Pomsets and Unfolding of Reset Petri Nets

Maurice Comlan[1], Thomas Chatain[2], David Delfieu[3], Loïg Jezequel[4], and Olivier H. Roux[5]

1 **Université d'Abomey-Calavi, LETIA, Bénin** – comlan@hotmail.fr
2 **LSV – ENS Cachan, France** – chatain@lsv.fr
3 **Université de Nantes, LS2N UMR 6004, France** – david.delfieu@ls2n.fr
4 **Université de Nantes, LS2N UMR 6004, France** – loig.jezequel@ls2n.fr
5 **École Centrale de Nantes, LS2N UMR 6004, France** – olivier-h.roux@ls2n.fr

Abstract

Reset Petri nets are a particular class of Petri nets where a transition firing can remove all tokens from a place without having to check 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 notion of pomset bisimulation for comparing concurrent behaviours of reset Petri nets. 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.

1998 ACM Subject Classification D.2.2 Design Tools and Techniques

Keywords and phrases Petri nets, Reset arcs, Unfolding, Pomset bisimulation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2017.23

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 [**?**, **?**] (allowing to read variables values without modifying them), and reset arcs [**?**] (allowing to modify variables values independently of their previous value).

While reset arcs increase the expressiveness of Petri nets, they compromise analysis techniques. Some properties become undecidable in general, such as boundedness [**?**] and reachability [**?**]. Some remain decidable but require to extend analysis techniques as this is the case for coverability [**?**]. For bounded reset Petri nets however, more properties are decidable, because full state spaces can be computed. This is in particular the case of reachability.

Reachability analysis based on full state-space computation (i.e. using state graphs) is however restricted due to combinatorial explosion. To face this problem, partial order techniques have been proposed, and, in particular, Petri nets unfolding [**?**]. It keeps the intrinsic parallelism and prevents the combinatorial interleaving of independent events. Petri nets unfolding has gained the interest of researchers in verification [**?**], diagnosis [**?**], and planning [**?**]. While the unfolding of a Petri net can be infinite, there exist algorithms for constructing finite prefixes of it, sufficient for reachability analysis [**?**, **?**]. Unfolding

techniques have also been developed for extensions of Petri nets, and in particular Petri nets with read arcs [**?**].

If reachability analysis is known to be feasible on bounded reset Petri nets, as far as we know no technique for computing finite prefixes of unfolding 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 [**?**, **?**, **?**]. From this characterization we can then express what should be an unfolding preserving the concurrent behaviour of a reset Petri net.

This paper is organized as follows: We first give basic definitions and notations for (safe) reset Petri nets. Then, in Section **??**, we propose the definition of a notion of pomset bisimulation for reset Petri nets. In Section **??** 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 **??** – building on the results of Section **??** – we propose a finite complete prefix construction for reset Petri nets where the prefix is itself a reset Petri net.

2 Reset Petri nets

2.1 Basic definitions

▶ **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 **??** (left) is a graphical representation of a reset Petri net. It has five places (p_1 to p_5, represented as circles) and three transitions (t_1 to t_3, represented as squares). Its set of arcs contains seven elements ((p_1, t_1), (t_1, p_2), (p_3, t_2), (t_2, p_4), (p_2, t_3), (p_4, t_3), and (t_3, p_5), depicted by arrows) and there is one reset arc ((p_3, t_1), depicted as a line with a diamond at its extremity).

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 $[\mathcal{N}_R\rangle$.

A reset Petri net with an empty set of reset arcs ($R = \emptyset$) is simply called a *Petri net*.

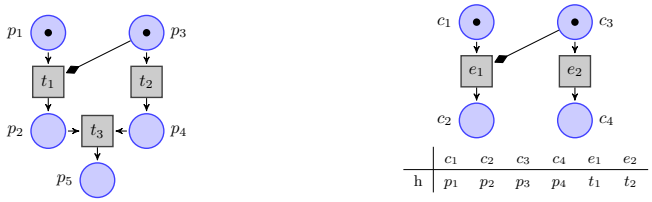

Figure 1 A reset Petri net (left) and one of its processes (right)

▶ **Definition 3** (underlying Petri net). Given a reset Petri net $\mathcal{N}_R = (P, T, F, R, M_0)$, 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 this formalism because it is sufficient 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 however notice that our results generalize to larger classes of nets, namely unbounded reset Petri nets for our pomset bisimulation definition (Section **??**), and bounded reset Petri nets for our finite complete prefix construction (Section **??**).

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 actually a restriction to our model as one can simply equip any transition t of a reset Petri net with a place p_t so that p_t is in the initial marking and ${}^\bullet p_t = p_t^\bullet = \{t\}$ to enforce the above property.

2.2 Comparing reset Petri nets

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 $t_1 \in \Sigma$ and $\lambda(t_1 \dots t_n) = \lambda(t_2 \dots t_n)$ else (that is if $t_1 = \varepsilon$).

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 $\rho \subseteq [\mathcal{N}_{R,1}\rangle \times [\mathcal{N}_{R,2}\rangle$ (called 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 arround (for every transition $t \in T_2 \dots$)

This notion of bisimulation however forgets an important part of the behaviours of (reset) Petri nets. Indeed, some transition firings may be concurrent when transitions are not in causal relation nor in conflict.

As an example, consider Figure **??**. The nets $\mathcal{N}_{R,1}$ and $\mathcal{N}_{R,2}$ are bisimilar (here we identify transition names and transition labels) and, in none of them t_1 and t_2 are in conflict. However, in $\mathcal{N}_{R,1}$ the transitions t_1 and t_2 are not in causal relation while in $\mathcal{N}_{R,2}$ they are in causal relation.

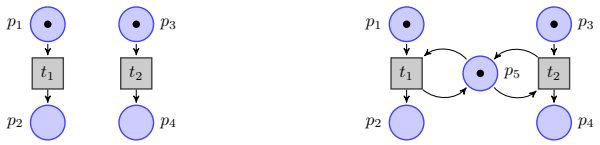

Figure 2 Two bisimilar nets $\mathcal{N}_{R,1}$ (left) and $\mathcal{N}_{R,2}$ (right)

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.

3.1 Processes of reset Petri nets

A process is a representation of a set of executions of a Petri net. It is the concurrent counterpart of paths in sequential discrete event systems (such as automata). We first recall a standard notion of processes of Petri nets and then 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 $(B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, {M^{\mathcal{O}}}_0)$ 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) $b \in {M^{\mathcal{O}}}_0$ if and only if ${}^\bullet b = \emptyset$.

Places of an occurrence net are usually referred to as *condition* 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.

▶ 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. 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 on Figure **??** from the graph composed of two vertices v_1 and v_2 connected by an edge.

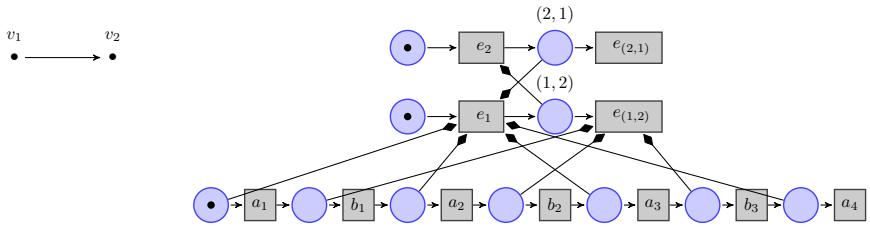

Figure 3 Reduction from 3-coloring to executability of an occurrence net. Events e_2 and $e_{(2,1)}$ reset the places of the sequence on the left, like e_1 and $e_{(1,2)}$, but these reset arcs are not represented.

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. Using additional events $a_1 \prec b_1 \prec a_2 \prec b_2 \prec a_3 \prec b_3 \prec a_4$ as bounds, one can ensure that, for 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 **??** 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 on the left, 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. ◀

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 homomorphisme 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** ((branching) processes of a Petri net)**.** Let $\mathcal{N} = (P, T, F, \emptyset, M_0)$ be a Petri net, $\mathcal{O} = (B, E, F^{\mathcal{O}}, \emptyset, {M^{\mathcal{O}}}_0)$ 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 $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, {M^{\mathcal{O}}}_0)$ 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 $\mathcal{O}' = (B, E, F^{\mathcal{O}}, \emptyset, {M^{\mathcal{O}}}_0)$, (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, $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, {M^{\mathcal{O}}}_0)$ 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 ${M^{\mathcal{O}}}_0[e_1\rangle M_1[e_2\rangle \ldots [e_n\rangle M_n$.

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 12. 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 and (2) such that $\forall i, j \in [1..n]$ if $e_i \prec e_j$ then $i < j$. Necessarily, there exists markings $M_1, \ldots, M_n$ so that ${M^{\mathcal{O}}}_0[e_1\rangle M_1[e_2\rangle \ldots [e_n\rangle M_n$.

This property (which purpose is, intuitively, to express that processes are partially ordered paths) is no longer true in processes of reset Petri nets. As an example, consider the reset Petri net of Figure **??** (left). On the right of the same figure is one of its processes (on top is the occurrence net and below is the homomorphism h). 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 13. 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 **??** 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 what is the marking reached by a process. As a corollary of the above property not holding for processes of reset Petri nets, there is no uniquely defined notion of maximal marking in these processes. Back to the above example $\{c_2\}$ is a maximal marking (no event can be fired from it), but $\{c_2, c_4\}$ is one as well (reached after firing e_2 and then e_1).

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

▶ **Definition 14** (maximal markings)**.** Let $\mathcal{P} = (\mathcal{O}, h)$ be a process with set of events $E = \{e_1, \ldots, e_n\}$ and initial marking ${M^{\mathcal{O}}}_0$ of a reset Petri net. The set $M_{max}(\mathcal{P})$ of maximal markings of $\mathcal{P}$ contains exactly the markings M so that $\exists M_1, \ldots, M_{n-1}$, and $\exists f : [1..n] \to [1..n]$ verifying ${M^{\mathcal{O}}}_0[e_{f(1)}\rangle M_1[e_{f(2)}\rangle \ldots M_{n-1}[e_{f(n)}\rangle M$.

In other words, the maximal markings of a process are all the marking that are reachable in it using all its events.

3.2 Abstracting processes

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

▶ **Definition 15** (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 $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, {M^{\mathcal{O}}}_0)$. 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 16** (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: the same multisets of labels with the same partial orderings.

Finally, we also need to be able to abstract processes as sequences of labels.

▶ **Definition 17** (linear abstraction of processes). 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 $\mathcal{O} = (B, E, F^{\mathcal{O}}, R^{\mathcal{O}}, {M^{\mathcal{O}}}_0)$, 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 ${M^{\mathcal{O}}}_0[e_1\rangle M_1[e_2\rangle \ldots M_{n-1}[e_n\rangle M$ and $\lambda'(e_1 \ldots e_n) = \omega$.

3.3 Pomset bisimulation

We now define a notion of pomset bisimulation between reset Petri nets. 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 18** (pomset 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 *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 arround (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 (this is, for example, a consequence of Lemma **??** below).

▶ 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 **??** can be obtained from Definition **??** 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.

In this section, we prove that it is, in general, 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 will notice in the below proofs that this is event 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 **??** (left) – and show that a reset Petri net including this pattern cannot be pomset bisimilar to a Petri net without reset arcs.

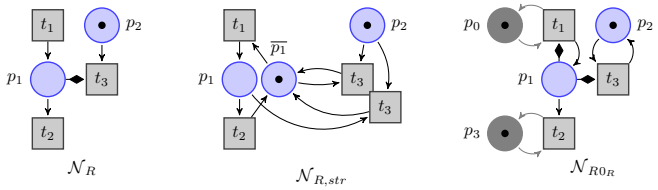

Figure 4 A remarkable pattern $\mathcal{N}_R$ and its structural transformation $\mathcal{N}_{R,str}$ (left), and a labelled reset Petri net $\mathcal{N}_{0_R}$ including the pattern $\mathcal{N}_R$ (right). Transition labels are given on transitions.

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

▶ **Definition 19** (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:

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

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

In the above definition, the reset Petri net $\mathcal{N}_R$ and the petri net $\mathcal{N}_{R,str}$ are bisimilar but not always pomset bisimilar. This can be remarked on the example of Figure **??**. Applying this translation to the reset Petri net $\mathcal{N}_R$ on the left gives the Petri net $\mathcal{N}_{R,str}$ on the middle. As we can see, this translation added 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 $\mathcal{N}_{R,str}$, 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 $\mathcal{N}_R$ 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 **??** (right). It uses the pattern $\mathcal{N}_R$ of Figure **??** 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 **??**). From that, by Proposition **??**, 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 20.** *Any 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.*

Proof. The sketch of the proof is 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. We now give the details of the 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 $t \prec t'$ so 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 $\tau^* t''$ and $\varepsilon^* t'$, with $\lambda_0(\tau^* t'') = \lambda_0(t'') = t_2$ and $\lambda_0(\varepsilon^* t') = \lambda_0(t') = t_3$, are fireable in the orders given by the sequences. Note that the set of transitions in ε^* and the set of transitions in τ^* are not necessarily disjoint. Without loss of generality, we take M, τ^*, and t'' so that the sequence $\tau^* t''$ can be fired infinitely often, that is, so that there exists a sequence of firing of transitions from the initial marking of N_0 in which the subsequence $\tau^* t''$ 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 so 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 $\tau^* t''$ and $\varepsilon^* t'$ necessarily have a direct conflict: there are a transition τ in $\tau^* t''$ and a transition ε_0 in $\varepsilon^* t'$ whose presets share at least one place. We call p'_1 such a place both in ${}^\bullet\tau$ and in ${}^\bullet\varepsilon_0$. 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 set S_{P_1} of such places p'_1, because $|{}^\bullet\tau \cap {}^\bullet\varepsilon_0| \geq 1$. 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_0 so that all the places in S_{P_1} is marked infinitely often in some infinite sequence of transitions firing in N_0 in which $\tau^* t''$ 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 $\tau^* t''$ 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 $\tau^* t''$ 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 so 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. ◀

5 Finite and complete prefixes of unfolding of reset Petri nets

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. The unfolding of a net can be defined as the union of all its branching processes [**?**] 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 **??** below we propose Definition **??**. 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 21** (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 **??**.

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

Proof. This comes directly from a result in [**?**]. It is stated in this paper that, if two Petri nets have the same unfolding (up to isomorphism), then they are pomset bisimilar (the notion of pomset bisimulation used is similar to our notion of pomset bisimulation when there are no resets in a net). 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 **??**, 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 [**?**, **?**].

▶ **Definition 22** (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 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 [**?**, **?**].

Because of Proposition **??**, 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 **??**. 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 **??** (left).

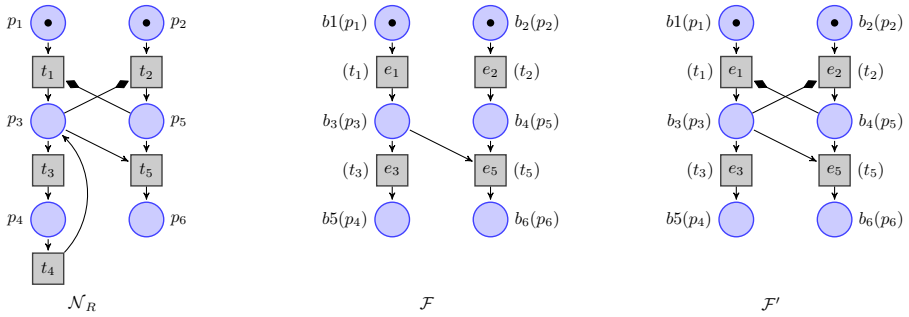

Figure 5 A reset Petri net $\mathcal{N}_R$ (left), a finite complete prefix $\mathcal{F}$ of its underlying Petri net (middle), and the same prefix after addition of reset arcs $\mathcal{F}'$ (right).

Applying the prefix construction procedure of [**?**] or [**?**] on the unfolding $\mathcal{U}$ of its underlying Petri net leads to the finite prefix $\mathcal{F}$ of Figure **??** (middle). The object $\mathcal{F}'$ obtained by adding reset arcs to $\mathcal{F}$ is represented in Figure **??** (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 $\{p_6\}$). ◀

In the above proof, we have shown that some reachable markings of $\mathcal{N}_R$ were 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 **??**). 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 and complete prefix construction for reset Petri nets). Let $\mathcal{N}_R$ be a safe reset Petri net, (step 1) compute the structural translation $\mathcal{N}_{R,str}$ 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 **??** 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 **??**. The output of the algorithm is $\mathcal{F}_R$.

Applying this algorithm to the net of Figure **??** (right) – using the algorithm from [**?**] at step 2 – leads to the reset Petri net of Figure **??**.

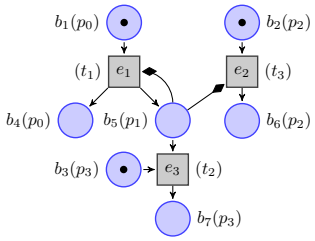

Figure 6 A finite complete prefix of the unfolding of the safe reset Petri net $\mathcal{N}_{0_R}$ of Figure **??**.

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 **??** actually builds finite complete prefixes of reset Petri nets.

▶ Proposition 6. The object $\mathcal{F}_R$ obtained by Algorithm **??** 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-folded. 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 [**?**]. 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.