Vector Addition Systems with States vs. Petri Nets

Florent Avellaneda, Rémi Morin

Second version: October 13, 2012, 10:36

Vector Addition Systems with States vs. Petri Nets

Florent Avellaneda and Rémi Morin

Université d'Aix-Marseille — CNRS, UMR 7279
Laboratoire d'Informatique Fondamentale de Marseille
163, avenue de Luminy, F-13288 Marseille Cedex 9, France

Abstract. Petri nets are a well-known and intensively studied model of concurrency often used to specify distributed or parallel systems. Vector addition systems with states are simply Petri nets provided control states. In this paper we introduce a natural partial order semantics for vector addition systems with states which extends the process semantics of Petri nets. The addition of control states to Petri nets leads to undecidable problems, namely the equality of two process languages given by two systems. However we show that basic problems about the set of markings reached along the processes of a VASS, such as boundedness, covering and reachability, can be reduced to the analogous problems for Petri nets. We show also how to check effectively any MSO property of these partial orders, provided that the system is bounded. This result generalizes known results and techniques for the model checking of compositional message sequence graphs.

Introduction

Consider a set of reactions that take place among a collection of particles such that each reaction consumes a multiset of available particles and produces a linear combination of other particle types. This kind of framework can be formalized by a vector addition system [22] or, equivalently, a (pure) Petri net [30]. Consider in addition some control state which determines whether a reaction can occur or not, and such that the occurrence of a reaction leads to a possibly distinct control state. Then the model becomes formally a vector addition system with states (a VASS), a notion introduced in [21]. It is well-known that all these models are computationally equivalent, because they can simulate each other [30, 33, 34]. More precisely any vector addition system with states over n places can be simulated by some vector addition system over $n + 3$ places [21]. These simulations do not preserve strictly the set of reachable markings because they require additional places to encode control states. Still they allow us to use techniques or tools designed for Petri nets to check the properties of the set of reachable markings of a VASS. The addition of control states to vector addition systems makes it easier to model and to analyse distributed or parallel systems. For instance it is convenient to use a vector of control states to check the structural properties of channels within a network of communicating finite state machines [24].

The popular model of message sequence graphs (MSGs) can be regarded as a particular case of VASSs where the only allowed reactions are the sending and the receipt of one message from one site to another [2, 3, 13, 18, 26]. Then each sequence of reactions can be described by a partial order of events called a message sequence chart (MSC). Each MSC corresponds to several sequences of elementary actions which are equivalent up to the reordering of independent events. Similarly each sequence of MSCs is equivalent to

several sequences of MSCs. Thus control states are used to focus on particular interleavings of events in order to avoid the state explosion problem due to concurrency. However there exists so far no way to regard an execution of a VASS as a partial order of events. Consequently there is no means to apply techniques or tools for Petri nets to the analysis of MSGs. The first contribution of this paper is the definition of a partial order semantics for VASSs in such a way that the framework of MSGs can effectively be regarded as a particular case of VASS.

Suggested by Petri in the restricted setting of condition/event systems [31], the process semantics of a Petri net defines labeled occurrence nets as partially ordered sets of events with non-branching conditions [4, 10, 16, 33, 37]. As opposed to the other classical partial-order semantics based on step firing sequences [17, 23, 37], a process records all causal dependencies between the events occurring along a run. We present in Section 1 a partial order semantics for VASSs which extends the usual process semantics of Petri nets. The approach is simple and natural. First we consider the set of firable computation sequences of a VASS and second we define the processes that represent a given sequence. Then each process describes some causal dependencies between events which are no longer linearly ordered. This means that two reactions that appear one after the other in a computation sequence can occur concurrently (that is, possibly in the reverse order) within a corresponding process. This situation is usual when modeling asynchronous systems. In particular this is similar to the way message sequence charts are derived from message sequence graphs (see, e.g. [2, **?**,3]). Thus, control states represent abstract stages of computations used to specify particular sets of reaction sequences: They do not appear formally in the process semantics. In this way, message sequence graphs are embedded in the framework of VASSs. However, one specific feature of the process semantics is that a computation sequence can yield several non-isomorphic processes depending on the order identical particles are consumed. Along this paper, we shall exhibit few other facts which make clear that the model of VASS is more general and more difficult to handle than MSGs.

It is easy to prove that checking the inclusion (or the equality) of two process languages given by two VASSs is undecidable. The reason is that the equality and the inclusion problems for rational Mazurkiewicz trace languages [7] are undecidable because the universality problem is undecidable [35]. Moreover rational Mazurkiewicz trace languages can be represented by MSGs [19] and MSGs are embedded into VASSs. This basic observation illustrates the computional gap between Petri nets and VASSs under the process semantics because these two problems are decidable for Petri nets, by means of a straightforward reduction to the covering problem [11].This shows also that the analysis of the partially ordered executions of a VASS does not boil down to the verification of a Petri net in general, in spite of the well-known simulation of a VASS by a Petri net. Synthesis problems have been investigated for various models of concurrency: Asynchronous automata [7, 8, 38], Petri nets [6, 9, 20, 28], communicating finite-state machines [2, 19], etc. They consist mainly in characterizing which formal behaviours correspond to some class of concurrent devices and to build if it exists such a device from its behavioural specification. We study in Section 2 a natural synthesis problem: Given some VASS we ask whether its processes are generated by some Petri net. We show that this problem is undecidable (even for bounded

systems) by means of a reduction to the universality problem for rational Mazurkiewicz trace languages. However we present in the rest of this paper several techniques to check properties of a VASS under the process semantics with the help of known algorithms and tools.

A key verification problem for MSGs is to detect channel divergence, i.e. to decide whether the number of pending messages along an execution is unbounded [2, **?**,3, 19]. This problem is NP-complete. An equivalent problem in the more general setting of VASSs is the prefix-boundedness problem. It consists in checking that the set of markings reached by *prefixes* of processes is finite. We present in Section 3 a technique to solve this problem by means of a new construction. We obtain that prefix-boundedness is computationally equivalent to the boundedness problem for Petri nets and requires exponential space [11]. This result exhibits an interesting complexity gap between MSGs and VASSs. It shows that algorithms to check properties of MSGs need to be revised in order to deal with the more expressive framework of VASSs. Other basic decision problems for the markings reached by prefixes are of course interesting. We show in particular that the reachability and the covering of a given marking by prefixes can be solved using the same technique.

The model-checking problem for MSGs against monadic second-order logic (MSO) was investigated first in [25]. As opposed to earlier works [2], formulas are interpreted on the partially ordered scenarios accepted by the MSGs. This problem was proved decidable for the whole class of safe MSGs [26] (see also [13]). Each safe MSG can be regarded as a bounded VASS. However a safe MSG can describe an infinite set of markings because the reordering of events can produce an unbounded number of pending messages within channels: In other words, a safe MSG may be divergent. We present in Section 4 a technique to check effectively that all processes of a given bounded VASS satisfy a given MSO formula. We shall explain in details why this result subsumes, but cannot be reduced to, previous works on the model-checking of MSGs.

1 Model and semantics

The goal of this section is to extend the usual process semantics from Petri nets to VASSs. In order to avoid repetitive definitions we introduce the model of Petri nets with states as a minimal framework which includes both Petri nets and VASSs. Thus Petri nets are regarded as Petri nets with states provided with a single state whereas VASSs are simply Petri nets with states using *pure* transition rules, only. Next we present the notions of firable computation sequence, reachable marking, and (non-branching) process as simple generalizations of the classical definitions in the restricted setting of Petri nets.

For simplicity's sake, for any mapping $\lambda : A \to B$ between two finite sets A and B, we shall denote also by λ the natural mapping $\lambda : A^\star \to B^\star$ from words over A to words over B and the mapping $\lambda : \mathbb{N}^A \to \mathbb{N}^B$ from multisets over A to multisets over B such that $\lambda(\mu) = \sum_{a \in A} \mu(a) \cdot \lambda(a)$ for each multiset $\mu \in \mathbb{N}^A$. Moreover we will often identify a set S with the multiset μ_S for which $\mu_S(x) = 1$ if $x \in S$ and $\mu_S(x) = 0$ otherwise.

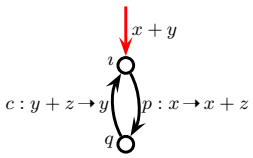

FIG. 1. A PNS with two control states

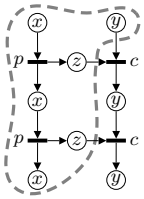

FIG. 2. A labeled causal net and a prefix

1.1 Petri net with states

We borrow from the setting of Petri nets the abstract notion of *places* which can represent different kinds of components within a system: a local control state of a sequential process, a communication channel, a shared register, a particle type, a molecule in a chemical system, etc. We let P denote a finite set of places throughout this paper. As usual a multiset of places is called a *marking* and it is regarded as a distribution of *tokens* in places. Further we fix a finite set N of *rule names*.

A transition rule (or a reaction) is a means to produce new tokens in some places by consuming tokens in some other places. Formally a *rule* is a triple $r = (\lambda, \alpha, \beta)$ where $\lambda \in N$ is a rule name and $\alpha, \beta \in \mathbb{N}^P$ are markings called the *guard* and the *update* respectively. Such a rule is denoted by $\lambda : \alpha \to \beta$. It means intuitively that a set of tokens α can be consumed to produce a set of tokens β in an atomic way. Different rules can share the same guard α and the same update β. That is why we use here rule names to distinguish between similar but distinct rules. For each rule $r = (\lambda, \alpha, \beta)$, we put ${}^\bullet r = \alpha$ and $r^\bullet = \beta$.

DEFINITION 1.1. *A* Petri net with states *(for short: a PNS) over a set of rules R is an automaton $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{\text{in}})$ where Q is a finite set of states, with a distinguished initial state $\imath \in Q$, $\longrightarrow \subseteq Q \times R \times Q$ is a finite set of arcs labeled by rules, and $\mu_{\text{in}} \in \mathbb{N}^P$ is some initial marking.*

Let $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{\text{in}})$ be a Petri net with states. A labeled arc $(q_1, r, q_2) \in \longrightarrow$ will be denoted by $q_1 \xrightarrow{r} q_2$. A rule sequence $s = r_1 ... r_n \in R^\star$ is called a *computation sequence of* $\mathcal{S}$ if there are states $q_0, ..., q_n \in Q$ such that $\imath = q_0$ and for each $i \in [1, n]$, $q_{i-1} \xrightarrow{r_i} q_i$. These conditions will be summed-up by the notation $\imath \xrightarrow{s} q_n$. For instance, $(p : x \to x + z) \cdot (c : y + z \to y) \cdot (p : x \to x + z) \cdot (c : y + z \to y)$ is a computation sequence of the PNS with two states depicted in Fig. 1. We denote by $\text{CS}(\mathcal{S})$ the set of all computation sequences of $\mathcal{S}$. This language is obviously a regular and prefix-closed set of words over R. Conversely any regular and prefix-closed language over a finite subset of rules is the set of computation sequences of some PNS. Actually the partial order semantics we shall adopt considers PNSs simply as a formal means to specify regular sets of rules.

A rule sequence $s = r_1 ... r_n \in R^\star$ is *firable* from a marking μ if there are multisets of places $\mu_0, ..., \mu_n$ such that $\mu_0 = \mu$ and for each $k \in [1, n]$: $\mu_{k-1} \geqslant {}^\bullet r_k$ and $\mu_k = \mu_{k-1} - {}^\bullet r_k + r_k^\bullet$. This means intuitively that each rule from s can be applied from the marking μ in the linear order specified by s: Each rule r_k consumes ${}^\bullet r_k$ tokens from μ_{k-1} and produces $r_k^\bullet$ new tokens which yields the subsequent multiset μ_k. Then we say that μ_n is reached by

the rule sequence s from the marking μ. We also say that s leads to μ_n. We denote by $\mathrm{FCS}(\mathcal{S})$ the set of all *firable* computation sequences of $\mathcal{S}$. A marking is *reachable* in $\mathcal{S}$ if it is reached by a firable computation sequence of $\mathcal{S}$. A PNS is said to be *bounded* if the set of its reachable markings is finite.

1.2 VASS, Petri net and causal net

Originally introduced in [21], the notion of a *vector addition system with states* (for short: a VASS) can be formally defined in several slightly different ways. In this paper, a *VASS* is simply a PNS such that each rule r labeling an arc is *pure*, which means that for all places $p \in P$, ${}^\bullet r(p) \times r^\bullet(p) = 0$. This amounts to require that ${}^\bullet r(p) \geqslant 1$ implies $r^\bullet(p) = 0$ and vice versa. For this reason each rule r in a VASS can be represented by a vector $v \in \mathbb{Z}^P$ where $v(p) = r^\bullet(p) - {}^\bullet r(p)$ for all $p \in P$. We could also require that a VASS uses a single rule name, i.e. for all rules $r_1, r_2 \in R$, ${r_1}^\bullet - {}^\bullet r_1 = {r_2}^\bullet - {}^\bullet r_2$ implies $r_1 = r_2$. In this way any two similar rules must carry the same rule name. This restriction would have no effect on the results presented in this paper.

We explain at present why we can identify the well-known formalism of Petri nets as particular PNSs provided with a single state.

Definition 1.2. *A* Petri net *is a quadruple* $\mathcal{N} = (P, T, W, \mu_{\mathrm{in}})$ *where*

- P *is a finite set of places and* T *is a finite set of transitions such that* $P \cap T = \emptyset$*;*
- W *is a map from* $(P \times T) \cup (T \times P)$ *to* $\mathbb{N}$*, called the* weight function*;*
- μ_{in} *is a map from* P *to* $\mathbb{N}$*, called the* initial marking*.*

We shall depict Petri nets in the usual way as in Fig. 4: Black rectangles represent transitions whereas circles represent places; moreover tokens in places describe the initial marking. Given a Petri net $\mathcal{N} = (P, T, W, \mu_{\mathrm{in}})$ and a transition $t \in T$, ${}^\bullet t = \sum_{p \in P} W(p,t) \cdot p$ is the *pre-multiset* of t and $t^\bullet = \sum_{p \in P} W(t,p) \cdot p$ is the *post-multiset* of t. Similarly we put ${}^\bullet p = \sum_{t \in T} W(t,p) \cdot t$ and $p^\bullet = \sum_{t \in T} W(p,t) \cdot t$ for each place $p \in P$.

Let $\mathcal{N} = (P, T, W, \mu_{\mathrm{in}})$ be a Petri net. We will regard $\mathcal{N}$ as a PNS $\mathcal{S}_\mathcal{N}$ with the same set of places P and the same initial marking. Moreover $\mathcal{S}_\mathcal{N}$ is provided with a single state $\imath$ such that each transition $t \in T$ is represented by a self-loop labeled arc $\imath \xrightarrow{r} \imath$ where $r = (t, {}^\bullet t, t^\bullet)$. In this way, the class of Petri nets is faithfully embedded into the subclass of PNSs provided with a single state such that each transition carries a rule with a distinct rule name. Conversely, take any PNS $\mathcal{S}$ with a single state $\imath$ such that each transition carries a rule with a distinct rule name. The corresponding Petri net shares with $\mathcal{S}$ its set of places and its initial marking. Moreover for each self-loop $\imath \xrightarrow{r} \imath$ it admits a transition t_r such that ${}^\bullet t_r = {}^\bullet r$ and ${t_r}^\bullet = r^\bullet$. For instance the PNS from Fig. 3 corresponds to the Petri net from Fig. 4.

If the weight function W takes only binary values then it is often described as a flow relation $F \subseteq (P \times T) \cup (T \times P)$ where $(x, y) \in F$ if $W(x, y) = 1$. Further F^+ denotes the transitive closure of F.

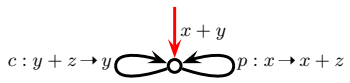

FIG. 3. A PNS with a single state

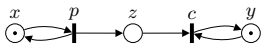

FIG. 4. and the corresponding Petri net

DEFINITION 1.3. *[10, 37] A* causal net *is a Petri net* $\mathcal{K} = (B, E, F, \mu_{\min})$ *whose places are called* conditions*, whose transitions are called* events*, and whose weight function takes values in* $\{0, 1\}$ *and is represented by a flow relation* $F \subseteq (B \times E) \cup (E \times B)$ *which satisfies the following requirements:*

1. *the net is acyclic, i.e. for all* $x, y \in B \cup E$, $(x, y) \in F^+$ *implies* $(y, x) \notin F^+$.
2. *the conditions do not branch, i.e.* $|{}^\bullet b| \leqslant 1$ *and* $|b^\bullet| \leqslant 1$ *for all* $b \in B$.
3. $\mu_{\min}(b) = 1$ *if* ${}^\bullet b = \emptyset$ *and* $\mu_{\min}(b) = 0$ *otherwise.*

Note that the third requirement guarantees that the initial marking $\mu_{\min}$ can be recovered from the structure (B, E, F) because it coincides with the set of minimal conditions. For that reason causal nets are often defined as a triple (B, E, F) satisfying the two first conditions of Def. 1.3. In the literature causal nets are also called *occurrence nets*, see e.g. [4, 14, 16, 15, 33]. However more general Petri nets are called occurrence nets in the theory of partial unfolding or branching processes [10, 12, 29].

The transitive and reflexive closure F^* of the flow relation F in a causal net $\mathcal{K} = (B, E, F, \mu_{\min})$ yields a partial order over the set of events E. A *configuration* is a subset of events $H \subseteq E$ that is downwards closed, i.e. $e' F^* e$ and $e \in H$ imply $e' \in H$. Each configuration H defines a *prefix causal net* $\mathcal{K}_H$ whose events are precisely the events from H and whose places consists of the minimal places of $\mathcal{K}$ (with respect to the partial order relation F^*) and all places related to some event from H. For instance Fig. 2 exhibits a subset of a causal net (circled with a dotted line) that is a prefix of that causal net. For each class of labeled causal nets $\mathcal{L}$, we denote by $\mathrm{Pref}(\mathcal{L})$ the class of all prefixes of all labeled causal nets from $\mathcal{L}$.

1.3 Simulation of a VASS by a Petri net

Let us now recall how a k-dimensional VASS or more generally a PNS $\mathcal{S}$ with k places can be simulated by a Petri net $\mathcal{N}$ with $k+n$ places, where n is the number of states [34, 30]. The usual construction is illustrated by Fig. 5 which shows on the left-hand side a PNS with 2 states ($\imath$ and q) and 3 places (x, y and z) and on the right-hand side the corresponding Petri net with 5 places: Each place from $\mathcal{S}$ and each state from $\mathcal{S}$ corresponds to a place from $\mathcal{N}$. The initial marking of $\mathcal{N}$ describes the initial marking of $\mathcal{S}$ and some token is added in the place corresponding to the initial state of $\mathcal{S}$. Moreover each arc $q_1 \xrightarrow{r} q_2$ in $\mathcal{S}$ is represented by a transition in $\mathcal{N}$. It is easy to see that there is a one-to-one correspondence between the firable computation sequences of $\mathcal{S}$ and the firable rule sequences of $\mathcal{N}$; moreover the marking reached by $\mathcal{N}$ describes the marking reached by $\mathcal{S}$ and the current state of $\mathcal{S}$.

This construction of $\mathcal{N}$ from $\mathcal{S}$ is interesting because it enables us to analyse the set of reachable markings of $\mathcal{S}$ by means of usual techniques from the Petri net literature (see

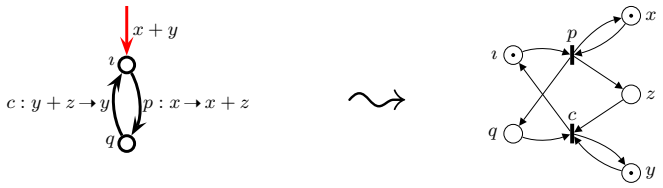

FIG. 5. Simulation of a PNS by a Petri net

[11] for a survey). In particular the boundedness problem asks whether the set of reachable markings is finite whereas the covering problem asks whether a given marking μ is covered by some reachable marking μ', i.e. $\mu \leqslant \mu'$. These two problems are decidable (for Petri nets and Petri nets with states) but they can require exponential space [32].

The simulation of a PNS by a Petri net leads us to the next result.

PROPOSITION 1.4. *Let $\mathcal{S}$ be a PNS and r be a rule attached to some arc of $\mathcal{S}$. We can decide whether r occurs in a* firable *computation sequence of $\mathcal{S}$.*

Proof. We have recalled that there is a one-to-one correspondence between the firable computation sequences of $\mathcal{S}$ and the firable computation sequences of $\mathcal{N}$. The rule r occurs in a firable computation sequence of $\mathcal{S}$ if and only if a corresponding transition t in $\mathcal{N}$ occurs in a firable transition sequence in $\mathcal{N}$. This is equivalent to check whether the marking of ${}^\bullet t$ is *covered* by a reachable marking of $\mathcal{N}$. As mentionned above this question is known to be decidable. ∎

1.4 Process semantics of a PNS

In this paper we are interested in a semantics of PNS based on causal nets which is a direct generalization of the process semantics of Petri nets [4, 10, 15, 16, 33, 37]. The process semantics of Petri nets characterizes the *labeled* causal nets that describe an execution of a given Petri net. We have already observed that each transition of a Petri net can be regarded as a rule. For that reason we adopt a graphical representation of rules similar to a transition of a Petri net, as depicted in Fig. 6. Given an initial multiset of places, each firable computation sequence can be represented by a causal net, called a process, which somehow glues together the representations of each rule. For instance the labeled causal net $\mathcal{K}$ from Fig. 2 depicts a process of the Petri net $\mathcal{N}$ from Fig. 4 in which each condition of $\mathcal{K}$ is labeled by a place of $\mathcal{N}$ and each event of $\mathcal{K}$ is labeled by a transition of $\mathcal{N}$.

The following definition explains how processes are derived from a given rule sequence. Next the processes of a PNS will be defined as the processes of its firable computation sequences (Def. 1.6).

DEFINITION 1.5. *A* process *of a rule sequence $s = r_1 ... r_n \in R^\star$ from a marking $\mu \in \mathbb{N}^P$ consists of a causal net $\mathcal{K} = (B, E, F, \mu_{\min})$ with n events e_1, ..., e_n provided with a labeling $\pi : B \cup E \to P \cup N$ such that the following conditions are satisfied:*

1. *$\pi(b) \in P$ for all $b \in B$, $\pi(e) \in N$ for all $e \in E$, and $\pi(\mu_{\min}) = \mu$;*

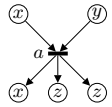

FIG. 6. Rule $a : x + y \to x + 2z$

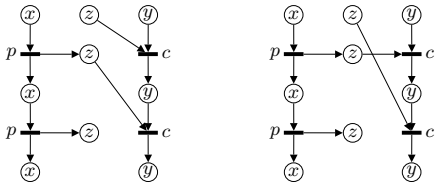

FIG. 7. Two processes from $x + y + z$

2. $r_i = (\pi(e_i), \pi({}^\bullet e_i), \pi(e_i{}^\bullet))$ *for all* $i \in [1, n]$;
3. $e_i F^+ e_j$ *implies* $i < j$ *for any two* $i, j \in [1, n]$.

We denote by $[\![s]\!]_\mu$ *the class of all processes of* s *from* μ.

In this definition the mapping π denotes the labeling of $\mathcal{K}$ and its natural extension to multisets. The first condition asserts that the initial marking of the causal net describes the marking μ; moreover each condition is associated with some place and each event corresponds to some rule name. The second condition requires that the label, the pre-set and the post-set of each event coincide with the name, the guard and the update of the corresponding rule. Finally the last property ensures that the total order of rules in s corresponds to an order extension of the partial order of events in $\mathcal{K}$. Consequently any subset of events $\{e_1, ..., e_k\}$ is downwards closed. Moreover the prefix causal net $\mathcal{K}'$ corresponding to the configuration $\{e_1, ..., e_{n-1}\}$ is a process of the rule sequence $r_1...r_{n-1}$ from the same marking μ. Consequently the class of processes of a rule sequence could be also defined inductively over its length, as we will see in Prop. 3.3. Furthermore it is easy to see that the class of processes of a rule sequence is empty if and only if the rule sequence is not firable from $\mu_{\min}$.

Let H be a configuration of a process $\mathcal{K} = (B, E, F, \mu_{\min}, \pi)$ of a rule sequence s from μ. Let $B_{\max}$ be the set of maximal conditions of the prefix $\mathcal{K}_H$ w.r.t. F^*. Then the multiset of places $\pi(B_{\max})$ is called the *marking reached by* $\mathcal{K}_H$ and we say that $\mathcal{K}_H$ *leads to the marking* $\pi(B_{\max})$. Let s_H be a linear extension of the events from H. Then it is clear that the rule sequence $\pi(s_H)$ is firable from μ and leads to the marking $\pi(B_{\max})$; moreover $\mathcal{K}_H$ is a process of $\pi(s_H)$ from μ.

Roughly speaking, any labeled causal net isomorphic to a process of s is also a process of s. In particular the class of processes of the empty rule sequence from some marking μ collects all labeled causal nets with no event and such that its set of labeled places represents the multiset μ. Further a rule sequence may give rise to multiple (non-isomorphic) causal nets depending on the consumption of tokens by each event and the initial marking. For instance the computation sequence $(p : x \to x+z)\cdot(c : y+z \to y)\cdot(p : x \to x+z)\cdot(c : y+z \to y)$ of the PNS from Fig. 1 corresponds to the causal net from Fig. 2. However if there are $x + y + z$ tokens initially, then this computation sequence corresponds to the two labeled causal nets from Fig. 7 among some others.

DEFINITION 1.6. *Let* $\mathcal{S}$ *be a PNS with initial marking* μ_{in}. *A* process of $\mathcal{S}$ *is a process of a computation sequence of* $\mathcal{S}$ *from* μ_{in}. *We let* $[\![\mathcal{S}]\!]$ *denote the class of all processes of* $\mathcal{S}$.

Thus $[\![\mathcal{S}]\!] = \bigcup_{s \in CS(\mathcal{S})} [\![s]\!]_{\mu_{in}}$. It is easy to check that the processes of a PNS provided with a single state are precisely the processes of the corresponding Petri net w.r.t. the usual process semantics [4, 15, 37]. Moreover any prefix of a process of $\mathcal{S}$ is a process of some rule sequence. Consequently the class of processes of a Petri net is closed by prefixes. However *the set of processes of a PNS need not to be prefix-closed in general*, as the next example shows.

Example 1.7. Consider the PNS from Fig. 1 with initial marking $x + y$ and its process depicted in Fig. 2. Clearly the prefix of this process circled with the dotted line in Fig. 2 is *not* a process of that PNS.

A PNS is said to be *prefix-bounded* if the set of markings reached by prefixes is finite. Clearly any prefix-bounded PNS is bounded. The converse property does not hold in general. Continuing Example 1.7, each process of the PNS from Fig. 1 leads to a marking with at most 3 tokens whereas prefixes of these processes lead to infinitely many distinct markings (see in Fig. 2 a prefix of a process which leads to a marking with 4 tokens). However we stress that each bounded Petri net is prefix-bounded because its class of processes is prefix-closed.

Note that the simulation of a PNS by a Petri net considered in Subsection 1.3 is *not* faithful from the partial order point-of-view we adopt here. Consider again Figure 5. The processes of the PNS $\mathcal{S}$ (with three places) on the left-hand side differ from the processes of the Petri net $\mathcal{N}$ (with five places) on the right-hand side. In Section 3 we introduce a simulation of a PNS by another PNS that allows us to analyse the set of markings reached along the prefixes of the processes of a given PNS.

1.5 From compositional MSGs to PNSs

The formalism of compositional message sequence graphs (cMSGs) was introduced in [18] in order to strengthen the expressive power of MSGs and to provide an algebraic framework for the whole class of regular sets of MSCs [19]. As opposed to usual MSGs, cMSGs are built on components MSCs in which unmatched send or receive events are allowed. It was argued in [18] that simple protocols such as the alternating bit protocol can be described by cMSGs but not by MSGs. With no surprise cMSGs can be regarded as a particular case of VASS under the process semantics.

Consider a distributed system consisting of a set I of sites and a set K of communication channels between pairs of sites. The behaviour of such a system can be specified by a PNS over the set $P = I \cup K$ of places such that the sending of a message from site i to site j within the channel $k_{i,j}$ from i to j is encoded by a rule $i \to i + k_{i,j}$ and the receipt of such a message is encoded by a rule $j + k_{i,j} \to j$. Then we require that the initial marking (and each reachable marking) contains a single token in each place $i \in I$ so that all events on a given site are linearly ordered. Such a PNS can actually be regarded as a compositional message sequence graph. The partial order semantics of cMSGs consists of message sequence charts which are simply a partial order of events obtained from a process by removing all conditions.

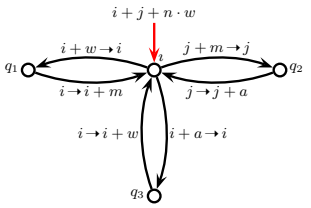

FIG. 8. Sliding window protocol

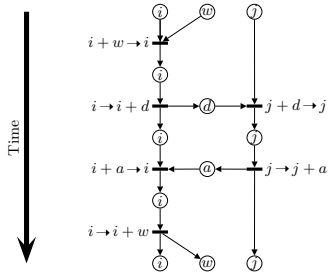

FIG. 9. A process with $n=1$

EXAMPLE 1.8. The PNS from Figure 8 describes a simplified sliding window protocol used to transmit data from a server i to a client j. The maximal number of missing acknowledgments is specified by the n initial tokens in the place w (the window). The system behaviour consists of three basic steps.

1. The server sends a new data formalized by a token d if some token w is available: It consumes first a w token: $i+w\to i$ and next sends a new data: $i\to i+d$.
2. The client receives a data and returns an acknowledgment formalized by a token a: It consumes first a data: $j+d\to j$ and next produces the ack: $j\to j+a$.
3. The server receives an acknowledgment and increments the window size: First the ack is consumed: $i+a\to i$ and then a new token w is released: $i\to i+w$.

A typical process of this system with $n=1$ is depicted in Figure 9. It is clear that this system is bounded and even prefix-bounded.

Since local variables are prohibited in MSGs, the size of any safe cMSG equivalent to the PNS from the above example is exponential in the size of n. Thus a bounded PNS can be exponentially more concise than an equivalent safe cMSG. If this protocol starts with an initial window size of $n=2^k\cdot w$, then any safe cMSG describing the same class of processes needs 2^k distinct states.

2 Checking inclusion properties

A classical issue in concurrency theory consists in characterizing the expressive power of a model. Then a usual problem is the synthesis of a system from its behavioural specification. In this section we consider Petri nets with states as a means to specify concurrent behaviours in the form of processes. We tackle the problem of building a Petri net equivalent to some given Petri net with states. Two classes of specifications are studied according to the notion of equivalence we adopt.

DEFINITION 2.1. *A Petri net with states* $\mathcal{S}$ *is* realizable *(resp.* prefix-realizable*) if there is some Petri net* $\mathcal{N}$ *such that* $[\![\mathcal{S}]\!]=[\![\mathcal{N}]\!]$ *(resp.* $\mathrm{Pref}([\![\mathcal{S}]\!])=[\![\mathcal{N}]\!]$*).*

Note that the Petri net with states $\mathcal{S}$ from Figure 1 is *not* realizable because the set of processes it accepts is not prefix-closed (Example 1.7) whereas the set of processes

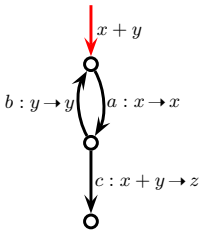

FIG. 10. A non prefix-realizable PNS

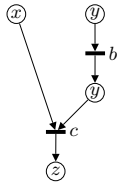

FIG. 11. Some implied process

recognized by any Petri net is prefix-closed. However $\mathcal{S}$ is prefix-realizable because the prefixes of its processes are precisely the processes of the Petri net with states provided with a single state depicted in Fig. 3 (i.e. the Petri net from Fig. 4). The next example exhibits a Petri net with states that is not prefix-realizable.

EXAMPLE 2.2. Consider the PNS $\mathcal{S}$ from Figure 10. Any Petri net $\mathcal{N}$ such that $[\![\mathcal{N}]\!] = \mathrm{Pref}([\![\mathcal{S}]\!])$ would accept the causal net $\mathcal{K}$ from Figure 11 as a process. However $\mathcal{K}$ is obviously not the prefix of some process from $\mathcal{S}$. Therefore $\mathcal{S}$ is *not* prefix-realizable.

Although realizability appears to be the simplest problem to consider, we claim that prefix-realizability is also a natural issue because the processes of a Petri net are prefix-closed. Further considering prefixes is often a means to focus on deadlock-free implementations of systems provided with a notion of accepting states. The next basic observation exhibits a canonical candidate for the synthesis of a Petri net from a Petri net with states.

PROPOSITION 2.3. *Let $\mathcal{S}_1$ be a Petri net with states and R_1 be the subset of rules occurring in some* firable *computation sequence of $\mathcal{S}_1$. Let $\mathcal{S}_2$ be the PNS provided with a single state and the same initial marking as $\mathcal{S}_1$ such that a rule occurs on a self-loop in $\mathcal{S}_2$ if and only if it belongs to R_1. Then*

1. *$\mathcal{S}_1$ is realizable if and only if $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$.*
2. *$\mathcal{S}_1$ is prefix-realizable if and only if $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = [\![\mathcal{S}_2]\!]$.*

Proof. Assume first that $\mathcal{S}_1$ is not prefix-realizable. Then $\mathrm{Pref}([\![\mathcal{S}_1]\!]) \neq [\![\mathcal{S}_2]\!]$ because $\mathcal{S}_2$ is equivalent to a Petri net. Assume now that $\mathcal{S}_1$ is prefix-realizable. Then there exists some PNS $\mathcal{S}'$ with a single state such that $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = [\![\mathcal{S}']\!]$. Any rule from R_1 occurs in some process of $\mathcal{S}_1$, so it must occur in some process of $\mathcal{S}'$: Therefore it occurs on a self-loop in $\mathcal{S}'$. Any other rule occurring on a self-loop in $\mathcal{S}'$ cannot occur in a firable computation sequence. Therefore we can remove it from $\mathcal{S}'$ without affecting the set of processes of $\mathcal{S}'$. In other words we can assume $\mathcal{S}' = \mathcal{S}_2$. A similar argument holds for realizability. ■

Note that R_1 can be computed from $\mathcal{S}_1$ (Prop. 1.4). Clearly $\mathrm{Pref}([\![\mathcal{S}_1]\!]) \subseteq [\![\mathcal{S}_2]\!]$. Thus the difference between the specification $\mathcal{S}_1$ and the canonical implementation $\mathcal{S}_2$ stems from processes built on rules of $\mathcal{S}_1$ that are not represented by some computation sequence of $\mathcal{S}_1$. This situation is similar to the notion of an implied scenario in the setting of realizable high-level message sequence charts [1].

2.1 An undecidable problem with Mazurkiewicz traces

The undecidability results presented in this section rely on the universality problem in the setting of Mazurkiewicz trace theory [7] that we recall now. Let Σ be some finite alphabet of actions. The concurrency of a distributed system is often represented by an *independence relation* over Σ, that is a binary, symmetric, and irreflexive relation $\| \subseteq \Sigma \times \Sigma$. Then the pair $(\Sigma, \|)$ is called an *independence alphabet*. The associated *trace equivalence* is the least congruence $\sim$ over $\Sigma^\star$ such that $a\|b$ implies $ab \sim ba$ for all $a, b \in \Sigma$. We let $[u]$ denote the trace equivalence class of a word $u \in \Sigma^\star$ and we put $[L] = \bigcup_{u \in L}[u]$ for any language $L \subseteq \Sigma^\star$.

THEOREM 2.4. *[36, Theorem IV.4.3] It is undecidable whether $[L] = \Sigma^\star$ for a given independence alphabet $(\Sigma, \|)$ and a given regular language $L \subseteq \Sigma^\star$.*

Since we have not provided the model of VASS with the notion of accepting states, we need the slightly stronger but immediate next statement.

COROLLARY 2.5. *It is undecidable whether $[L] = \Sigma^\star$ for some given independence alphabet $(\Sigma, \|)$ and some given regular* and prefix-closed *language $L \subseteq \Sigma^\star$.*

Proof. We proceed by contradiction. We assume that this problem is decidable and show that the problem from Theorem 2.4 becomes decidable. Let $(\Sigma, \|)$ be some independence alphabet and $L \subseteq \Sigma^\star$ be some regular language. We consider some additional letter $\perp$ and the new alphabet $\Gamma = \Sigma \cup \{\perp\}$ provided with the same independence relation: The new letter $\perp$ is dependent with all letters from Σ. Let $L' = \mathrm{Pref}(L) \cup (L \cdot \{\perp\} \cdot \Gamma^\star)$. It is clear that L' is regular and prefix-closed. Moreover $L \subseteq L'$. To conclude the proof we can check easily that $[L'] = \Gamma^\star$ if and only if $[L] = \Sigma^\star$.

Assume first that $[L] = \Sigma^\star$. It is clear that $[L'] \subseteq \Gamma^\star$. Let $v \in \Gamma^\star$. We distinguish two cases: If $v \in \Sigma^\star$ then $v \sim u$ for some $u \in L$. If $v \notin \Sigma^\star$ then $v = v_0.\perp.v_1$ with $v_0 \in \Sigma^\star$ and $v_1 \in \Gamma^\star$. Furthermore $v_0 \sim u_0$ for some $u_0 \in L$. It follows that $v \sim u_0.\perp.v_1$ and $u_0.\perp.v_1 \in L'$. In both cases we get $v \in [u]$ for some $u \in L'$. Hence $[L'] = \Gamma^\star$.

Conversely assume now that $\Gamma^\star = [L']$ and consider $v \in \Sigma^\star$. Then $v.\perp \in \Gamma^\star$. There exists some $u \in L'$ such that $v.\perp \sim u$. Then $u = u_0.\perp$ because $\perp$ is dependent with all letters. Moreover $v \sim u_0$ (because the trace equivalence is right-cancellative) and $u_0 \in L' \cap \Sigma^\star$ (because the trace equivalence is a Parikh equivalence). It follows that $u_0 \in L$. Hence $[L] = \Sigma^\star$. ∎

COROLLARY 2.6. *Let $(\Sigma, \|)$ be an independence alphabet. It is undecidable whether $[L_1] \subseteq [L_2]$ for any two regular and prefix-closed language $L \subseteq \Sigma^\star$.*

Proof. Consider $L_1 = \Sigma^\star$ and apply Cor. 2.5. ∎

In the sequel of this section, we present a natural encoding of Mazurkiewicz traces in the form of causal nets. Then each prefix-closed rational Mazurkiewicz trace languages can be represented by a prefix-bounded PNS. As a consequence the inclusion relation $[\![\mathcal{S}_1]\!] \subseteq [\![\mathcal{S}_2]\!]$ is undecidable for two given prefix-bounded PNSs $\mathcal{S}_1$ and $\mathcal{S}_2$.

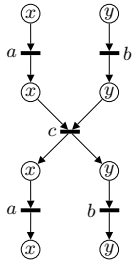

FIG. 12. Some process corresponding to the rule sequence $\rho(abcab)$ with $a \| b$

2.2 From Mazurkiewicz traces to processes

Let $(\Sigma, \|)$ be a fixed independence alphabet. We consider a finite set of places P and some mapping $\mathrm{Loc} : \Sigma \to 2^P$ such that $a \| b$ iff $\mathrm{Loc}(a) \cap \mathrm{Loc}(b) = \emptyset$. There are several ways to find such a set P together with the location mapping Loc, one of which is to consider any subset $\{a, b\} \subseteq \Sigma$ to be a place whenever $a \not\| b$ and to put $\mathrm{Loc}(a) = \{p \in P \mid a \in p\}$. We assume that each place $p \in P$ occurs in some location $\mathrm{Loc}(a)$ of some action $a \in \Sigma$. We put $N = \Sigma$, $R_\Sigma = \{(a, \alpha, \alpha) \in R \mid \alpha = \mathrm{Loc}(a)\}$ and $\mu_{\mathrm{in}} = P$. Note that there is exactly one rule $(a, \mathrm{Loc}(a), \mathrm{Loc}(a)) \in R_\Sigma$ for each action $a \in \Sigma$. Moreover these rules are *synchronisation rules* according to the next definition.

DEFINITION 2.7. *A rule $r = (\lambda, \alpha, \beta)$ is a* synchronisation rule *if $\alpha = \beta$ and $\alpha(m) \leqslant 1$ for each $m \in P$.*

We consider the mapping $\rho : \Sigma \to R_\Sigma$ such that $\rho(a) = (a, \mathrm{Loc}(a), \mathrm{Loc}(a))$. This bijection extends naturally to mapping between words over Σ and words over R_Σ.

EXAMPLE 2.8. Let $\Sigma = \{a, b, c\}$ provided with the independence relation $a \| b$. We consider $P = \{x, y\}$ together with $\mathrm{Loc}(a) = \{x\}$, $\mathrm{Loc}(b) = \{y\}$ and $\mathrm{Loc}(c) = \{x, y\}$. Figure 12 depicts some process corresponding to the rule sequence $\rho(abcab)$.

Note that for any word $u \in \Sigma^\star$ the rule sequence $\rho(u)$ is firable from μ_{in} and leads to the marking μ_{in}. It follows from Prop. 3.3 that all processes from $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}}$ are isomorphic to each other, i.e. there is intuitively only one process for $\rho(u)$ from μ_{in}.

The next result asserts that trace equivalent words give rise to the same processes. And conversely, if two words correspond to the same processes, then these two words are trace equivalent. In this way equivalence classes of words are identified with processes. This property is actually similar to the well-known fact that trace equivalence classes of words can be represented by particular labeled partial orders.

LEMMA 2.9. *For all $u, v \in \Sigma^\star$: $u \sim v$ if and only if $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v)]\!]_{\mu_{\mathrm{in}}}$.*

Proof. Let $u \in \Sigma^\star$ and $a, b \in \Sigma$ such that $a \neq b$. If $u.ab \sim u.ba$ then $a \| b$, $\mathrm{Loc}(a) \cap \mathrm{Loc}(b) = \emptyset$, and $[\![\rho(u.ab)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(u.ba)]\!]_{\mu_{\mathrm{in}}}$ by Prop. 3.3. Therefore $u \sim v$ implies $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v)]\!]_{\mu_{\mathrm{in}}}$ for all $u, v \in \Sigma^\star$ (again with the help of Prop. 3.3). To prove the converse property, we proceed by induction over the length of u. The base case is trivial. We consider $u, v \in \Sigma^\star$ of length $n + 1$ such that $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v)]\!]_{\mu_{\mathrm{in}}}$. We distinguish two cases:

1. $u = u'.a$ and $v = v'.a$ for some $u', v' \in \Sigma^\star$ and $a \in \Sigma$. We have $\rho(u) = \rho(u').\rho(a)$ and $\rho(v) = \rho(v').\rho(a)$. By Prop. 3.3 we have $[\![\rho(u')]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v')]\!]_{\mu_{\mathrm{in}}}$. It follows from induction hypothesis that $u' \sim v'$ hence $u'.a \sim v'.a$.
2. $u = u'.a$ and $v = v'.b$ for some $u', v' \in \Sigma^\star$ and $a, b \in \Sigma$ with $a \neq b$. We have $\rho(u) = \rho(u').\rho(a)$ and $\rho(v) = \rho(v').\rho(b)$ with $\rho(a) \neq \rho(b)$. Then any labeled causal net $\mathcal{K}$ from $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v)]\!]_{\mu_{\mathrm{in}}}$ includes two maximal events e_a and e_b labeled by a and b respectively. It follows that $a \| b$. Let $\mathcal{K}'$ be the prefix of $\mathcal{K}$ obtained by erasing the two maximal event e_a and e_b. We consider a linear extension w' of the Σ-labeled events from $\mathcal{K}'$. Then $\mathcal{K}'$ is the process from $[\![\rho(w')]\!]_{\mu_{\mathrm{in}}}$. Moreover $[\![\rho(w'.a)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v')]\!]_{\mu_{\mathrm{in}}}$ and $[\![\rho(w'.b)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(u')]\!]_{\mu_{\mathrm{in}}}$. By induction hypothesis, we get $w'.a \sim v'$ and $w'.b \sim u'$. On the other hand $w'.ab \sim w'.ba$ because $a \| b$. Hence $u \sim w'.ba \sim w'.ab \sim v$.

■

We consider at present a regular and prefix-closed language $L \subseteq \Sigma^\star$ and a finite automaton $\mathcal{A}(L) = (Q, \imath, \longrightarrow_{\mathcal{A}(L)})$ whose states are all accepting and which recognized L. We may assume that each state of $\mathcal{A}(L)$ is reachable from the initial state and each action of Σ appears on a labeled arc of $\mathcal{A}(L)$. We build from the automaton $\mathcal{A}(L)$ the PNS $\mathcal{S}(L) = (Q, \imath, \longrightarrow_{\mathcal{S}(L)}, \mu_{\mathrm{in}})$ with the same set of states Q, the same initial state $\imath \in Q$ and such that for each rule $r = (a, \alpha, \alpha) \in R_\Sigma$ and all states $q_1, q_2 \in Q$, there is some labeled arc $q_1 \xrightarrow{r}_{\mathcal{S}(L)} q_2$ if $q_1 \xrightarrow{a}_{\mathcal{A}(L)} q_2$. Observe here the multiset of tokens is left unchanged by each rule. Consequently the set of markings reached by prefixes of $[\![\mathcal{S}(L)]\!]$ is finite, i.e. the PNS $\mathcal{S}_L$ is prefix-bounded. For any two regular and prefix-closed languages $L_1, L_2 \subseteq \Sigma^\star$ Lemma 2.9 shows that we have $[L_1] \subseteq [L_2]$ if and only if $[\![\mathcal{S}_{L_1}]\!] \subseteq [\![\mathcal{S}_{L_2}]\!]$. Thus the property $[\![\mathcal{S}_1]\!] \subseteq [\![\mathcal{S}_2]\!]$ is undecidable for two given prefix-bounded PNS $\mathcal{S}_1$ and $\mathcal{S}_2$. We can strengthen this observation by the next statement.

THEOREM 2.10. *The property $[\![\mathcal{N}]\!] \subseteq [\![\mathcal{S}]\!]$ is undecidable for a prefix-bounded PNS $\mathcal{S}$ and a bounded Petri net $\mathcal{N}$.*

Proof. Let $L \subseteq \Sigma^\star$ be a regular and prefix-closed language and $\mathcal{S}(L)$ be the corresponding PNS. Let $\mathcal{N}(L)$ be the Petri net collecting all rules $R^\star_\Sigma$ of $\mathcal{S}(\mathcal{L})$. Then $[\![\mathcal{N}(L)]\!] = [\![R^\star_\Sigma]\!]$. Moreover we can check that $[\![\mathcal{S}(L)]\!] = [\![R^\star_\Sigma]\!]$ if and only if $[L] = \Sigma^\star$.

Assume first that $[\![\mathcal{S}]\!] = [\![R^\star_\Sigma]\!]$. Let $u \in \Sigma^\star$. We have $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}} = [\![w]\!]_{\mu_{\mathrm{in}}}$ for some $w \in \mathrm{CS}(\mathcal{S})$. Let $v = \rho^{-1}(w)$. Clearly $v \in L$. Since $[\![\rho(u)]\!]_{\mu_{\mathrm{in}}} = [\![\rho(v)]\!]_{\mu_{\mathrm{in}}}$ we get $u \sim v$ by Lemma 2.9. Hence $\Sigma^\star = [L]$.

Assume now that $[L] = \Sigma^\star$. Let $w \in R^\star_\Sigma$. We have $\rho^{-1}(w) \in \Sigma^\star$. Then $\rho^{-1}(w) \sim u$ for some $u \in L$. It follows from Lemma 2.9 that $[\![w]\!]_{\mu_{\mathrm{in}}} = [\![\rho(u)]\!]_{\mu_{\mathrm{in}}}$. Moreover $\rho(u) \in \mathrm{CS}(\mathcal{S})$. Therefore $[\![R^\star_\Sigma]\!]_{\mu_{\mathrm{in}}} = [\![\mathrm{CS}(\mathcal{S})]\!]_{\mu_{\mathrm{in}}}$. ■

By means of a slightly more involved encoding of Mazurkiewicz traces, we show in the next section that Theorem 2.10 holds also if $\mathcal{S}$ is a prefix-bounded VASS.

2.3 Gap between VASS and Petri nets

At present we focus on the subclass of vector addition systems with states, i.e. Petri nets with states with pure rules only. So far no rule from R_Σ is pure, so the processes obtained

FIG. 13. Building the pure rule corresponding to a synchronisation rule

FIG. 14. A release rule

from R_Σ cannot be described by a VASS. For this reason we have to use a slightly more involved encoding of Mazurkiewicz tracesbut we keep the same set of rule names ΣLet us consider the set of places $P^\circ = P \times \{0,1\}$ and the initial marking $\mu^\circ_{\text{in}} = P \times \{0\}$. This means that we use two copies of each place from P. Moreover we will make sure that any reachable marking will contain exactly one of these two copies. Intuitively places tagged by 0 are *available* and may be consumed by the system whereas places tagged by 1 are *locked* and need to be released. We let $\pi : P^\circ \to P$ denote the first projection: $\pi(m,n) = m$ for each $m \in P$. This mapping extends naturally to a mapping from multisets over P° to multisets over P: $\pi(\mu) = \sum_{(m,n)\in P^\circ} \mu(m,n) \cdot \pi(m,n)$.

We let R_1 collect all rules (a, α, β) over P° such that $(a, \pi(\alpha), \pi(\beta)) \in R_\Sigma$, $\alpha(m,n) \geqslant 1$ implies $n = 0$ and $\beta(m,n) \geqslant 1$ implies $n = 1$. Thus we require that α and β correspond to the same set of untagged tokens, i.e. $\pi(\alpha) = \pi(\beta)$. Moreover we require that the tokens consumed are available whereas the tokens produced are locked. We denote by $\pi : R_1 \to R_\Sigma$ the function which maps each rule $(a, \alpha, \beta) \in R_1$ to $(a, \pi(\alpha), \pi(\beta)) \in R_\Sigma$. It is clear that this mapping is a bijection. For instance Figure 13 depicts a synchronization rule from R_Σ together with the corresponding rule from R_1.

We consider also a set of additional *release rules* that consume a locked place and produce the corresponding available oneas depicted in Figure 14Formally we let R_2 denote the set of rules (a, α, β) such that $|\alpha| = |\beta| = 1$, $\alpha(m,0) = 0$ for all $m \in P$, and $\alpha(m,1) = 1$ implies $\beta(m,0) = 1$. We put $R_0 = R_1 \cup R_2$.

We build from $\mathcal{S}$ the Petri net with states $\mathcal{S}^\circ = (Q, \imath, \longrightarrow_{\mathcal{S}^\circ}, \mu^\circ_{\text{in}})$ with the same set of states Q, the same initial state $\imath \in Q$, and $\mu^\circ_{\text{in}} = P \times \{0\}$ as initial marking. The labeled arcs of $\mathcal{S}^\circ$ are defined as follows: For each rule $r \in R_0$ and for all states $q_1, q_2 \in Q$, we put $q_1 \xrightarrow{r}_{\mathcal{S}^\circ} q_2$ if one of these two conditions is satisfied:

- $r \in R_1$ and $q_1 \xrightarrow{a}_{\mathcal{S}} q_2$ with $a = \pi(r)$;
- $r \in R_2$ and $q_1 = q_2$.

Note here that $\mathcal{S}^\circ$ is a VASS because each rule from R_0 is pure. Since each place $p \in P$ occurs in the location $\text{Loc}(a)$ of some action $a \in \Sigma$ and each action a appears on a labeled arc of $\mathcal{A}$ starting from a state reachable from its initial state $\imath$, it is clear that each rule from R_0 appears in some *firable* computation sequence of $\mathcal{S}^\circ$.

The bijection $\pi : R_1 \to R_\Sigma$ can be regarded as a function $\pi : R_0 \to R_\Sigma \cup \{\varepsilon\}$ where $\pi(r)$ map to the empty word ε for each $r \in R_2$. This function extends naturally to a mapping $\pi : R_0^\star \to R_\Sigma^\star$ which associates each rule sequence $r_1 ... r_n$ from $R_0^\star$ to the rule sequence $\pi(r_1) ... \pi(r_n)$ from $R_\Sigma^\star$. Due to the similar structure between $\mathcal{S}^\circ$ and $\mathcal{S}$, it is clear

that each computation sequence u of $\mathcal{S}^\circ$ maps to some computation sequence $\pi(u)$ of $\mathcal{S}$. Furthermore, firable computation sequences correspond to firable computation sequences. Thus we have $\pi : \mathrm{FCS}(\mathcal{S}^\circ) \to \mathrm{FCS}(\mathcal{S})$. The next observation asserts that the mapping $\pi : \mathrm{FCS}(\mathcal{S}^\circ) \to \mathrm{FCS}(\mathcal{S})$ is actually onto.

PROPOSITION 2.11. *For all $u \in FCS(\mathcal{S})$ there exists $u^\circ \in FCS(\mathcal{S}^\circ)$ such that $\pi(u^\circ) = u$.*

Proof. By an immediate induction over the length of u, we can check that for each $u \in \mathrm{FCS}(\mathcal{S})$ there exists some $u^\circ \in \mathrm{FCS}(\mathcal{S}^\circ)$ such that $\pi(u^\circ) = u$ and the marking reached by u° is μ°_{in}. ■

Recall that each rule of $\mathcal{S}$ is a synchronisation rule and the initial marking of $\mathcal{S}$ consists of a single token in each place. As a consequence, for each rule sequence $u \in R^\star_\Sigma$, the class of processes $[\![u]\!]_{\mu^\circ_{\mathrm{in}}}$ consists of isomorphic labeled causal nets. For any two rule sequences $u, v \in R^\star_\Sigma$, we put $u \simeq v$ if $[\![u]\!]_{\mu^\circ_{\mathrm{in}}} = [\![v]\!]_{\mu^\circ_{\mathrm{in}}}$. Similarly, for each rule sequence $u \in R^\star_0$, the set of processes $[\![u]\!]_{\mu^\circ_{\mathrm{in}}}$ consists of isomorphic labeled causal nets because the marking reached by a firable rule sequence is a set (not a multiset). Moreover we have $[\![u]\!]_{\mu^\circ_{\mathrm{in}}} \neq \emptyset$ if and only if the rule sequence u is firable. For any two rule sequences $u, v \in R^\star_0$, we put $u \simeq^\circ v$ if $u = v$ or $[\![u]\!]_{\mu^\circ_{\mathrm{in}}} = [\![v]\!]_{\mu^\circ_{\mathrm{in}}} \neq \emptyset$. The second observation ensures that this process equivalence is preserved by the mapping $\pi : \mathrm{FCS}(\mathcal{S}^\circ) \to \mathrm{FCS}(\mathcal{S})$.

PROPOSITION 2.12. *For all $u_1, u_2 \in FCS(\mathcal{S}^\circ)$, $u_1 \simeq^\circ u_2$ implies $\pi(u_1) \simeq \pi(u_2)$.*

Proof. Let $u_1, u_2 \in \mathrm{FCS}(\mathcal{S}^\circ)$ be such that $u_1 \simeq^\circ u_2$ and $u_1 \neq u_2$. Let $\mathcal{K}$ be the labeled causal net from $[\![u_1]\!]_{\mu^\circ_{\mathrm{in}}}$. Then u_1 and u_2 are two linear extensions of the partial order of rules occurring in $\mathcal{K}$. We may assume that $u_1 = v.ab.w$ and $u_2 = v.ba.w$ with $v, w \in R^\star_0$ and $a, b \in R_0$. We distinguish then two cases.

1. $a \in R_2$ or $b \in R_2$. Then $\pi(u_1) = \pi(u_2)$ hence $\pi(u_1) \simeq \pi(u_2)$.
2. $a \in R_1$ and $b \in R_1$. Since u_1 and u_2 are two linear extensions of $\mathcal{K}$, the guards of a and b are disjoint. It follows that $\pi(u_1).\pi(a).\pi(b) \simeq \pi(u_1).\pi(b).\pi(a)$ hence $\pi(u_1) \simeq \pi(u_2)$. ■

We will also need the next technical result.

PROPOSITION 2.13. *For all firable computation sequences $v \in FCS(\mathcal{S})$ and all firable rule sequences $u \in R^\star_0$, if $\pi(u) \simeq v$ then $u \simeq^\circ w$ for some firable computation sequence $w \in FCS(\mathcal{S}^\circ)$.*

Proof. We distinguish two cases.

1. The marking reached by u consists of available places only. We consider the rule sequence $w \in R_0$ built inductively over the length of v by replacing each rule r from v by the corresponding rule $\pi^{-1}(r) \in R_1$ followed by a series of release rules from R_2 such that all locked places produced by $\pi^{-1}(r)$ are released. Then $w \in \mathrm{FCS}(\mathcal{S}^\circ)$ and $\pi(w) = v$. Hence $\pi(u) \simeq \pi(w)$. It follows that $u \simeq^\circ w$.
2. Some places in the marking reached by u are locked. We add a sequence of release rules z to u to get $w = u.z$ such that the marking reached by w consists of available places only. Then $\pi(w) \simeq v$. We apply the first case to get some firable computation sequence $w' \in \mathrm{FCS}(\mathcal{S}^\circ)$ such that $w \simeq^\circ w'$. We can remove from w' the release rules of z and get some firable computation sequence $w'' \in \mathrm{FCS}(\mathcal{S}^\circ)$ such that $u \simeq^\circ w''$. ■

Observe here the number of tokens is constant whenever a rule is applied. Consequently the set of markings reached by prefixes of $[\![\mathcal{S}^\circ]\!]$ is finite, i.e. $\mathcal{S}^\circ$ is prefix-bounded. We can prove that $\mathcal{S}^\circ$ is realizable if and only if $\mathcal{S}$ is realizable. Thus,

THEOREM 2.14. *It is undecidable whether a given prefix-bounded VASS is realizable.*

Proof. Let $\mathcal{N}$ be the Petri net consisting of all rules of R_Σ with the initial marking $\mu_{\text{in}} = P$. By Prop. 2.3, $\mathcal{S}$ is realizable if and only if $[\![\mathcal{S}]\!] = [\![\mathcal{N}]\!]$. Moreover $\mathcal{N}$ is equivalent to a PNS with only synchronisation rules (and with only one state). We let $\mathcal{N}^\circ$ be the VASS corresponding to $\mathcal{N}$ w.r.t. the above construction of $\mathcal{S}$ from $\mathcal{S}^\circ$. We may apply the three above propositions with $\mathcal{N}$ and $\mathcal{N}^\circ$ respectively. Since $\mathcal{N}^\circ$ has a single state, it is equivalent to a pure Petri net. Further $\mathcal{N}^\circ$ consists of all rules of R_0 and its initial marking is μ°_{in}. Since each rule from R_0 appears in some *firable* computation sequence of $\mathcal{S}^\circ$, Prop. 2.3 claims that $\mathcal{S}^\circ$ is realizable if and only if $[\![\mathcal{S}^\circ]\!] = [\![\mathcal{N}^\circ]\!]$. We can check that $\mathcal{S}^\circ$ is realizable if and only if $\mathcal{S}$ is realizable.

Assume first that $\mathcal{S}^\circ$ is realizable: We have $[\![\mathcal{S}^\circ]\!] = [\![\mathcal{N}^\circ]\!]$. Let $\mathcal{K} \in [\![\mathcal{N}]\!]$. Let u be a linear extension of the partial order of rules occurring in $\mathcal{K}$. Then $u \in \text{FCS}(\mathcal{N})$. There exists some firable computation sequence $u^\circ \in \text{FCS}(\mathcal{N}^\circ)$ such that $\pi(u^\circ) = u$ (Prop. 2.11 applied with $\mathcal{N}$ and $\mathcal{N}^\circ$). Then $u^\circ \simeq^\circ v^\circ$ for some $v^\circ \in \text{FCS}(\mathcal{S}^\circ)$ because $[\![\mathcal{S}^\circ]\!] = [\![\mathcal{N}^\circ]\!]$. Furthermore $u = \pi(u^\circ) \simeq \pi(v^\circ) \in \text{FCS}(\mathcal{S})$ by Prop. 2.12. Hence $\mathcal{K} \in [\![\mathcal{S}]\!]$. It follows that $[\![\mathcal{S}]\!] = [\![\mathcal{N}]\!]$, i.e. $\mathcal{S}$ is realizable.

Assume now that $\mathcal{S}$ is realizable: We have $[\![\mathcal{S}]\!] = [\![\mathcal{N}]\!]$. By construction, $[\![\mathcal{S}^\circ]\!] \subseteq [\![\mathcal{N}^\circ]\!]$. We can check $[\![\mathcal{N}^\circ]\!] \subseteq [\![\mathcal{S}^\circ]\!]$, hence $[\![\mathcal{N}^\circ]\!] = [\![\mathcal{S}^\circ]\!]$ and $\mathcal{S}^\circ$ is realizable. Let $\mathcal{K}^\circ$ be a process of $\mathcal{N}^\circ$ and u° be a linear extension of the rules occurring in $\mathcal{K}^\circ$. Then $u^\circ \in \text{FCS}(\mathcal{N}^\circ)$. Let $u = \pi(u^\circ)$. Then $u \in \text{FCS}(\mathcal{N})$. Since $[\![\mathcal{S}]\!] = [\![\mathcal{N}]\!]$ we have $u \simeq v$ for some $v \in \text{FCS}(\mathcal{S})$. By Prop. 2.13, there exists some $v^\circ \in \text{FCS}(\mathcal{S}^\circ)$ such that $u^\circ \simeq^\circ v^\circ$. Then $\mathcal{K}^\circ \in [\![v^\circ]\!]_{\mu^\circ_{\text{in}}}$ hence $\mathcal{K}^\circ \in [\![\mathcal{S}^\circ]\!]$. ∎

Finally we can consider now the problem of prefix-realizability. We call *terminating rule* each rule $\bot : M \to \emptyset$ for which $M \subseteq P^\circ$ is a subset of places such that $\pi(M) = P$. We denote by R_3 the set of all terminating rules and we put $R_\bot = R_0 \cup R_3$. We build from $\mathcal{S}^\circ$ the Petri net with states $\mathcal{S}^\circ_\bot = (Q, \imath, \longrightarrow_{\mathcal{S}^\circ_\bot}, \mu^\circ_{\text{in}})$ with the same set of states Q, the same initial state $\imath \in Q$, the same set of places P° and the same initial marking μ°_{in}. Each labeled arc from $\mathcal{S}^\circ$ appears in $\mathcal{S}^\circ_\bot$. For each terminating rule $r \in R_3$ and each state $q \in Q$ we add a self-loop $q \xrightarrow{r}_{\mathcal{S}^\circ_\bot} q$. Then we can check that $\mathcal{S}^\circ_\bot$ is prefix-realizable if and only if $\mathcal{S}^\circ$ is realizable. This leads us to the main result of this section.

THEOREM 2.15. *It is undecidable whether a prefix-bounded VASS is prefix-realizable.*

Proof. So far, we have proved that the PNS $\mathcal{S}$ is realizable if and only if the VASS $\mathcal{S}^\circ$ is realizable. We can prove that $\mathcal{S}^\circ_\bot$ is prefix-realizable if and only if $\mathcal{S}^\circ$ is realizable. Assume first that $\mathcal{S}^\circ_\bot$ is prefix-realizable. Then $\text{Pref}([\![\mathcal{S}^\circ_\bot]\!]) = [\![\mathcal{N}^\circ_\bot]\!]$ for some Petri net $\mathcal{N}^\circ_\bot$. Let $\mathcal{N}^\circ$ be the Petri net obtained from $\mathcal{N}^\circ_\bot$ by erasing all transitions corresponding to some terminating rule. It is clear that $[\![\mathcal{S}^\circ]\!] \subseteq [\![\mathcal{N}^\circ]\!]$. To prove that $\mathcal{S}^\circ$ is realizable, we simply check that $[\![\mathcal{N}^\circ]\!] \subseteq [\![\mathcal{S}^\circ]\!]$. Let $\mathcal{K} \in [\![\mathcal{N}^\circ]\!]$. We build the label causal net $\mathcal{K}_\bot$ from $\mathcal{K}$ by adding

an occurrence of some terminating rule $\bot : M \to \emptyset$. This requires that M coincides with the marking reached by $\mathcal{K}$. Then $\mathcal{K}_\bot$ is a process of $\mathcal{N}^\circ_\bot$. Hence $\mathcal{K}_\bot \in \mathrm{Pref}([\![\mathcal{S}^\circ_\bot]\!])$. Thus $\mathcal{K}_\bot$ is a prefix of some process $\mathcal{K}' \in [\![u']\!]_{\mu_{\mathrm{in}}}$ where $u' \in \mathrm{CS}(\mathcal{S}^\circ_\bot)$. Since the terminating rule $\bot : M \to \emptyset$ consumes all places from the marking reached by $\mathcal{K}$, it must be the last rule of u', and the single terminating rule of u', i.e. $u' = u \cdot (\bot : M \to \emptyset)$ for some $u \in \mathrm{CS}(\mathcal{S}^\circ)$. Hence $\mathcal{K}_\bot = \mathcal{K}'$. Therefore $\mathcal{K} \in [\![u]\!]_{\mu_{\mathrm{in}}}$ and $\mathcal{K} \in [\![\mathcal{S}^\circ]\!]$.

Assume now that $\mathcal{S}^\circ$ is realizable. Then $[\![\mathcal{S}^\circ]\!] = [\![\mathcal{N}^\circ]\!]$ for some Petri net $\mathcal{N}^\circ$. Adding to the Petri net $\mathcal{N}^\circ$ a transition for each terminating rule yields a new Petri net denoted by $\mathcal{N}^\circ_\bot$. We can check that $[\![\mathcal{S}^\circ_\bot]\!] = [\![\mathcal{N}^\circ_\bot]\!]$, so $\mathcal{S}^\circ_\bot$ is prefix-realizable. It is clear that $[\![\mathcal{S}^\circ_\bot]\!] \subseteq [\![\mathcal{N}^\circ_\bot]\!]$. Let $\mathcal{K} \in [\![\mathcal{N}^\circ_\bot]\!]$ and u be a linear extension of the partial order of rules in $\mathcal{K}$. Each terminating rule $\bot : M \to \emptyset$ may only occur in u as the last rule of u. Let v be the word obtained by removing the possible occurrence of some terminating rule $\bot : M \to \emptyset$ from u. Then v is a firable rule sequence of $\mathcal{N}^\circ$ because u is a firable rule sequence of $\mathcal{N}^\circ_\bot$. Since $[\![\mathcal{S}^\circ]\!] = [\![\mathcal{N}^\circ]\!]$, we have $[\![v]\!]_{\mu^\circ_{\mathrm{in}}} = [\![v']\!]_{\mu^\circ_{\mathrm{in}}}$ for some $v' \in \mathrm{FCS}(\mathcal{S}^\circ)$. Then $[\![u]\!]_{\mu^\circ_{\mathrm{in}}} = [\![u']\!]_{\mu^\circ_{\mathrm{in}}}$ for some $u' \in \mathrm{FCS}(\mathcal{S}^\circ_\bot)$ obtained from v' by adding possibly an occurrence some terminating rule. Therefore $\mathcal{K} \in [\![\mathcal{S}^\circ_\bot]\!]$. ∎

As an immediate consequence, we can now establish the following fact.

Corollary 2.16. *Given two prefix-bounded vector addition systems with states $\mathcal{S}_1$ and $\mathcal{S}_2$, it is undecidable whether $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$ (resp. $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = [\![\mathcal{S}_2]\!]$, $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = \mathrm{Pref}([\![\mathcal{S}_2]\!])$).*

Proof. By Proposition 2.3, a VASS $\mathcal{S}$ is realizable if and only if $[\![\mathcal{S}]\!] = [\![\mathcal{S}']\!]$ where $\mathcal{S}'$ is the VASS with a single state which admits a self-loop carrying r if r occurs in some firable computation sequence of $\mathcal{S}$. By Proposition 1.4, we can effectively build $\mathcal{S}'$ from $\mathcal{S}$. By Theorem 2.14, $[\![\mathcal{S}]\!] = [\![\mathcal{S}']\!]$ is undecidable. Therefore $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$ is undecidable for two vector addition systems with states $\mathcal{S}_1$ and $\mathcal{S}_2$.

Observe now that $[\![\mathcal{S}']\!] = \mathrm{Pref}([\![\mathcal{S}']\!])$. It follows that $[\![\mathcal{S}_1]\!] = \mathrm{Pref}([\![\mathcal{S}_2]\!])$ is undecidable for two given chemical rule systems $\mathcal{S}_1$ and $\mathcal{S}_2$.

Finally, $\mathcal{S}$ is prefix-realizable if and only if $\mathrm{Pref}([\![\mathcal{S}]\!]) = \mathrm{Pref}([\![\mathcal{S}']\!])$. It follows from Theorem 2.15 that $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = \mathrm{Pref}([\![\mathcal{S}_2]\!])$ is undecidable for two vector addition systems with states. ∎

The gap between vector addition systems with states and Petri nets is illustrated by the next result which shows that these decision problems are decidable if one considers (possibly unbounded) Petri nets only.

Proposition 2.17. *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two Petri nets. The property $[\![\mathcal{N}_1]\!] \subseteq [\![\mathcal{N}_2]\!]$ is decidable.*

Proof. Observe first that this property requires that $\mathcal{N}_1$ and $\mathcal{N}_2$ share the *same* initial marking. Let R_i be the set of rules occurring in some firable computation sequence of $\mathcal{N}_i$. The set R_i can be effectively computed (Prop. 1.4). Then $[\![\mathcal{N}_1]\!] \subseteq [\![\mathcal{N}_2]\!]$ if and only if $R_1 \subseteq R_2$. ∎

3 Checking reachability properties of process prefixes

Basic decision problems about the set of reachable markings of a Petri net are known to be decidable, namely boundedness, covering and reachability. Due to the simple simulation of a VASS by a Petri net recalled in Subsection 1.3 these results apply to the analysis of the reachable markings of a VASS or a PNS.

On the other hand, adopting a partial order semantics leads us to new difficulties and the model of VASSs is no longer equivalent to Petri nets. For instance the process language equality $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$ is

- *decidable* for two Petri nets $\mathcal{S}_1$ and $\mathcal{S}_2$, because one need simply to compare their initial markings and the two subsets of rules occurring in their firable rule sequences.
- *undecidable* for two prefix-bounded VASSs $\mathcal{S}_1$ and $\mathcal{S}_2$, because rational sets of Mazurkiewicz traces can be described by prefix-bounded VASSs.

Thus VASSs and Petri nets are no longer equivalent models under the process semantics.

In this section we investigate three basic verification problems about the set of markings reached by *prefixes* of processes: boundedness, covering and reachability. We show how to reduce these problems to the particular case of Petri nets in such a way that all complexity and decidability results extend from Petri nets to PNSs under the process semantics.

Definition 3.1. *A marking μ is* prefix-reachable *in a PNS $\mathcal{S}$ if there exists a prefix of a process of $\mathcal{S}$ which leads to the marking μ.*

Thus any reachable marking marking is prefix-reachable. In the particular case of Petri nets, conversely, any prefix-reachable marking is reachable, because the class of processes is prefix-closed. However the set of prefix-reachable markings can differ from the set of reachable markings in general. For instance, each process of the PNS from Fig. 1 leads to a marking with at most 3 tokens whereas prefixes of these processes lead to infinitely many distinct markings (see in Fig. 2 a prefix of a process which leads to a marking with 4 tokens).

The first basic problem we consider is the prefix-boundedness problem, which asks whether the set of prefix-reachable markings of a given PNS $\mathcal{S}$ is finite. We give below a linear construction of a PNS $\mathcal{S}^\circ$ from $\mathcal{S}$ such that $\mathcal{S}$ is prefix-bounded if and only if $\mathcal{S}^\circ$ is bounded. Since the boundedness of $\mathcal{S}^\circ$ boils down to the boundedness of a Petri net, we get that the prefix-boundedness problem for PNSs is computationally equivalent to the boundedness problem of Petri nets. Further we show that this technique apply to other similar basic problems about prefix-reachable markings, namely covering and reachability.

3.1 From Petri nets with states to Petri nets

Let $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{\text{in}})$ be a fixed PNS. We build a PNS $\mathcal{S}^\circ$ that allows us to analyse the set of prefix-reachable markings of $\mathcal{S}$. The construction of $\mathcal{S}^\circ$ from $\mathcal{S}$ is illustrated by Fig. 15. The PNS $\mathcal{S}^\circ$ makes use of three disjoint sets of places: $P_{\text{pre}}, P_{\text{suf}}, P_{\text{cut}}$ which are copies of the set of places P of $\mathcal{S}$. We let $\pi_{\text{pre}} : P \to P_{\text{pre}}$, $\pi_{\text{suf}} : P \to P_{\text{suf}}$, and $\pi_{\text{cut}} : P \to P_{\text{cut}}$ be

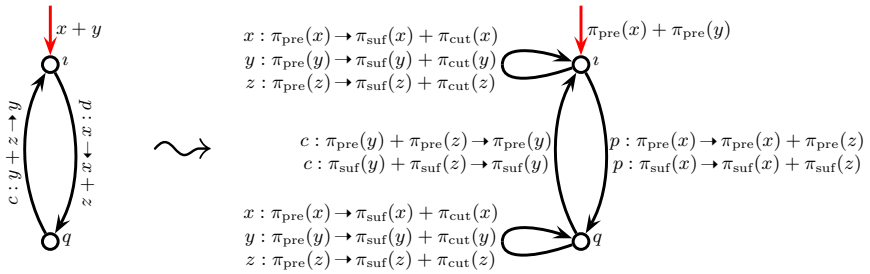

FIG. 15. Verification of prefix-reachable markings

the bijections that map each place from P to the corresponding place in P_{pre}, P_{cut} and P_{suf} respectively. These mappings extend naturally to mappings from multisets to multisets. The initial marking μ°_{in} of $\mathcal{S}^\circ$ is the multiset $\mu^\circ_{\mathrm{in}} = \pi_{\mathrm{pre}}(\mu_{\mathrm{in}})$.

The PNS $\mathcal{S}^\circ$ shares with $\mathcal{S}$ its set of states Q and its initial state $\imath$. It consists of three disjoint sets of labeled arcs: $\longrightarrow_{\mathrm{pre}}$, $\longrightarrow_{\mathrm{suf}}$, $\longrightarrow_{\mathrm{cut}}$. The restriction of $\mathcal{S}^\circ$ to the labeled arcs from $\longrightarrow_{\mathrm{pre}}$ and to the places from P_{pre} yields a PNS $\mathcal{S}^\circ_{\mathrm{pre}}$ isomorphic to $\mathcal{S}$. Thus for each labeled arc $q_1 \xrightarrow{r} q_2$ in $\mathcal{S}$ with $r = (a, {}^\bullet r, r^\bullet)$ there exists some labeled arc $q_1 \xrightarrow{s}_{\mathrm{pre}} q_2$ with $s = (a, \pi_{\mathrm{pre}}({}^\bullet r), \pi_{\mathrm{pre}}(r^\bullet))$. Similarly the restriction of $\mathcal{S}^\circ$ to the labeled arcs from $\longrightarrow_{\mathrm{suf}}$ and to the places from P_{suf} yields a PNS $\mathcal{S}^\circ_{\mathrm{suf}}$ isomorphic to $\mathcal{S}$, except that its initial marking is empty: For each labeled arc $q_1 \xrightarrow{r} q_2$ in $\mathcal{S}$ with $r = (a, {}^\bullet r, r^\bullet)$ there exists some labeled arc $q_1 \xrightarrow{s}_{\mathrm{suf}} q_2$ with $s = (a, \pi_{\mathrm{suf}}({}^\bullet r), \pi_{\mathrm{suf}}(r^\bullet))$. Note that the two PNSs $\mathcal{S}^\circ_{\mathrm{pre}}$ and $\mathcal{S}^\circ_{\mathrm{suf}}$ are synchronized because they share a common set of state. The set of labeled arcs $\longrightarrow_{\mathrm{cut}}$ consists of a self-loop $q \xrightarrow{s}_{\mathrm{cut}} q$ for each state q and each place $p \in P$; this labeled arc allows to move a token from the place $\pi_{\mathrm{pre}}(p)$ to the place $\pi_{\mathrm{suf}}(p)$ and to keep track of that transfer in the place $\pi_{\mathrm{cut}}(p)$, i.e. ${}^\bullet s = \pi_{\mathrm{pre}}(p)$ and $s^\bullet = \pi_{\mathrm{suf}}(p) + \pi_{\mathrm{cut}}(p)$. Note that tokens in P_{cut} cannot be consumed.

Intuitively, for any process $\mathcal{K}$ of $\mathcal{S}$ and for any prefix $\mathcal{K}'$ of $\mathcal{K}$, the PNS $\mathcal{S}^\circ$ can simulate a computation sequence of $\mathcal{S}$ which corresponds to $\mathcal{K}$ in such a way that each event from the prefix $\mathcal{K}'$ corresponds to the occurrence of a labeled arc from $\longrightarrow_{\mathrm{pre}}$ and each event from the suffix $\mathcal{K} \setminus \mathcal{K}'$ corresponds to the occurrence of a labeled arc from $\longrightarrow_{\mathrm{suf}}$. Moreover the set of places P_{cut} keeps track of the tokens transferred from $\mathcal{K}$ to $\mathcal{K}'$, i.e. from $\mathcal{S}^\circ_{\mathrm{pre}}$ to $\mathcal{S}^\circ_{\mathrm{suf}}$, by labeled arcs from $\longrightarrow_{\mathrm{cut}}$. Thus any prefix-reachable marking of $\mathcal{S}$ is represented by the restriction to $P_{\mathrm{pre}} \cup P_{\mathrm{cut}}$ of some reachable marking of $\mathcal{S}^\circ$. The key property of this representation, stated in Prop. 3.2 below, asserts that, conversely, each firable computation sequence of $\mathcal{S}^\circ$ corresponds to a process $\mathcal{K}$ of $\mathcal{S}$ and a prefix $\mathcal{K}'$ of $\mathcal{K}$ such that the marking of $P_{\mathrm{pre}} \cup P_{\mathrm{cut}}$ describes the marking reached by $\mathcal{K}'$.

In order to prove our results in details, we shall adopt also the next notations:

- We denote by τ_{pre} and τ_{suf} the bijections that maps each labeled arc $q \xrightarrow{r} q'$ from $\mathcal{S}$ to the corresponding labeled arc in $\longrightarrow_{\mathrm{pre}}$ and $\longrightarrow_{\mathrm{suf}}$ respectively.
- For each state $q \in Q$, we denote τ^q_{cut} the bijection between P and the self-loop labeled on q from $\longrightarrow_{\mathrm{cut}}$.

In the next statement and the sequel of this section, for each marking μ and for each subset of places X, we denote by $\mu|X$ the restriction of μ to the places from X. The main results of this section rely essentially on the next observation.

PROPOSITION 3.2. *A multiset of places $\mu \in \mathbb{N}^P$ is prefix-reachable in $\mathcal{S}$ if and only if there exists some reachable marking μ° of $\mathcal{S}^\circ$ such that $\mu = \pi_{pre}^{-1}(\mu^\circ|P_{pre}) + \pi_{cut}^{-1}(\mu^\circ|P_{cut})$.*

3.2 Proof of Proposition 3.2

For each rule sequence $u = r_1...r_n \in R^\star$, the cost of u is the vector $\mathrm{cost}(u) \in \mathbb{Z}^P$ such that $\mathrm{cost}(u)(p) = \sum_{i=1}^n {}^\bullet r_i(p) - r_i^\bullet(p)$ for each $p \in P$. In particular the cost of the empty rule sequence is the null vector. If a rule sequence u is firable from a marking μ then the marking reached by u from μ is $\mu - \mathrm{cost}(u)$. Let $\mathcal{K}$ be a process of a rule sequence u firable from μ. Then $\mu - \mathrm{cost}(u)$ is the marking reached by $\mathcal{K}$ from μ. Moreover the marking $\mu - \mathrm{cost}(u)$ corresponds to the set of conditions in $\mathcal{K}$ that are not input places of any event in $\mathcal{K}$ (which means intuitively that they are still available), i.e. to the *maximal places* of $\mathcal{K}$: We have $\mu - \mathrm{cost}(u) = \sum_{p \in \max(\mathcal{K}) \cap P} \pi(p)$.

For each rule r such that ${}^\bullet r \leqslant \mu - \mathrm{cost}(u)$, we let $\mathcal{K} \cdot r$ denote the class of labeled causal nets obtained by adding to $\mathcal{K}$ an event that describes an occurrence of rule r which consumes ${}^\bullet r$ available conditions from $\mathcal{K}$.

PROPOSITION 3.3. *Let $\mu \in \mathbb{N}^P$. The class of processes of a rule sequence $u \in R^\star$ satisfies the three following properties:*

- *If u is empty, i.e. $u = \varepsilon$, then each process from $[\![\varepsilon]\!]_\mu$ consists of $\sum_{p \in P} \mu(p)$ conditions and no event.*
- *for all rules $r \in R$, $[\![u.r]\!]_\mu$ is empty if $[\![u]\!]_\mu$ is empty or ${}^\bullet r \gneq \mu - cost(u)$.*
- *for all rules $r \in R$, if $[\![u]\!]_\mu$ is not empty and ${}^\bullet r \leqslant \mu - cost(u)$ then $[\![u.r]\!]_\mu$ collects all processes from $\mathcal{K} \cdot r$ for all processes $\mathcal{K} \in [\![u]\!]_\mu$.*

Proof. By Definition 1.5, a process of the empty rule sequence from a marking μ consists of a set of labeled places which represents μ. Consider a rule sequence u and a rule r. Assume that $[\![u.r]\!]_\mu$ is not empty. Let $\mathcal{K}$ be a labeled causal net from $[\![u.r]\!]_\mu$. We have already noticed that some prefix $\mathcal{K}'$ of $\mathcal{K}$ is a process of u from μ. Therefore $[\![u]\!]_\mu$ is not empty. Moreover $\mathcal{K}$ belongs to $\mathcal{K}' \cdot r$. Since $u.r$ is a firable rule sequence, ${}^\bullet r$ is smaller than the marking reached by u from μ, i.e. ${}^\bullet r \leqslant \mu - \mathrm{cost}(u)$. Thus, if $[\![u]\!]_\mu$ is empty or ${}^\bullet r > \mu - \mathrm{cost}(u)$ then $[\![u.r]\!]_\mu$ is empty. On the other hand, if $[\![u]\!]_\mu$ is not empty and ${}^\bullet r \leqslant \mu - \mathrm{cost}(u)$ then any labeled causal net from $\mathcal{K}' \cdot r$ where $\mathcal{K}' \in [\![u]\!]_\mu$ is clearly a process of $u.r$ from μ. Further any process from $[\![u.r]\!]_\mu$ can be obtained in this way. ∎

Given two multisets of places $\mu_1, \mu_2 \in \mathbb{N}^P$, the maximum $\max(\mu_1, \mu_2)$ collects the maximal number of tokens in each place: $\max(\mu_1, \mu_2)(p) = \max(\mu_1(p), \mu_2(p))$ for each $p \in P$. We will make use of the following *requirement function* $\mathrm{req} : R^\star \to \mathbb{N}^P$.

DEFINITION 3.4. *The* requirement *of a rule sequence $u \in R^\star$ is the multiset of places defined inductively as follows:*

- $req(\varepsilon) = 0$,
- $req(u.a) = \max(req(u), {}^\bullet a + cost(u))$ *for all* $u \in R^\star$ *and all* $a \in R$.

The next observation shows that the requirement of a rule sequence u is the minimal marking μ such that u is firable from μ, i.e. $[\![u]\!]_\mu \neq \emptyset$.

PROPOSITION 3.5. *Let* $u \in R^\star$ *and* $\mu \in \mathbb{N}^P$. *Then* $[\![u]\!]_\mu \neq \emptyset$ *if and only if* $\mu \geqslant req(u)$.

Proof. We proceed by induction over the length of u. If $u = \varepsilon$ then $\mathrm{req}(u) = 0$ hence $\mu \geqslant \mathrm{req}(u)$. Moreover $[\![\varepsilon]\!]_\mu$ contains each labeled causal net with no event and with a set of conditions representing the marking μ. Induction step: Let $u \in R^\star$ and $a \in R$. Assume first that $[\![u.a]\!]_\mu \neq \emptyset$. By Prop. 3.3, we have $\mu \geqslant {}^\bullet a + \mathrm{cost}(u)$. On the other hand, we have $[\![u]\!]_\mu \neq \emptyset$ hence $\mu \geqslant \mathrm{req}(u)$ by induction hypothesis. Thus $\mu \geqslant \max(\mathrm{req}(u), {}^\bullet a + \mathrm{cost}(u)) = \mathrm{req}(u.a)$. Assume now that $[\![u.a]\!]_\mu = \emptyset$. We distinguish two cases.

1. $[\![u]\!]_\mu = \emptyset$. Then, by induction hypothesis, we have $\mu < \mathrm{req}(u) \leqslant \mathrm{req}(u.a)$.
2. $[\![u]\!]_\mu \neq \emptyset$. Then, by Prop. 3.3, we have $\mu < {}^\bullet a + \mathrm{cost}(u) \leqslant \max(\mathrm{req}(u), {}^\bullet a + \mathrm{cost}(u)) = \mathrm{req}(u.a)$.

Thus $[\![u.a]\!]_\mu \neq \emptyset$ if and only if $\mu \geqslant \mathrm{req}(u.a)$. ∎

For each rule sequence $u = r_1 ... r_n \in R^\star$ firable from μ_{in}, we let μ_u denote the marking reached by u from μ_{in}, i.e. $\mu_u = \mu_{\mathrm{in}} + \sum_{i=1}^{n} (r_i{}^\bullet - {}^\bullet r_i)$. Similarly for each transition sequence $s \in T_{\mathcal{N}}^\star$ firable from the initial marking μ_{in}°, μ_s° denotes the marking reached by s in $\mathcal{S}^\circ$.

We shall use the following notion of partial computation: A *partial computation* is a triple $(u, v, w) \in R^\star \times R^\star \times R^\star$ such that $[\![v.w]\!]_{\mu_{\mathrm{in}}} \cap [\![u]\!]_{\mu_{\mathrm{in}}} \neq \emptyset$ and $u \in \mathrm{CS}(\mathcal{S})$. Then $[\![v]\!]_{\mu_{\mathrm{in}}} \neq \emptyset$ hence the rule sequence v is firable from μ_{in}. A partial computation is used as a witness for a process $\mathcal{K}_u$ of u and a prefix $\mathcal{K}_v$ of $\mathcal{K}_u$ with $\mathcal{K}_v \in [\![v]\!]_{\mu_{\mathrm{in}}}$. Note that v need not to be a prefix of u, nor to be a computation sequence of $\mathcal{S}$. Partial computations are closely related to prefix-reachable markings, as the next basic observation shows.

PROPOSITION 3.6. *For each partial computation* (u, v, w), *the marking* μ_v *is prefix-reachable. Conversely, for any prefix-reachable marking* μ, *there exists some partial computation* (u, v, w) *such that* $\mu = \mu_v$.

Proof. Let (u, v, w) be a partial computation: There exists some labeled causal net $\mathcal{K}$ such that $\mathcal{K} \in [\![v.w]\!]_{\mu_{\mathrm{in}}} \cap [\![u]\!]_{\mu_{\mathrm{in}}}$. By Prop. 3.3, $\mathcal{K}$ may be built from a causal net $\mathcal{K}_v \in [\![v]\!]_{\mu_{\mathrm{in}}}$ by adding the sequence of rules w, one after the other. Thus $\mathcal{K}_v$ is a prefix of $\mathcal{K}$ and μ_v is prefix-reachable.

Let $\mathcal{K} \in [\![u]\!]_{\mu_{\mathrm{in}}}$ with $u \in \mathrm{CS}(\mathcal{S})$ and $\mathcal{K}'$ be a prefix of $\mathcal{K}$. Let v be a linear extension of the partial order of rules occurring in $\mathcal{K}'$. Then $\mathcal{K}' \in [\![v]\!]_{\mu_{\mathrm{in}}}$ and μ_v is the marking reached by $\mathcal{K}'$. Let w be a linear extension of the partial order of rules occurring in the suffix $\mathcal{K} \setminus \mathcal{K}'$. Then $v.w$ is a linear extension of the partial order of rules occurring in $\mathcal{K}$ hence $\mathcal{K} \in [\![v.w]\!]_{\mu_{\mathrm{in}}}$. Therefore (u, v, w) is a partial computation. ∎

LEMMA 3.7. *Let* (u, v, w) *be a partial computation and* $a \in R$ *be a rule such that* ${}^\bullet a \leqslant \mu_u$. *If* $u.a \in CS(\mathcal{S})$ *then* $(u.a, v, w.a)$ *is a partial computation.*

Proof. Let $\mathcal{K}$ be a labeled causal net from $[\![v.w]\!]_{\mu_{\text{in}}} \cap [\![u]\!]_{\mu_{\text{in}}}$. Since ${}^\bullet a \leqslant \mu_u$, the class $[\![u.a]\!]_{\mu_{\text{in}}}$ is not empty (Prop. 3.3). Moreover all causal nets from $[\![u.a]\!]_{\mu_{\text{in}}}$ are obtained from some causal net from $[\![u]\!]_{\mu_{\text{in}}}$ by adding an occurrence of rule a. In particular we can add an occurrence of rule a to $\mathcal{K}$ and get a causal net from $[\![u.a]\!]_{\mu_{\text{in}}}$. The latter is also a causal net from $[\![v.w.a]\!]_{\mu_{\text{in}}}$ since $\mathcal{K} \in [\![v.w]\!]_{\mu_{\text{in}}}$. ∎

The proof of Prop. 3.2 relies on the two next technical lemmas which can be established by means of a bit tedious inductions. The first one asserts that for each firable computation sequence $u \in \text{FCS}(\mathcal{S})$ and each prefix $\mathcal{K}_v$ of each process $\mathcal{K}_u \in [\![u]\!]_{\mu_{\text{in}}}$, the VASS $\mathcal{S}^\circ$ can be guided in order to simulate each rule of u in its sequential order so that the marking reached by u is described by the current marking of $P_{\text{pre}} \cup P_{\text{suf}}$ while the marking reached by $\mathcal{K}_v$ is described by the current marking of $P_{\text{pre}} \cup P_{\text{cut}}$. Furthermore we have to make sure that the state $q \in Q$ reached by u is also reached by s in $\mathcal{S}^\circ$ and to check that all events from $\mathcal{K}_u$ that do not occur in $\mathcal{K}_v$ are performed by transitions from $\longrightarrow_{\text{suf}}$. To do so, we have to guide $\mathcal{S}^\circ$ to transfer *exactly* the required number of tokens from P_{pre} to P_{suf}, which corresponds to the marking of P_{cut}.

LEMMA 3.8. *Let (u, v, w) be a partial computation in $\mathcal{S}$ and q be some state such that $\imath \xrightarrow{u} q$ in $\mathcal{S}$. There exists some firable rule sequence s in $\mathcal{S}^\circ$ which leads to the marking μ_s° such that*

(a) $\pi_{cut}^{-1}(\mu_s^\circ|P_{cut}) + \pi_{pre}^{-1}(\mu_s^\circ|P_{pre}) = \mu_v$,
(b) $\pi_{suf}^{-1}(\mu_s^\circ|P_{suf}) + \pi_{pre}^{-1}(\mu_s^\circ|P_{pre}) = \mu_u$,
(c) $\pi_{cut}^{-1}(\mu_s^\circ|P_{cut}) = req(w)$,
(d) $\imath \xrightarrow{s} q$ *in* $\mathcal{S}^\circ$.

Proof. We proceed by induction over the length of u. If $|u| = 0$ then $u = \varepsilon$, $q = \imath$, $v = \varepsilon$ and $w = \varepsilon$. The empty firing sequence of $\mathcal{S}^\circ$ satisfies the four above properties because $\mu_{\text{in}}^\circ = \pi_{\text{pre}}(\mu_{\text{in}})$. Induction step: We consider some partial computation (u', v', w') with $|u'| = n + 1$. We put $u' = u.a$ with $|u| = n$ and $\imath \xrightarrow{u} q \xrightarrow{a} q'$. Let $\mathcal{K}_{u'}$, $\mathcal{K}_{v'}$ and $\mathcal{K}_{w'}$ be three labeled causal nets such that $\mathcal{K}_{u'} \in [\![u']\!]_{\mu_{\text{in}}} \cap [\![v'.w']\!]_{\mu_{\text{in}}}$, $\mathcal{K}_{v'} \in [\![v']\!]_{\mu_{\text{in}}}$ is a prefix of $\mathcal{K}_{u'}$ and $\mathcal{K}_{w'} \in [\![w']\!]_{\mu_{v'}}$ is the corresponding suffix. We know that $\mathcal{K}_{u'}$ is obtained from some process $\mathcal{K}_u \in [\![u]\!]_{\mu_{\text{in}}}$ by adding an event e_a that corresponds to an occurrence of rule a. We distinguish two cases.

1. Event e_a occurs in $\mathcal{K}_{w'}$. Then there is some rule sequence w such that $\mathcal{K}_{w'} \in [\![w.a]\!]_{\mu_{v'}}$ and $\mathcal{K}_u \in [\![v'.w]\!]_{\mu_{\text{in}}}$ so (u, v', w) is a partial computation of length n. By induction hypothesis there exists some firable computation sequence s in $\mathcal{S}^\circ$ such that the four above properties are satisfied. In particular, $\pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) = \text{req}(w)$. Moreover Prop. 3.5 ensures that $\text{req}(w.a) \leqslant \mu_{v'}$ because $\mathcal{K}_{w'} \in [\![w.a]\!]_{\mu_{v'}}$. Furthermore we have on one hand $\pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) = \text{req}(w) \leqslant \text{req}(w.a)$; and on the other hand $\text{req}(w.a) \leqslant \mu_{v'}$ i.e. $\text{req}(w.a) \leqslant \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}})$. Therefore

$$\pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) \leqslant \text{req}(w.a) \leqslant \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}})$$

It follows that there is some multiset of places $\hat{\mu} \in \mathbb{N}^P$ such that $\pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \hat{\mu} = \text{req}(w.a)$ with $\pi_{\text{pre}}(\hat{\mu}) \leqslant \mu_s^\circ|P_{\text{pre}}$. We consider a sequence of transitions $A \in \longrightarrow_{\text{cut}}^\star$ in $\mathcal{S}^\circ$ which consumes the multiset of tokens $\pi_{\text{pre}}(\hat{\mu})$ and produces the multiset of tokens $\pi_{\text{suf}}(\hat{\mu})$. We have

$$^\bullet a + \mu_v - \mu_u = {}^\bullet a + \text{cost}(w) \leqslant \max(\text{req}(w), {}^\bullet a + \text{cost}(w)) = \text{req}(w.a)$$

Hence $\mu_u - \mu_v + \text{req}(w.a) \geqslant {}^\bullet a$. On the other hand

$$\mu_u - \mu_v + \text{req}(w.a) = \mu_u - \mu_v + \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \hat{\mu} = \mu_u - \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) + \hat{\mu} = \pi_{\text{suf}}^{-1}(\mu_s^\circ|P_{\text{suf}}) + \hat{\mu}$$

Hence $\pi_{\text{suf}}^{-1}(\mu_{s.A}^\circ|P_{\text{suf}}) = \pi_{\text{suf}}^{-1}(\mu_s^\circ|P_{\text{suf}}) + \hat{\mu} \geqslant {}^\bullet a$. It follows that $s.A.\tau_{\text{suf}}(a)$ is a firable computation sequence of $\mathcal{S}^\circ$. The latter satisfies the required conditions:

(a) $\pi_{\text{pre}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{pre}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) - \hat{\mu}$ and $\pi_{\text{cut}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{cut}}) = \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \hat{\mu}$ hence $\pi_{\text{pre}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{cut}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) = \mu_{v'}$.

(b) $\pi_{\text{pre}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{pre}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) - \hat{\mu}$, $\pi_{\text{suf}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{suf}}) = \pi_{\text{suf}}^{-1}(\mu_s^\circ|P_{\text{suf}}) + \hat{\mu} - {}^\bullet a + a^\bullet$ therefore $\pi_{\text{pre}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{pre}}) + \pi_{\text{suf}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{suf}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) + \pi_{\text{suf}}^{-1}(\mu_s^\circ|P_{\text{suf}}) - {}^\bullet a + a^\bullet = \mu_{u.a} = \mu_{u'}$.

(c) $\pi_{\text{cut}}^{-1}(\mu_{s.A.\tau_{\text{suf}}(a)}^\circ|P_{\text{cut}}) = \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \hat{\mu} = \text{req}(w.a)$.

(d) $\imath \xrightarrow{s} q \xrightarrow{A} q'' \xrightarrow{\tau_{\text{suf}}(a)} q'$.

2. Event e_a occurs in $\mathcal{K}_{v'}$. Then there exists some v such that $\mathcal{K}_{v'} \in [\![v.a]\!]_{\mu_{\text{in}}}$ and $\mathcal{K}_u \in [\![v.w']\!]_{\mu_{\text{in}}}$. It follows that (u, v, w') is a partial computation of length n. By induction hypothesis, there exists some firable computation sequence s in $\mathcal{S}^\circ$ such that the four above properties are satisfied. Then we have

$$^\bullet a \leqslant \mu_v - \text{req}(w') = \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) - \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}).$$

Therefore $\tau_{\text{pre}}(a)$ can be fired from the marking μ_s° and we get a new firable computation sequence $s.\tau_{\text{pre}}(a)$. The latter fulfills the required properties.

(a) We have $\pi_{\text{cut}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{cut}}) = \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}})$ and $\pi_{\text{pre}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{pre}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) - {}^\bullet a + a^\bullet$, hence $\pi_{\text{cut}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{pre}}) = \mu_{v.a}$.

(b) Since $\pi_{\text{suf}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{suf}}) = \pi_{\text{suf}}^{-1}(\mu_s^\circ|P_{\text{suf}})$ and $\pi_{\text{pre}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{pre}}) = \pi_{\text{pre}}^{-1}(\mu_s^\circ|P_{\text{pre}}) - {}^\bullet a + a^\bullet$ we get $\pi_{\text{suf}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{suf}}) + \pi_{\text{pre}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{pre}}) = \mu_{u.a} = \mu_{u'}$.

(c) $\pi_{\text{cut}}^{-1}(\mu_{s.\tau_{\text{pre}}(a)}^\circ|P_{\text{cut}}) = \pi_{\text{cut}}^{-1}(\mu_s^\circ|P_{\text{cut}}) = \text{req}(w')$.

(d) $\imath \xrightarrow{s} q \xrightarrow{\tau_{\text{pre}}(a)} q'$.

■

LEMMA 3.9. *Let (u, v, w) be a partial computation and $a \in R$ be a rule such that ${}^\bullet a + \text{req}(w) \leqslant \mu_v$. If $u.a \in CS(\mathcal{S})$ then $(u.a, v.a, w)$ is a partial computation.*

Proof. Let $\mathcal{K}_u \in [\![u]\!]_{\mu_{\text{in}}}$ and let $\mathcal{K}_v \in [\![v]\!]_{\mu_{\text{in}}}$ be a prefix of $\mathcal{K}_u$. Since w is firable from $\text{req}(w)$, there exists some labeled causal net $\mathcal{K}_w \in [\![w]\!]_{\text{req}(w)}$. We can build a process $\mathcal{K}'_u$

of u which admits $\mathcal{K}_v$ as prefix and such that the suffix $\mathcal{K}'_u \setminus \mathcal{K}_v$ corresponds to $\mathcal{K}_w$. In particular only $\mathrm{req}(w)$ tokens from the multiset μ_v reached by $\mathcal{K}_v$ are consumed by $\mathcal{K}_w$. Then we can add an occurrence of rule a to $\mathcal{K}'_u$ which consumes ${}^\bullet a$ remaining tokens from $\mu_v - \mathrm{req}(w)$. ∎

Conversely we need to show that the marking of $P_{\mathrm{pre}} \cup P_{\mathrm{cut}}$ reached after any firable transition sequence s of $\mathcal{S}^\circ$ corresponds to a prefix-reachable marking of $\mathcal{S}$, i.e. to some partial computation (u, v, w). To do so, we have to build a firable rule sequence $u \in \mathrm{FCS}(\mathcal{S})$, a process $\mathcal{K}_u \in [\![u]\!]_{\mu_{\mathrm{in}}}$ and a prefix $\mathcal{K}_v \in [\![v]\!]_{\mu_{\mathrm{in}}}$ inductively from s. At each step the state reached by s coincides with the state reached by u. When $\mathcal{S}^\circ$ applies an additional labeled arc a, the corresponding partial computation is either (u, v, w) if $a \in \xrightarrow{r}_{\mathrm{cut}}$; or $(u.r, v, w.r)$ if $a \in \xrightarrow{r}_{\mathrm{suf}}$; or $(u.r, v.r, w)$ if $a \in \xrightarrow{r}_{\mathrm{pre}}$. In this last case, the rule r and the sequence of rules w can be performed concurrently: Formally we shall establish that ${}^\bullet r + \mathrm{req}(w) \leqslant \mu_v$. This property follows actually from the fact that w can be fired from the marking obtained by the tokens transferred from P_{pre} to P_{suf}, i.e. $\pi_{\mathrm{cut}}(\mathrm{req}(w)) \leqslant \mu^\circ_s | P_{\mathrm{cut}}$.

Lemma 3.10. *Let s be a firable rule sequence in $\mathcal{S}^\circ$ leading to the state q and the marking μ°_s. There exists some partial computation (u, v, w) of $\mathcal{S}$ such that*

(a) $\pi^{-1}_{cut}(\mu^\circ_s|P_{cut}) + \pi^{-1}_{pre}(\mu^\circ_s|P_{pre}) = \mu_v$,
(b) $\pi^{-1}_{suf}(\mu^\circ_s|P_{suf}) + \pi^{-1}_{pre}(\mu^\circ_s|P_{pre}) = \mu_u$,
(c) $\pi^{-1}_{cut}(\mu^\circ_s|P_{cut}) \geqslant req(w)$, *and*
(d) $\imath \xrightarrow{u} q$ *in* $\mathcal{S}$.

Proof. We proceed by induction over the length of s. If $|s| = 0$ then $s = \varepsilon$; the empty partial computation $(\varepsilon, \varepsilon, \varepsilon)$ satisfies the four requirements because $\mu^\circ_{\mathrm{in}} = \pi_{\mathrm{pre}}(\mu_{\mathrm{in}})$. Induction step: Let $s.a$ be a firable computation sequence of length $n + 1$. By induction hypothesis, there exists some partial computation (u, v, w) which fulfills the four above requirements. We distinguish three cases:

1. $q \xrightarrow{a}_{\mathrm{cut}} q$ in $\mathcal{S}^\circ$. Then we can check that (u, v, w) satisfies the four requirements for $s.a$.
   (a) $\pi^{-1}_{\mathrm{cut}}(\mu^\circ_{s.a}|P_{\mathrm{cut}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_{s.a}|P_{\mathrm{pre}}) = \pi^{-1}_{\mathrm{cut}}(\mu^\circ_s|P_{\mathrm{cut}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_s|P_{\mathrm{pre}}) = \mu_v$.
   (b) $\pi^{-1}_{\mathrm{suf}}(\mu^\circ_{s.a}|P_{\mathrm{suf}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_{s.a}|P_{\mathrm{pre}}) = \pi^{-1}_{\mathrm{suf}}(\mu^\circ_s|P_{\mathrm{suf}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_s|P_{\mathrm{pre}}) = \mu_u$.
   (c) We have $\pi^{-1}_{\mathrm{cut}}(\mu^\circ_{s.a}|P_{\mathrm{cut}}) > \pi^{-1}_{\mathrm{cut}}(\mu^\circ_s|P_{\mathrm{cut}}) \geqslant \mathrm{req}(w)$.
   (d) $\imath \xrightarrow{u} q$ by induction hypothesis.
2. $q \xrightarrow{a}_{\mathrm{suf}} q'$. Then $\tau^{-1}_{\mathrm{suf}}(a)$ is an arc $q \xrightarrow{r}_{\mathcal{S}} q'$ in $\mathcal{S}$ and $u.r$ is a computation sequence of $\mathcal{S}$. Moreover $\pi^{-1}_{\mathrm{suf}}({}^\bullet a|P_{\mathrm{suf}}) = {}^\bullet r$ and $\pi^{-1}_{\mathrm{suf}}(a^\bullet|P_{\mathrm{suf}}) = r^\bullet$. Furthermore ${}^\bullet a \leqslant \mu^\circ_s$, hence ${}^\bullet r \leqslant \pi^{-1}_{\mathrm{suf}}(\mu^\circ_s|P_{\mathrm{suf}}) \leqslant \mu_u$. By Lemma 3.7 we know that $(u.r, v, w.r)$ is a partial computation of $\mathcal{S}$. Moreover
   (a) $\pi^{-1}_{\mathrm{cut}}(\mu^\circ_{s.a}|P_{\mathrm{cut}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_{s.a}|P_{\mathrm{pre}}) = \pi^{-1}_{\mathrm{cut}}(\mu^\circ_s|P_{\mathrm{cut}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_s|P_{\mathrm{pre}}) = \mu_v$.
   (b) $\pi^{-1}_{\mathrm{suf}}(\mu^\circ_{s.a}|P_{\mathrm{suf}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_{s.a}|P_{\mathrm{pre}}) = \pi^{-1}_{\mathrm{suf}}(\mu^\circ_s|P_{\mathrm{suf}}) + \pi^{-1}_{\mathrm{pre}}(\mu^\circ_s|P_{\mathrm{pre}}) + \pi^{-1}_{\mathrm{suf}}(a^\bullet - {}^\bullet a|P_{\mathrm{suf}}) = \mu_u - {}^\bullet r + r^\bullet = \mu_{u.r}$.
   (c) We have $\mu_{s.a}|P_{\mathrm{cut}} = \mu_s|P_{\mathrm{cut}}$. Moreover ${}^\bullet a \leqslant \mu^\circ_s$, hence

   $$\pi^{-1}_{\mathrm{cut}}(\mu^\circ_s|P_{\mathrm{cut}}) \geqslant \pi^{-1}_{\mathrm{cut}}(\mu^\circ_s|P_{\mathrm{cut}}) + \pi^{-1}_{\mathrm{suf}}({}^\bullet a - \mu^\circ_s|P_{\mathrm{suf}}) = \pi^{-1}_{\mathrm{suf}}({}^\bullet a|P_{\mathrm{suf}}) + (\mu_v - \mu_u) = {}^\bullet r + \mathrm{cost}(w)$$

   Then $\pi^{-1}_{\mathrm{cut}}(\mu^\circ_{s.a}|P_{\mathrm{cut}}) = \pi^{-1}_{\mathrm{cut}}(\mu^\circ_s|P_{\mathrm{cut}}) \geqslant \max(\mathrm{req}(w), {}^\bullet r + \mathrm{cost}(w)) = \mathrm{req}(w.r)$.

(d) $\imath \xrightarrow{u} q \xrightarrow{r} q'$ and $\imath \xrightarrow{s} q \xrightarrow{a} q'$.

3. $q \xrightarrow{a}_{\mathrm{pre}} q'$. Then $\tau_{\mathrm{pre}}^{-1}(a)$ is an arc $q \xrightarrow{r}_{\mathcal{S}} q'$. Moreover $\pi_{\mathrm{pre}}^{-1}({}^\bullet a|P_{\mathrm{pre}}) = {}^\bullet r$ and $\pi_{\mathrm{pre}}^{-1}(a^\bullet|P_{\mathrm{pre}}) = r^\bullet$. We observe first that
   (a) $\pi_{\mathrm{cut}}^{-1}(\mu_{s.a}^\circ|P_{\mathrm{cut}}) + \pi_{\mathrm{pre}}^{-1}(\mu_{s.a}^\circ|P_{\mathrm{pre}}) = \mu_v - \pi_{\mathrm{pre}}^{-1}({}^\bullet a|P_{\mathrm{pre}}) + \pi_{\mathrm{pre}}^{-1}(a^\bullet|P_{\mathrm{pre}}) = \mu_{v.r}$
   (b) $\pi_{\mathrm{suf}}^{-1}(\mu_{s.a}^\circ|P_{\mathrm{suf}}) + \pi_{\mathrm{pre}}^{-1}(\mu_{s.a}^\circ|P_{\mathrm{pre}}) = \mu_u - \pi_{\mathrm{pre}}^{-1}({}^\bullet a|P_{\mathrm{pre}}) + \pi_{\mathrm{pre}}^{-1}(a^\bullet|P_{\mathrm{pre}}) = \mu_{u.r}$
   (c) $\pi_{\mathrm{cut}}^{-1}(\mu_{s.a}^\circ|P_{\mathrm{cut}}) = \pi_{\mathrm{cut}}^{-1}(\mu_s^\circ|P_{\mathrm{cut}}) \geqslant \mathrm{req}(w)$
   (d) $q \xrightarrow{r} q'$ in $\mathcal{S}$.

   Since $\mu_s^\circ \geqslant {}^\bullet a$, we have $\pi_{\mathrm{pre}}^{-1}(\mu_s^\circ|P_{\mathrm{pre}}) \geqslant \pi_{\mathrm{pre}}^{-1}({}^\bullet a|P_{\mathrm{pre}})$. On the other hand, $\pi_{\mathrm{cut}}^{-1}(\mu_s^\circ|P_{\mathrm{cut}}) \geqslant \mathrm{req}(w)$, hence $\pi_{\mathrm{pre}}^{-1}(\mu_s^\circ|P_{\mathrm{pre}}) + \pi_{\mathrm{cut}}^{-1}(\mu_s^\circ|P_{\mathrm{cut}}) \geqslant \pi_{\mathrm{pre}}^{-1}({}^\bullet a|P_{\mathrm{pre}}) + \mathrm{req}(w)$, i.e $\mu_v \geqslant {}^\bullet r + \mathrm{req}(w)$. By Lemma 3.9, $(u.r, v.r, w)$ is a partial computation. ∎

We are now ready to prove Prop. 3.2. Let μ be the marking reached by a prefix $\mathcal{K}'$ of a process $\mathcal{K} \in [\![\mathcal{S}]\!]$. According to Prop. 3.6, there exists some partial computation (u, v, w) such that $\mu_v = \mu$. By Lemma 3.8, there exists some firable rule sequence s in $\mathcal{S}^\circ$ such that $\pi_{\mathrm{cut}}^{-1}(\mu_s^\circ|P_{\mathrm{cut}}) + \pi_{\mathrm{pre}}^{-1}(\mu_s^\circ|P_{\mathrm{pre}}) = \mu_v = \mu$. Conversely if $\pi_{\mathrm{cut}}^{-1}(\mu_s^\circ|P_{\mathrm{cut}}) + \pi_{\mathrm{pre}}^{-1}(\mu_s^\circ|P_{\mathrm{pre}}) = \mu$ for some firable rule sequence s in $\mathcal{S}^\circ$ then Lemma 3.10 ensures that there exists some partial computation (u, v, w) such that $\pi_{\mathrm{cut}}^{-1}(\mu_s^\circ|P_{\mathrm{cut}}) + \pi_{\mathrm{pre}}^{-1}(\mu_s^\circ|P_{\mathrm{pre}}) = \mu_v$. Moreover Prop. 3.6 asserts that μ_v is the marking reached by some prefix $\mathcal{K}'$ of some process $\mathcal{K} \in [\![\mathcal{S}]\!]$.

3.3 Analysis of prefix-reachable markings

At present we make use of the properties of the PNS $\mathcal{S}^\circ$, in particular Prop. 3.2, in order to derive some techniques to analyse the set of prefix-reachable markings of $\mathcal{S}$. First, the *prefix-boundedness problem* asks whether the set of prefix-reachable markings of a given PNS $\mathcal{S}$ is finite. It is easy to prove that the PNS $\mathcal{S}$ is prefix-bounded if and only if the PNS $\mathcal{S}^\circ$ is bounded, which can be checked by means of the usual linear simulation by a Petri net. Moreover, prefix-boundedness is equivalent to boundedness in the particular case of Petri nets because the set of processes of a Petri net is closed by prefixes. Thus,

THEOREM 3.11. *The prefix-boundedness problem of PNSs is computationally equivalent to the boundedness problem of Petri nets.*

Proof. It is clear from Prop. 3.2 that if $\mathcal{S}^\circ$ is bounded then there are finitely many prefix-reachable markings in $\mathcal{S}$. Conversely, assume that $\mathcal{S}$ is prefix-bounded. Then there exists some $M \in \mathbb{N}$ such that $\mu_v(p) \leqslant M$ and $\mu_u(p) \leqslant M$ for all partial computations (u, v, w) of $\mathcal{S}$ and all places $p \in P$. Then Lemma 3.10 ensures that if a firable rule sequence of $\mathcal{S}^\circ$ leads to some marking μ then $\mu(p) \leqslant M$ for each place $p \in P_{\mathcal{S}^\circ}$. ∎

Second, the *prefix-covering problem* asks whether a given multiset of places $\mu \in \mathbb{N}^P$ is covered by some prefix-reachable marking $\mu' \in \mathbb{N}^P$, i.e. $\mu(p) \leqslant \mu'(p)$ for all $p \in P$. It is easy to see that μ is prefix-covered in $\mathcal{S}$ if and only if the multiset of places $\pi_{\mathrm{cut}}(\mu)$ is covered by some reachable marking of $\mathcal{S}^\circ$. Thus,

THEOREM 3.12. *The prefix-covering problem for PNSs is computationally equivalent to the covering problem in Petri nets.*

Proof. The prefix-covering problem is equivalent to the covering problem in the particular case of Petri nets. Thus the prefix-covering problem of PNSs is as difficult as the covering problem of Petri nets. Moreover checking the coverability of a given marking by the reachable markings of a PNS can be checked by means of the linear simulation of a PNS by a Petri net. Thus we need simply to show that a marking μ_0 is prefix-covered in $\mathcal{S}$ if and only if the multiset of places $\pi_{\mathrm{cut}}(\mu_0)$ is covered by some reachable marking of $\mathcal{S}^\circ$.

Let $\mu_0 \in \mathbb{N}^P$ be a multiset of places. Assume first that μ_0 is covered by some prefix-reachable marking μ of $\mathcal{S}$: $\mu_0 \leqslant \mu$. By Prop. 3.2, there exists some reachable marking μ° in $\mathcal{S}^\circ$ such $\mu = \pi_{\mathrm{pre}}^{-1}(\mu^\circ|P_{\mathrm{pre}}) + \pi_{\mathrm{cut}}^{-1}(\mu^\circ|P_{\mathrm{cut}})$. Due to the self-loop labeled arcs from $\longrightarrow_{\mathrm{cut}}$ in $\mathcal{S}^\circ$, we can assume that $\mu^\circ|P_{\mathrm{pre}} = 0$ hence $\pi_{\mathrm{cut}}(\mu) = \mu^\circ|P_{\mathrm{cut}}$ and $\pi_{\mathrm{cut}}(\mu_0) \leqslant \mu^\circ|P_{\mathrm{cut}}$.

Conversely, assume now that $\pi_{\mathrm{cut}}(\mu_0) \leqslant \mu^\circ$ for some reachable marking μ° of $\mathcal{S}^\circ$. Then, by Lemma 3.10 and Prop. 3.6, $\mu_0 \leqslant \pi_{\mathrm{cut}}^{-1}(\mu^\circ|P_{\mathrm{cut}}) \leqslant \mu$ for some prefix-reachable marking μ of $\mathcal{S}$. ∎

Last but not least, the *prefix-reachability problem* asks whether a given multiset of places is prefix-reachable in $\mathcal{S}$. Let us consider a slight modification $\mathcal{S}'$ of $\mathcal{S}^\circ$ where for each place $p \in P_{\mathrm{suf}}$, each state $q \in \mathcal{S}^\circ$ is provided with an additional self-loop labeled arc which carries a rule that consumes a token from p and produces nothing. Then a multiset μ of places is prefix-reachable in $\mathcal{S}$ if and only if $\pi_{\mathrm{cut}}(\mu)$ is reachable in $\mathcal{S}'$. Thus,

THEOREM 3.13. *The prefix-reachability problem of PNSs is computationally equivalent to the reachability problem of Petri nets.*

Proof. The prefix-reachability problem is equivalent to the reachability problem in the particular case of Petri nets. Thus the prefix-reachability problem of PNSs is as difficult as the reachability problem of Petri nets. Moreover checking the reachability of a given marking in a PNS can be checked by means of the linear simulation of a PNS by a Petri net. Thus we need simply to show that a multiset μ of places is prefix-reachable in $\mathcal{S}$ if and only if $\pi_{\mathrm{cut}}(\mu)$ is reachable in $\mathcal{S}'$.

Let $\mu \in \mathbb{N}^P$ be a multiset of places. Assume first that μ is prefix-reachable in $\mathcal{S}$. By Prop. 3.2, there exists some reachable marking μ° in $\mathcal{S}^\circ$ such $\mu = \pi_{\mathrm{pre}}^{-1}(\mu^\circ|P_{\mathrm{pre}}) + \pi_{\mathrm{cut}}^{-1}(\mu^\circ|P_{\mathrm{cut}})$. Due to the self-loop labeled arcs from $\longrightarrow_{\mathrm{cut}}$ in $\mathcal{S}^\circ$, we can assume that $\mu^\circ|P_{\mathrm{pre}} = 0$ hence $\pi_{\mathrm{cut}}(\mu) = \mu^\circ|P_{\mathrm{cut}}$. Then $\pi_{\mathrm{cut}}(\mu)$ is reachable in $\mathcal{S}'$ because the additional self-loop labeled arcs of $\mathcal{S}'$ enable us to remove all tokens in all places from P_{suf}.

Conversely, assume that $\pi_{\mathrm{cut}}(\mu)$ is reachable in $\mathcal{S}'$. Then there exists some reachable marking μ° in $\mathcal{S}^\circ$ such that $\pi_{\mathrm{cut}}(\mu) = \mu'|P_{\mathrm{cut}}$ and $\mu^\circ|P_{\mathrm{pre}} = 0$. It follows from Lemma 3.10 and Prop. 3.6 that μ is prefix-reachable in $\mathcal{S}$. ∎

4 Checking MSO properties of processes

In this section we show how to check effectively whether all processes of a given bounded Petri net with states $\mathcal{S}$ satisfy a formula ψ expressed in monadic second-order (MSO) logic. To the best of our knowledge, this model-checking problem has not been tackled yet, *even in the particular case of Petri nets*. Our approach is rather simple and relies only on Büchi

Theorem [5] which states the equivalence between the definability of a set of words by an MSO formula and its recognizability by a finite word automaton.

In the rest of this section, we fix a bounded PNS $\mathcal{S}$ with an initial marking μ_{in} over the finite set of places P and the finite set of rules R. In order to simplify the presentation of our result, we consider in this section that the events of a process are labeled by a rule instead of a rule name. The MSO logic we consider applies to the class of partial orders whose nodes are labeled by letters from the disjoint union $\Sigma = P \dot\cup R$, which includes in particular the processes of each rule sequence $s \in R^\star$. Thus the models we consider here are triples $(N, \preccurlyeq, \lambda)$ where N is a finite set of nodes, $\preccurlyeq$ is a partial order over N, and λ is a mapping from N to $\Sigma = P \dot\cup R$.

4.1 MSO logic over words and labeled partial orders

Formulae of the MSO logic that we consider involve first-order variables $x, y, z...$ for nodes and second-order variables $X, Y, Z...$ for sets of nodes. They are built up from the atomic formulae $P_a(x)$ for $a \in \Sigma$ (which stands for "the node x is labeled by the letter a"), $x \preccurlyeq y$, and $x \in X$ by means of the Boolean connectives $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ and quantifiers $\exists, \forall$ (both for first order and for set variables). Formulae without free variables are called *sentences*.

The satisfaction relation $\models$ between a labeled partial order $(N, \preccurlyeq, \lambda)$ and a sentence is defined canonically with the understanding that first order variables range over nodes of N and second order variables over subsets of N. The class of labeled partial orders which satisfy a sentence φ is denoted by $\text{Mod}(\varphi)$. We say that a class of labeled partial orders $\mathcal{L}$ is MSO-*definable* if there exists a sentence φ such that $\mathcal{L} = \text{Mod}(\varphi)$.

4.2 A technique to decide $\mathcal{S} \models \psi$

Since $\mathcal{S}$ is bounded, we can compute and fix some natural number B such that each reachable marking μ of $\mathcal{S}$ is B-bounded, that is, $\mu(p) \leqslant B$ for each $p \in P$. A rule sequence $s = r_1...r_m \in R^\star$ firable from μ_{in} is said to be B-*bounded* if the marking reached by each subsequence $r_1...r_l$ is B-bounded. In particular any firable computation sequence of $\mathcal{S}$ is B-bounded.

We fix a word $w_{\text{in}} \in P^\star$ that is a linear extension of μ_{in}, i.e. $|w_{\text{in}}|_p = \mu_{\text{in}}(p)$ for all $p \in P$. Similarly, for each rule $r \in R$, we fix a word $w_r = r.w'_r$ where $|w'_r|_p = r^\bullet(p)$ for all $p \in P$. Then for each rule sequence $s = r_1...r_m \in R^\star$, the sequence $w_s = w_{\text{in}}.w_{r_1}...w_{r_m}$ is called the *representative word of* s. As usual, we will regard w_s as a linearly ordered set of nodes labeled by letters from Σ and we will write $w_s = (N, \leqslant, \lambda)$ where N is a set of nodes, $\leqslant$ is a total order over N, and $\lambda : N \rightarrow \Sigma$ is a labeling. Nodes labeled by a place are called *place nodes* whereas nodes labeled by a rule are called *rule nodes*. Interestingly, w_s is a linear extension of any process of s, where the place nodes following a rule node labeled by r correspond to the multiset of tokens $r^\bullet$ produced by this occurrence of r.

In order to recover a process of s from the representative word w_s, we need to specify which available tokens are consumed by each occurrence of rule. To do so, we use a coloring of the place nodes of w_s so that at each step all available tokens in a given place get distinct

w_s	x	y	z	p	z	x	c	y
C_1	✗		✗					✗
C_2		✗			✗	✗		
$D_{x,1}$				✗				
$D_{y,2}$							✗	
$D_{z,2}$							✗	

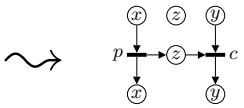

FIG. 16. A process coloring of $w_s = xyzpzxcy$ and the corresponding process

colors. Moreover we also provide rule nodes with a series of other colors in order to specify which tokens are consumed at each step of s.

DEFINITION 4.1. *Let $w = (N, \leqslant, \lambda)$ be a linear order of nodes labeled by Σ. A* process coloring *of w consists of*

- *a* partition $C = \{C_1, ..., C_B\}$ *of the set of place nodes; a place node $n \in N$ is said to be colored by k in place p if $\lambda(n) = p$ and $n \in C_k$.*
- *for each place $p \in P$ and each $k \in [1..B]$, a subset of rule nodes $D_{p,k}$; we say that a rule node $n \in N$ consumes a token colored by k in place p if $n \in D_{p,k}$.*

Moreover the three next conditions must be satisfied:

PC_1: *For each rule node n, for each place $p \in P$: $\#\{k \in [1..B] \mid n \in D_{p,k}\} = ({}^\bullet\lambda(n))(p)$;*

PC_2: *For each place $p \in P$ and each color $k \in [1..B]$, any two place nodes colored by k in place p are separated by some rule node which consumes a token colored by k in place p;*

PC_3: *For each rule node n which consumes a token colored by k in place p, there exists some preceding place node $n' < n$ colored by k in place p such that no rule node between n' and n consumes a token colored by k in place p.*

Intuitively a place node belongs to C_k if it describes a token colored by k in place $\lambda(n) \in P$. A rule node n belongs to $D_{p,k}$ if it describes an occurrence of the rule $\lambda(n) \in R$ which consumes a token colored by k in place p. Thus the condition PC_1 asserts that n consumes the appropriate multiset of tokens in each place, provided that these tokens have distinct colors. Precisely PC_2 guarantees that the colors given to new tokens produced by the occurrence of a rule in a place differ from the colors used by available tokens in this place. It ensures also that the tokens produced in some place by the occurrence of a rule get distinct colors. Consequently, at each step all available tokens in a place have distinct colors. In order to recover a process of s from a process coloring of w_s, we have to make sure that there are enough available tokens when each rule is applied. The last requirement PC_3 guarantees that for each rule node which consumes a token colored by k in place p, some token of this kind occurred before the rule and has not been consumed in between.

We can show that the notion of process coloring characterizes the linear extensions of processes and allows to recover a process from a word. This property is established by the two next statements (Prop. 4.2 and 4.3). Consider for instance the rule sequence $s = pc$ from the initial marking $\mu_{\text{in}} = \{x, y, z\}$ where $p : x \to x + z$ and $c : y + z \to y$. A process coloring of $w_s = xyzpzxcy$ with $B = 2$ is given by the tabular on the left-hand side of Fig. 16. The corresponding process is depicted on the right-hand side of Fig. 16.

PROPOSITION 4.2. *Let $w_s = (N, \leqslant, \lambda)$ be a linear order of nodes labeled by Σ which corresponds to the representative word of a rule sequence $s \in R^\star$. Let $C = (C_k)_{k\in[1..B]}$ and $D = (D_{p,k})_{p\in P,k\in[1..B]}$ be a process coloring of w_s. Let $\prec\!\!\cdot$ be the binary relation over N such that $x \prec\!\!\cdot\, y$ if*

- *either x is a rule node and y is a following place node with no rule node in between*
- *or y is a rule node and x is a preceding place node colored by k in place p such that y consumes a token colored with k in place p and no rule node between x and y consumes a token colored with k in place p.*

Let $\preccurlyeq$ be the reflexive and transitive closure of $\prec\!\!\cdot$. Then the labeled partial order $(N, \preccurlyeq, \lambda)$ is a process of s firable from μ_{in}, denoted by $\mathcal{K}_{C,D}(s)$. Moreover s is B-bounded.

Proof. We proceed by induction over the length of s. The base case where $|s| = 0$ is trivial because $\prec\!\!\cdot$ is empty and $\mu_{\text{in}}(p) \leqslant B$ for each $p \in P$. Induction step. Let $s = r_1...r_m$ be a rule sequence of length $m \geqslant 1$ and $w_s = (N, \leqslant, \lambda)$ be its representative word. Let (C, D) be a process coloring of w_s. We consider the subsequence $s' = r_1...r_{m-1}$ and its representative word $w_{s'} = (N', \leqslant, \lambda)$ with $N' \subsetneq N$. It is easy to check that the restriction of the process coloring (C, D) to the nodes from N' is a process coloring of $w_{s'}$. Let $\prec\!\!\cdot_{s'}$ be the corresponding binary relation over N' and $\preccurlyeq_{s'}$ be the corresponding partial order. By induction hypothesis, $(N', \preccurlyeq_{s'}, \lambda)$ is a process $\mathcal{K}'$ of s' and s' is a rule sequence firable from μ_{in} and B-bounded. Note that the binary relation $\prec\!\!\cdot$ is acyclic. Moreover the restriction of $(N, \preccurlyeq, \lambda)$ to nodes of N' is precisely the process $\mathcal{K}' = (N', \preccurlyeq_{s'}, \lambda)$ of s'. We need to check that adding the nodes from $N \setminus N'$ to $\mathcal{K}'$ according to $\prec\!\!\cdot$ yields a process of s. To do so, we check the three next properties:

1. For each place $p \in P$, the rule node $n^\circ \in N \setminus N'$ corresponding to r_k covers at most ${}^\bullet r_k(p)$ place nodes labeled by p, i.e. $\#\{n \in N' \mid n \prec\!\!\cdot\, n^\circ \wedge \lambda(n) = p\} \leqslant {}^\bullet r_k(p)$. This follows from PC_1 and PC_2.
2. For each place $p \in P$, the rule node $n^\circ \in N \setminus N'$ corresponding to r_k covers at least ${}^\bullet r_k(p)$ place nodes labeled by p, i.e. $\#\{n \in N' \mid n \prec\!\!\cdot\, n^\circ \wedge \lambda(n) = p\} \geqslant {}^\bullet r_k(p)$. This follows from PC_1 and PC_3.
3. The conditions do not branch, i.e. for each place node $n \in N$, if $n \prec\!\!\cdot\, n_1$ and $n \prec\!\!\cdot\, n_2$ then $n_1 = n_2$. This is ensured by the definition of $\prec\!\!\cdot$ itself.

It follows that the labeled partial order $\mathcal{K} = (N, \preccurlyeq, \lambda)$ is a process of s. Thus s is firable from μ_{in}. Finally for each place p and each color k, PC_2 guarantees that at most one place node colored by k in place p is not covered by a rule node. Therefore the marking μ reached by the process $\mathcal{K}$ satisfies $\mu(p) \leqslant B$ for all $p \in P$. Thus s is B-bounded. ∎

Thus each process coloring of w_s yields a process from $[\![s]\!]_{\mu_{\text{in}}}$. Consequently s is firable from μ_{in} as soon as it admits a process coloring. With no surprise s has to be B-bounded, too. Conversely the next result asserts that each process of any rule sequence s firable from μ_{in} can be obtained by some process coloring of w_s, provided that s is B-bounded.

PROPOSITION 4.3. *Let $s = r_1...r_m$ be a B-bounded rule sequence firable from μ_{in} and $\mathcal{K}$ be a process of s. Then there exists a process coloring (C, D) of the representative word w_s such that $\mathcal{K}_{C,D}(s)$ is isomorphic to $\mathcal{K}$.*

Proof. We proceed by induction over the length of s. The base case where $|s| = 0$ is trivial because μ_{in} is B-bounded. Induction step. Let $\mathcal{K} = (B \cup E, F^*, \lambda)$ be a process of $s = r_1 ... r_m$ of length $m \geqslant 1$ with set of conditions C and set of events E. We consider the subsequence $s' = r_1 ... r_{m-1}$ and the prefix $\mathcal{K}' = (B' \cup E', F^*, \lambda)$ of $\mathcal{K}$ in which the event $e^\circ \in E$ corresponding to the last occurrence of r_m and the subsequent conditions are removed. Clearly $\mathcal{K}'$ is a process of s'. By induction hypothesis, there exists a process coloring (C', D') of $w_{s'}$ such that $\mathcal{K}_{C',D'}(s') = (N', \preccurlyeq_{C',D'}, \lambda)$ is isomorphic to $\mathcal{K}' = (B' \cup E', F, \lambda)$. We let $\sigma : B' \cup E' \to N'$ denote an isomorphism from $\mathcal{K}'$ to $\mathcal{K}_{C',D'}(s')$, i.e. a bijection such that

- $x_1 F^* x_2$ iff $\sigma(x_1) \preccurlyeq_{C',D'} \sigma(x_2)$ for all $x_1, x_2 \in B' \cup E'$ and
- $\lambda(\sigma(x)) = \lambda(x)$ for all $x \in B' \cup E'$.

We extend the bijection $\sigma : B' \cup E' \to N'$ to a bijection $\sigma : B \cup E \to N$ such that

- $\sigma(e^\circ)$ is the rule node from $N \setminus N'$, and
- each condition c from $B \setminus B'$ maps to a place node $\sigma(c) \in N \setminus N'$ such that $\lambda(\sigma(c)) = \lambda(c)$.

We extend also the process coloring (C', D') to a process coloring (C, D) of w_s in two steps:

1. For all places $p \in P$ and for all colors $k \in [1..B]$, the rule node $\sigma(e^\circ)$ belongs to $D_{p,k}$ if the event e° covers a condition c such that the corresponding node $\sigma(c)$ is labeled by p and colored by k, i.e. $\lambda(c) = p$ and $\sigma(c) \in C'_k$.
2. The colors of the additionnal place nodes from $N \setminus N'$ are chosen arbitrarily such that all maximal conditions of $\mathcal{K}$ labeled by the same place p have distinct colors. This is possible because the marking reached by s is B-bounded.

We can check that the resulting coloring (C, D) is a process coloring.

PC_1: Let $p \in P$ and $n^\circ = \sigma(e^\circ)$. Since $\mathcal{K}$ is a process, $\#\{k \in [1..B] \mid n^\circ \in D_{p,k}\} \leqslant ({}^\bullet\lambda(n^\circ))(p)$. We can check that all conditions labeled by p and covered by the event e° have distinct colors, because of PC_2. Thus $\#\{k \in [1..B] \mid n^\circ \in D_{p,k}\} = ({}^\bullet\lambda(n^\circ))(p)$.

PC_2: The required property holds for any two place nodes from $w_{s'}$ because (C', D') is a process coloring. It holds also by construction for any two place nodes from $N \setminus N'$. It holds also obviously if one place node belongs to N' and the other to $N \setminus N'$ because the rule node n° occurs in between.

PC_3: The required property holds for each rule node $n \in N'$. Let $p \in P$ and $k \in [1..B]$ be such that $n^\circ \in D_{p,k}$. By definition of $D_{p,k}$ the event e° covers a condition c such that the corresponding node $n = \sigma(c)$ is labeled by p and colored by k. Moreover there is no rule node n' between n and n° with $n' \in D_{p,k}$. Otherwise the condition c would be also covered by some event in $\mathcal{K}'$, hence c would be a branching condition in $\mathcal{K}$.

Recall that σ maps each condition from $B \setminus B'$ to a place node from $N \setminus N'$ with the same label. All these conditions cover e° and all these place nodes cover $\sigma(e^\circ)$. To conclude, we need to check that for each place node $n \in N'$, we have $n \prec_{C,D} n^\circ$ in $\mathcal{K}_{C,D}(s)$ if and only if $\sigma^{-1}(n) F e^\circ$ in $\mathcal{K}$.

Assume first that $n \prec\cdot_{C,D} n^\circ$. Then there exists a place $p \in P$ and a color $k \in [1..B]$ such that $\lambda(n) = p$, $n \in C_k$, and $n^\circ \in D_{p,k}$. Further there exists some condition c in the process $\mathcal{K}$ such that cFe°, $\lambda(c) = p$ and $\sigma(c) \in C_k$. Then $\sigma(c) = n$ because of PC_2. Hence $\sigma^{-1}(n)Fe^\circ$ in $\mathcal{K}$.

Conversely, assume now that $\sigma^{-1}(n)Fe^\circ$ in $\mathcal{K}$. Let $p = \lambda(n)$ and k be the color such that $n \in C_k$. We have $n^\circ \in D_{p,k}$ according to the definition of $D_{p,k}$. No event in $\mathcal{K}'$ covers $\sigma^{-1}(n)$ so there exists no rule node in $w_{s'}$ after n which is colored by $D_{p,k}$. It follows that $n \prec\cdot_{C,D} n^\circ$. ∎

Thus the notion of process coloring characterizes the processes of any B-bounded rule sequence firable from μ_{in}.

Following the easy part of Büchi Theorem, we can design an MSO formula $\phi_{\mathcal{S}}$ which defines the words $w = (N, \leqslant, \lambda)$ over Σ which are representative words of a computation sequence of $\mathcal{S}$. We can also design a formula $\phi_{pc}(C, D)$ with $B \times (|P| + 1)$ second-order free variables $C = (C_k)_{k\in[1..B]}$ and $D = (D_{k,p})_{k\in[1..B],p\in P}$ which characterizes the notion of a process coloring for a word $w = (N, \leqslant, \lambda)$ over Σ. Moreover, by means of Prop. 4.2, we can build a formula $\phi_{\preccurlyeq}(x, y, C, D)$ with two first-order free variables x and y and $B \times (|P|+1)$ second-order free variables such that for any interpretation of $C = (C_k)_{k\in[1..B]}$ and $D = (D_{k,p})_{k\in[1..B],p\in P}$ and any interpretation of x and y, $\phi_{\preccurlyeq}(x, y, C, D)$ is satisfied if and only if we have $x \preccurlyeq y$ in the process corresponding to the process coloring given by the interpretation.

Let ψ be an MSO sentence for labeled partial orders over Σ. We consider the following formula $\overline{\psi}_{\mathcal{S}}$ for words over Σ:

$$\overline{\psi}_{\mathcal{S}} = \phi_{\mathcal{S}} \wedge \exists C, \exists D, (\phi_{pc}(C, D) \wedge \neg\psi'(C, D))$$

where the formula $\psi'(C, D)$ is obtained from ψ by replacing each occurrence of $x \preccurlyeq y$ by $\phi_{\preccurlyeq}(x, y, C, D)$. Thus a word satisfies $\overline{\psi}_{\mathcal{S}}$ if (and only if) it is a representative word of a computation sequence s of $\mathcal{S}$ for which there exists a process coloring which describes a process satisfying $\neg\psi$. In this way we get the main result of this section.

THEOREM 4.4. *Let $\mathcal{S}$ be a bounded PNS and ψ be an MSO sentence over causal nets. All processes of $\mathcal{S}$ satisfy ψ if and only if the word sentence $\overline{\psi}_{\mathcal{S}}$ is* not *satisfiable.*

Thus the model-checking problem for a bounded PNS against an MSO-sentence is decidable. In practice, the unsatisfiability of $\overline{\psi}_{\mathcal{S}}$ is reduced to the emptiness problem of a finite automaton. It is of course more efficient not to include the sentence $\phi_{\mathcal{S}}$ and to compare the resulting automaton with the automaton that recognizes the representative words of computation sequences of $\mathcal{S}$. Noteworthy we could provide the PNS $\mathcal{S}$ with a subset of accepting states and check alternatively that all processes derived from an accepting computation sequence satisfy ψ.

4.3 Comparisons to related works

This result subsumes previous works in several extents. A cMSG is said to be safe if all paths from the initial state to some fixed state lead to the same marking. Consequently,

any safe PNS is bounded. This constraint is interesting because checking whether a PNS is bounded and computing a bound is hard and requires exponential space [11]. Yet the technique presented in this section allows us to check effectively MSO properties of the processes derived from a bounded (non-safe) cMSG or more generally a bounded PNS. Since MSO behaviours can be characterized by some MSO sentence, this result extends the main result from [26] which asserts that the model-checking problem for *safe* cMSGs is decidable (see also [13, Cor 6.1]). As opposed to [13, 26], we do not assume FIFO behaviours and consequently we cannot make use of the notion of representative linearizations. The fact is that, as already mentionned, a computation sequence can correspond to several non-isomorphic processes depending on the order identical particles are consumed. Therefore we need the notion of process coloring (Def. 4.1) to recover a process from a word. This is the main difference with the setting of MSCs because these are completely specified by any of their linearizations. Still, the FIFO restriction can be formalized in MSO logic and our technique applies also in this special case. Second Petri nets and VASSs abstract away from the notions of sites and channels in the setting of MSCs: A place can describe the local state of a site, a communication channel, a shared-variable, etc. In particular our approach applies to any bounded Petri net. To the best of our knowledge, the model checking problem of bounded Petri nets against MSO formulae under the process semantics has not been investigated so far in the literature. We show with an example below that the process semantics of Petri nets can be used to model and check systems with specific behavioural constraints, such as FIFO channels, causal communication, or private keys, as soon as these restrictions can be formalized by an MSO sentence. Note finally that it may be possible to encode a bounded Petri net with a safe cMSGs. However this requires to represents each reachable marking by a distinct control state so the size of the resulting cMSG would be exponential in the size of the Petri net even for 1-safe Petri nets.

5 Conclusion

We introduce a natural partial order semantics for vector addition systems with states (and Petri nets with states) which extends the non-branching process semantics of Petri nets and follows the asynchronous approach of message sequence charts. We show how basic problems about the set of markings reached along concurrent executions, such as boundedness, covering and reachability, can be solved similarly to the analogous problems for Petri nets. We show also how to check effectively any MSO property of these partial orders provided that the system is bounded. This result generalizes results known for message sequence graphs and is new, even in the restricted case of Petri nets.

However vector addition systems with states are not always as easy as Petri nets to handle. We have observed that vector addition systems with states are more expressive than (pure) Petri nets under this process semantics. Moreover the synthesis problem of Petri nets from prefix-bounded vector addition systems with states turns out to be undecidable. As a consequence we have illustrated the gap between Petri nets and vector addition systems with states as follows: It is undecidable whether two prefix-bounded vector addition systems with states are semantically equivalent, contrary to Petri nets.

Introduced as a formal model for the semantics of elementary Petri nets, Mazurkiewicz traces [27] are particular labeled partial orders that benefit from a rich and still growing theory [7]. In particular we have made use of the undecidability of the universality problem for rational trace languages to establish all undecidability results presented in this paper. We have shown that Mazurkiewicz traces can be described as the processes of very particular prefix-bounded systems. However Mazurkiewicz trace theory enjoys several nice positive results for particular rational languages. Most of these results have been already adapted to the setting of message sequence charts (see in particular [19]). Thus, this study leads us to investigate an extension of Mazurkiewicz trace theory to the whole setting of prefix-bounded Petri nets with states. Moreover some classes of Petri nets with states for which realizability and prefix-realizability become decidable might arise from this study.

References

1. R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Transactions on Software Engineering*, 29(7):623–633, 2003.
2. R. Alur and M. Yannakakis. Model checking of message sequence charts. In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 1999.
3. H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In Ed Brinksma, editor, *TACAS*, volume 1217 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 1997.
4. E. Best and R. R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.
5. J.R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.
6. Ph. Darondeau. Unbounded Petri Net Synthesis. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 413–438. Springer, 2003.
7. V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific Publ. Co., 1995.
8. M. Droste, P. Gastin, and D. Kuske. Asynchronous cellular automata for pomsets. *Theoretical Computer Science*, 247(1-2):1–38, 2000.
9. A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. Part II: State spaces of concurrent systems. *Acta Informatica*, 27(4):343–368, 1989.
10. J. Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28:575–591, 1991.
11. J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *Bulletin of the EATCS*, 52:244–262, 1994.
12. J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20:285–310, 2002.
13. B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006.
14. H.J. Genrich and E. Stankiewicz-Wiechno. A dictionary of some basic notions of net theory. In Wilfried Brauer, editor, *Advanced Course: Net Theory and Applications*, volume 84 of *Lecture Notes in Computer Science*, pages 519–531. Springer, 1975.
15. R. J. van Glabbeek, U. Goltz, and J.-W. Schicke. On causal semantics of Petri nets. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2011.
16. U. Goltz and W. Reisig. The non-sequential behavior of Petri nets. *Information and Control*, 57(2/3):125–147, 1983.
17. J. Grabowski. On partial languages. *Fundamenta Informaticae*, 4(2):427–498, 1981.
18. E. L. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. In Tiziana Margaria and Wang Yi, editors, *TACAS*, volume 2031 of *Lecture Notes in Computer Science*, pages 496–511. Springer, 2001.

19. J. G. Henriksen, M. Mukund, K. Narayan Kumar, M. A. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.
20. P. W. Hoogers, H. C. M. Kleijn, and P. S. Thiagarajan. A trace semantics for Petri nets. *Information and Computation*, 117(1):98–114, 1995.
21. J.E. Hopcroft and J-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979.
22. R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.
23. A. Kiehn. On the interrelation between synchronized and non-synchronized behaviour of Petri nets. *Journal of Information Processing and Cybernetics / Elektronische Informationsverarbeitung und Kybernetik*, 24(1/2):3–18, 1988.
24. S. Leue, R. Mayr, and W. Wei. A scalable incomplete test for the boundedness of UML RT models. In Kurt Jensen and Andreas Podelski, editors, *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2004.
25. P. Madhusudan. Reasoning about sequential and branching behaviours of message sequence graphs. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 809–820. Springer, 2001.
26. P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FSTTCS*, volume 2245 of *LNCS*, pages 256–267. Springer, 2001.
27. A. Mazurkiewicz. Concurrent program schemes and their interpretation. Technical report, DAIMI Report PB-78, Aarhus University, 1977.
28. M. Mukund. Petri nets and step transition systems. *International Journal of Foundations of Computer Science*, 3(4):443–478, 1992.
29. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.
30. J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1981.
31. C. A. Petri. *Non-Sequential Processes: Translation of a Lecture given at the IMMD Jubilee Colloquium on 'Parallelism in Computer Science', Universität Erlangen-Nürnberg.* St. Augustin: Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-77–5, 1977.
32. C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.
33. W. Reisig. *Petri Nets, An Introduction.* EATCS Monographs, vol. 4. Springer, 1985.
34. C. Reutenauer. *The Mathematics of Petri Nets.* New York, USA: Prentice-Hall, 1990.
35. J. Sakarovitch. The "last" decision problem for rational trace languages. In Imre Simon, editor, *LATIN*, volume 583 of *Lecture Notes in Computer Science*, pages 460–473. Springer, 1992.
36. J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.
37. W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*, volume 625 of *Lecture Notes in Computer Science*. Springer, 1992.
38. W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. - Informatique Théorique et Applications*, 21:99–135, 1987.