. Sci.*, 3(1):85–104, 1976.
2. P. Baldan, A. Bruni, A. Corradini, B. König, C. Rodríguez, and S. Schwoon. Efficient unfolding of contextual Petri nets. *Theor. Comput. Sci.*, 449:2–22, 2012.
3. P. Baldan, A. Corradini, and U. Montanari. Contextual Petri nets, asymmetric event structures and processes. *Information and Computation*, 171(1):1–49, 2001.
4. A. Benveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete-event systems: a net unfolding approach. *IEEE TAC*, 48(5):714–727, 2003.
5. E. Best, R. R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri nets. *Acta Inf.*, 28(3):231–264, 1991.
6. C. Dufourd, P. Schnoebelen, and P. Jančar. Boundedness of reset P/T nets. In *ICALP*, pages 301–310, 1999.
7. J. Esparza and K. Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. Springer, 2008.
8. J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
9. S. Hickmott, J. Rintanen, S. Thiébaux, and L. White. Planning via Petri net unfolding. In *IJCAI*, pages 1904–1911, 2007.
10. K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *CAV*, pages 164–177, 1993.
11. U. Montanari and F. Rossi. Contextual nets. *Acta Inf.*, 32(6):545–596, 1995.
12. R. J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *MFCS*, pages 237–248, 1989.
13. R. J. van Glabbeek and F. W. Vaandrager. Petri net models for algebraic theories of concurrency. In *PARLE*, pages 224–242, 1987.
14. W. Vogler. Bisimulation and action refinement. *Theor. Comput. Sci.*, 114(1):173–200, 1993.