Vector Addition Systems with States vs. Petri nets

Florent Avellaneda, Rémi Morin

Vector Addition Systems with States vs. Petri nets

Florent Avellaneda & Rémi Morin

Aix-Marseille Université — CNRS, UMR 7279
Laboratoire d'Informatique Fondamentale de Marseille

Abstract. Vector addition systems with states are known to be computationally equivalent to Petri nets. In this paper we compare these two models with respect to the non-branching process semantics based on labeled occurrence nets. We investigate first some basic problems such as reachability, boundedness and covering, when one considers the markings reached by the *prefixes* of the processes accepted by some vector addition system with states. By means of a new simulation by a Petri net, we prove that these problems are computationally equivalent to the analogous problems for Petri nets. Second, based on some undecidability result from Mazurkiewicz trace theory, we show that one cannot check effectively whether a vector addition system with states is semantically equivalent to some Petri net, even for systems whose prefix-reachable markings are bounded.

April 10, 2012

Introduction

Introduced by Karp and Miller in [21] the notion of a vector addition system is equivalent to pure Petri nets: The pre-set and the post-set of each transition are disjoint. A vector addition system with states (a VASS) is simply a vector addition system provided with some control state which determines whether a transition may be fired or not [20]. Moreover the firing of a transition leads to a new control state. It is well-known that all these models are computationally equivalent, for they can somehow simulate each other [27, 30, 31]. In particular any vector addition system with states over n places can be simulated by some vector addition system over $n+3$ places [20]. The addition of control states to vector addition systems makes it easier to model distributed or parallel systems of processes. For instance high-level message sequence charts can be regarded as particular cases of vector addition systems with states. Further, it was proved convenient to use a vector of control states to check the structural properties of channels within a network of communicating finite state machines [23].

Suggested by Petri in the restricted setting of condition/event systems [28] , the process semantics of a Petri net defines labeled occurrence nets as partially ordered sets of events with non-branching conditions [5, 11, 16, 30, 34]. As opposed to the other classical partial-order semantics based on step firing sequences [17, 22, 34], a process records all causal dependencies between the events occurring along a run. We present in Section 1 a partial order semantics for vector addition systems with states which extends the usual process semantics of Petri nets. The approach is simple and natural. We consider first the set of firable computation sequences of a VASS and second we define the processes that represent a given sequence. Each process describes some causal dependencies between events which are no longer totally ordered. This means that two transitions 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 in concurrency theory. In particular this is similar to the way message sequence charts are derived from high-level specifications (see, e.g. [2–4, 18]). Thus, control states are simply a mean to specify particular sets of transition sequences and they are not represented in the process semantics. In this way, high-level message sequence charts, rational Mazurkiewicz trace languages [8], or more generally concurrent executions of stably concurrent automata [6] can be encoded in the framework of vector addition systems with states. However, one specific feature of process semantics is that a computation sequence can be described by several non-isomorphic processes depending on the order identical tokens are consumed.

A key verification problem for high-level message sequence charts is to detect channel divergence, i.e. to decide whether the number of pending messages along an execution is unbounded [2–4, 18]. This problem is NP-complete. The analogous problem for vector addition systems with states under the process semantics 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 2 a technique to solve this problem by means of a simulation of the prefixes by a Petri net. Consequently this problem is computationally equivalent to the boundedness problem for Petri nets and requires at least exponential space [12]. Other basic decision problems for the prefix-reachable markings of a VASS are of course interesting. We show in particular that the reachability and the covering of a given marking by some prefix can be solved using the same technique.

Synthesis problems have been investigated for various models of concurrency: asynchronous automata [8, 9, 35], Petri nets [7, 10, 19, 25], communicating finite-state machines [2, 18], 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 3 a natural synthesis problem for Petri nets: Given some vector addition system with states we ask whether its processes are generated by some Petri net. We show that this problem is undecidable (even for prefix-bounded systems) by means of a reduction to the universality problem for rational Mazurkiewicz trace languages [33]. As a consequence, it is undecidable whether two vector addition systems with states generate the same processes, as opposed to Petri nets. Thus, despite of the positive results from Section 2, vector addition systems with states turn out to be more complicated than Petri nets when one considers their process semantics, even under the restriction to prefix-bounded systems.

In order to simplify the presentation, we introduce in this paper the notion of a *Petri net with states* as an equivalent to vector addition systems with states without their restriction to pure transitions. This framework appears as the minimal joined formal generalization of Petri nets and vector addition systems with states. Thus Petri nets are regarded as Petri nets with states provided with a single state whereas vector addition systems with states are simply Petri nets with states using *pure* transitions, only. 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 $\lambda(\mu) = \sum_{a \in A} \mu(a) \cdot \lambda(a)$

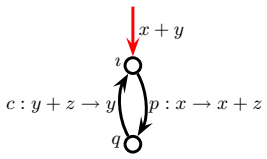

FIG. 1. A PNS with two states

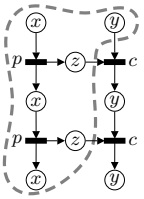

FIG. 2. A labeled causal net and a prefix

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.

1 Model and semantics

In this section we introduce the model of Petri nets with states as a simple generalisation of Petri nets. This framework corresponds to vector addition systems with states without its restriction to pure transitions. We extend also in a natural way the non-branching process semantics of Petri nets to the more general setting of Petri nets with states.

1.1 Petri nets 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 molecule type in biological systems, 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*.

We consider a (transition) rule as a mean to produce some new tokens in 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 will be denoted by $\lambda : \alpha \to \beta$. It means intuitively that the multiset of places α can be consumed to produce the multiset of places β in an atomic way. Different rules may 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 $r^\bullet = \beta$. We let R denote the (infinite) set of all rules.

DEFINITION 1.1. *A* Petri net with states *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 (for short: a PNS). A labeled arc $(q_1, r, q_2) \in \longrightarrow$ will be denoted by $q_1 \xrightarrow{r} q_2$. A rule sequence $u = r_1 ... r_n \in R^\star$ is called a *computation sequence of* $\mathcal{S}$ if there are some 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 $\imath \xrightarrow{u} q_n$. 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 Figure 1. We denote by 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 Petri nets with states simply as a formal mean to specify regular sets of rules.

A rule sequence $u = r_1...r_n \in R^\star$ is *firable* from some marking μ if there are some multisets of places μ_0, ..., μ_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 u can be applied from the marking μ in the total order specified by u: Each rule r_k consumes ${}^\bullet r_k$ tokens from μ_{k-1} and produces $r_k^\bullet$ new tokens which yield the subsequent multiset μ_k. Then we say that μ_n is reached by the rule sequence u from the marking μ. We also say that u leads to μ_n. We denote by FCS($\mathcal{S}$) the set of all *firable* computation sequences of $\mathcal{S}$.

Originally introduced in [20], the notion of a vector addition system with states can be formally defined in several slightly different ways. In this paper, a *vector addition system with states* (for short: a VASS) is simply a Petri net with states 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 identical vectors would be forbidden. This restriction would have no effect on the results presented in this paper. We explain below why we can identify the well-known formalism of Petri nets as particular Petri nets with states provided with a single state.

1.2 Initialized Place/Transition Petri nets and causal nets

Definition 1.2. *A* Petri net *is a quadruple* $\mathcal{N} = (P, T, W, \mu_{\text{in}})$ *where*
- P *is a set of places and* T *is a 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* weight function*;*
- μ_{in} *is a map from* P *to* $\mathbb{N}$*, called* initial marking*.*

We shall depict Petri nets in the usual way as in Figure 4: Black rectangles represent transitions whereas circles represent places; moreover tokens in places describe the initial marking. Given a Petri net $\mathcal{N} = (P, T, W, \mu_{\text{in}})$ 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 $\mathcal{N} = (P, T, W, \mu_{\text{in}})$ 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 Petri nets with states provided with a single state such that each transition carries a rule with a distinct rule name. Take any PNS $\mathcal{S}$ with a single state $\imath$ such that each transition

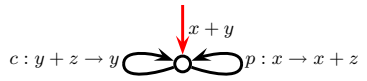

FIG. 3. A PNS with a single state

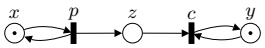

FIG. 4. and the corresponding Petri net

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 Figure 3 corresponds to the Petri net from Figure 4.

In case 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.

DEFINITION 1.3. *[11, 34] 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 initial marking can be recovered from the structure (B, E, F). For that reason causal nets are often defined as a triple (B, E, F) satisfying the two first conditions of Definition 1.3. In the literature causal nets are also called *occurrence nets*, see e.g. [5, 14, 16, 15, 30]. However more general Petri nets are called occurrence nets in the theory of partial unfolding or branching processes [11, 13, 26].

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 Figure 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 Process semantics of a Petri net with states

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 [5, 11, 15, 16, 30, 34]. 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 Figure 5. Given some initial multiset of places,

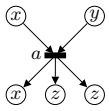

FIG. 5. Rule $a : x + y \to x + 2z$

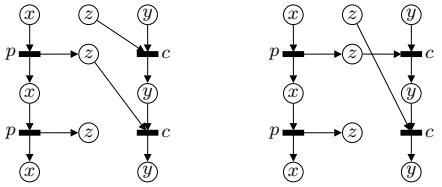

FIG. 6. Two processes from $x + y + z$

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 Figure 2 depicts a process of the Petri net $\mathcal{N}$ from Figure 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 processes of a Petri net (with states) are simply the processes of its firable computation sequences, a notion that we introduce now.

DEFINITION 1.4. *A* process *of a rule sequence $u = r_1...r_n \in R^\star$ 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 $\pi(\mu_{\min}) = \mu$;*
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 $[\![u]\!]_\mu$ the class of all processes of u 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 u is 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 Proposition 2.3. 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.

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

Roughly speaking, any causal net "isomorphic" to a process of u is also a process if u. 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

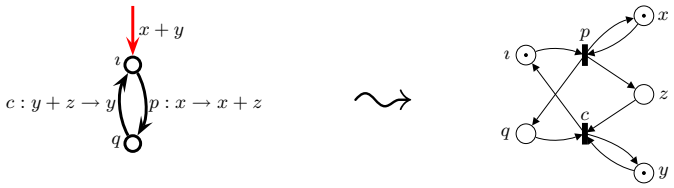

FIG. 7. Simulation of a PNS by a Petri net

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 Figure 1 corresponds to the causal net from Figure 2. However if there are $x + y + z$ tokens initially, then this computation sequence corresponds to the two labeled causal nets from Figure 6 among some others.

DEFINITION 1.5. *Let $\mathcal{S}$ be a Petri net with states with initial marking μ_{in}. A process of $\mathcal{S}$ is a process of some computation sequence of $\mathcal{S}$ from μ_{in}. We let $[\![\mathcal{S}]\!]$ denote the class of all processes of $\mathcal{S}$.*

Thus $[\![\mathcal{S}]\!] = \bigcup_{s \in \text{CS}(\mathcal{S})} [\![s]\!]_{\mu_{\text{in}}}$. It is easy to check that the processes of a PNS provided with a single state are precisely the finite processes of the corresponding finite Petri net w.r.t. the usual process semantics [5, 15, 34]. Moreover any prefix of a process of $\mathcal{S}$ is a process of some rule sequence. Consequently the set of processes of some 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 1.6. 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.

1.4 Simulation of a VASS by a Petri net

Let us now recall how a k-dimensional vector addition system with states 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 [31]. The usual construction is illustrated by Figure 7 which shows on the left-hand side a PNS with 2 states (ι 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. 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 $\mathcal{S}$. 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. In particular, it enables us to prove the next result.

PROPOSITION 1.7. *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 some* 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 some firable computation sequence of $\mathcal{S}$ if and only if some corresponding transition t in $\mathcal{N}$ occurs in some firable transition sequence in $\mathcal{N}$. This is equivalent to check whether the marking of ${}^\bullet t$ is *covered* by some reachable marking of $\mathcal{N}$. This question is known to be decidable [29]. ∎

However this representation is *not* faithful from the partial order point-of-view we have adopted. Continuing the above example, the processes of $\mathcal{S}$ (with three places) differ from the processes of $\mathcal{N}$ (with five places). In this paper, we investigate basic decision problems for the class of processes of a given PNS.

2 Analysis of prefix-reachable markings

The sequential simulation of a PNS by a Petri net described in Figure 7 enables us to analyse the set of markings reached by a PNS by means of well-known results from Petri net theory (see [12] for a survey). However this simulation is useless if we consider the markings reached by *prefixes* of processes.

DEFINITION 2.1. *A marking μ is* prefix-reachable *in a Petri net with states $\mathcal{S}$ if there exists a prefix of a process of $\mathcal{S}$ which leads to the marking μ. A Petri net with states $\mathcal{S}$ is called* prefix-bounded *if the set of prefix-reachable markings is finite.*

The markings reached after a prefix of a process can be very different from the markings reached by complete processes. Continuing Example 1.6, each process of the PNS from Figure 1 leads to a marking with at most 3 tokens whereas prefixes of these processes lead to infinitely many distinct markings (see Fig. 2 for a prefix of a process which leads to a marking with 4 tokens).

In this section we investigate some natural decision problems about the set of prefix-reachable markings of a given PNS $\mathcal{S}$. In particular the prefix-boundedness problem asks whether the set of prefix-reachable markings of $\mathcal{S}$ is finite. We have already noticed that the class of processes of a Petri net is prefix-closed. In this case, a marking is prefix-reachable if and only if it is reachable by some computation sequence. Thus prefix-boundedness is more difficult than the boundedness problem of Petri nets. We prove in this section that the prefix-boundedness of Petri nets with states is actually as difficult as the boundedness of Petri nets. Moreover our techniques apply to similar basic problems such as covering and reachability.

2.1 From Petri nets with states to Petri nets

Let $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{\text{in}})$ be a fixed PNS. We build some Petri net $\mathcal{N}$ that allows us to analyse the set of prefix-reachable markings of $\mathcal{S}$. The construction of $\mathcal{N}$ from $\mathcal{S}$ is illustrated by

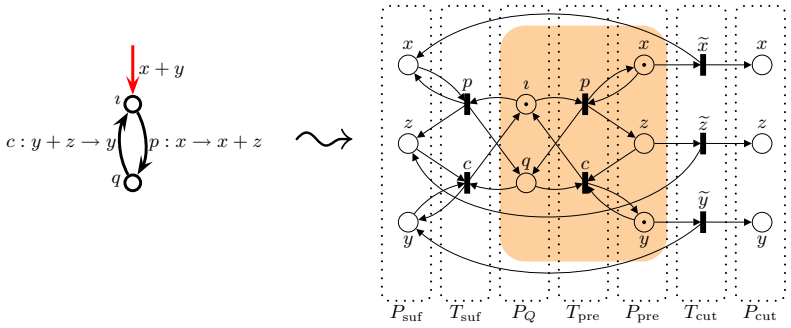

FIG. 8. Analysis of prefix-reachable markings by a Petri net

Figure 8. The Petri net $\mathcal{N}$ consists of four disjoint sets of places: $P_{\text{pre}}, P_{\text{suf}}, P_{\text{cut}}, P_Q$ and three disjoint sets of transitions: $T_{\text{pre}}, T_{\text{suf}}, T_{\text{cut}}$. The subnet of $\mathcal{N}$ restricted to the transitions from T_{pre} and to the places from $P_Q \cup P_{\text{pre}}$ yields a Petri net $\mathcal{N}_{\text{pre}}$ isomorphic to the representation of a PNS illustrated in Fig. 7. In particular, each place from P_{pre} corresponds to some place of $\mathcal{S}$ whereas P_Q describes the set of states Q. Similarly the restriction of $\mathcal{N}$ to the transitions from T_{suf} and to the places from $P_{\text{suf}} \cup P_Q$ yields another Petri net $\mathcal{N}_{\text{suf}}$ isomorphic to $\mathcal{N}_{\text{pre}}$, but with some empty initial marking of P_{suf}. Note that these two sub-nets are synchronized because they share the state places P_Q. Intuitively, for any process $\mathcal{K}$ of $\mathcal{S}$ and for any prefix $\mathcal{K}'$ of $\mathcal{K}$, the Petri net $\mathcal{N}$ can simulate a computation sequence corresponding to $\mathcal{K}$ in such a way that all events from the prefix $\mathcal{K}'$ occur in $\mathcal{N}_{\text{pre}}$ and all events from the suffix $\mathcal{K} \setminus \mathcal{K}'$ occur in $\mathcal{N}_{\text{suf}}$. Moreover some additional places P_{cut} keep track of the places transferred from $\mathcal{N}_{\text{pre}}$ to $\mathcal{N}_{\text{suf}}$ in order to represent the actual marking reached by the prefix $\mathcal{K}'$. To do so, the transitions from T_{cut} allow to consume a token from P_{pre} and to produce a token representing the same place in P_{suf} plus some copy of that transfer in P_{cut}. The key property of this representation, stated in Proposition 2.2 below, asserts that, conversely, any firable computation sequence of $\mathcal{N}$ corresponds to a process $\mathcal{K}$ of $\mathcal{S}$ and a prefix $\mathcal{K}'$ of $\mathcal{K}$ such that the marking of $P_{\text{pre}} \cup P_{\text{cut}}$ describes the marking reached by $\mathcal{K}'$.

Formally each place of $\mathcal{S}$ is represented by three places in P_{pre}, P_{suf} and P_{cut} respectively. These three places are related by some transition in T_{cut} which consumes a token in the place from P_{pre} and produces one token in both places from P_{suf} and P_{cut}. Further each state $q \in Q$ is represented by a place in P_Q. Only one of the places from P_Q contains a token in any reachable marking of $\mathcal{N}$: The place that corresponds to the current state of $\mathcal{S}$. Each arc $q \xrightarrow{r} q'$ in $\mathcal{S}$ gives rise to two transitions in $\mathcal{N}$: The first one, in T_{pre}, is related to places in P_{pre} and P_Q; the second one, in T_{suf}, is related to places in P_{suf} and P_Q. More precisely, these two transitions consume

- the multiset of tokens corresponding to ${}^\bullet r$ in P_{pre} for the first transition, and in P_{suf} for the second one,
- a token in the place from P_Q corresponding to q.

Further these two transitions produce

- the multiset of tokens corresponding to $r^\bullet$ in P_{pre} for the first transition, and in P_{suf} for the second one

– a token in the place from P_Q corresponding to q'.

In the sequel of this section we shall use the following notations:

– 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 the bijections that maps 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.
– We put $\pi_Q : Q \to P_Q$ for the bijection which maps each state $q \in Q$ from $\mathcal{S}$ to the corresponding place in P_Q.

The initial marking μ°_{in} of $\mathcal{N}$ is twofold: First each token in $\mu_{\text{in}} \in \mathbb{N}^P$ gives rise to some token in the corresponding place from P_{pre}. Second a token is added in the place from P_Q that represents the initial state $\imath$ of $\mathcal{S}$. Thus $\mu^\circ_{\text{in}} = \pi_{\text{pre}}(\mu_{\text{in}}) + \{\pi_Q(\imath)\}$.

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 transition in T_{pre} and T_{suf} respectively.
– Finally we put $\tau_{\text{cut}} : P \to T_{\text{cut}}$ for the bijection between P and T_{cut}.

Then the Petri net $\mathcal{N}$ over the set of places $P_{\mathcal{N}} = P_Q \cup P_{\text{pre}} \cup P_{\text{suf}} \cup P_{\text{cut}}$ and the set of transitions $T_{\mathcal{N}} = T_{\text{pre}} \cup T_{\text{suf}} \cup T_{\text{cut}}$ is defined by the following weight function: For each transition $t \in T_{\text{pre}}$, if $\tau^{-1}_{\text{pre}}(t) = q \xrightarrow{r} q'$ then ${}^\bullet t = \pi_{\text{pre}}({}^\bullet r) \cup \{\pi_Q(q)\}$ and $t^\bullet = \pi_{\text{pre}}(r^\bullet) \cup \{\pi_Q(q')\}$. Similarly for each transition $t \in T_{\text{suf}}$, if $\tau^{-1}_{\text{suf}}(t) = q \xrightarrow{r} q'$ then ${}^\bullet t = \pi_{\text{suf}}({}^\bullet r) \cup \{\pi_Q(q)\}$ and $t^\bullet = \pi_{\text{suf}}(r^\bullet) \cup \{\pi_Q(q')\}$. Finally for each transition $t \in T_{\text{cut}}$, if $\tau^{-1}_{\text{cut}}(t) = p$ then ${}^\bullet t = \{\pi_{\text{pre}}(p)\}$ and $t^\bullet = \{\pi_{\text{suf}}(p), \pi_{\text{cut}}(p)\}$. In 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 mainly on the next statement.

PROPOSITION 2.2. *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 $\mathcal{N}$ such that $\mu = \pi^{-1}_{pre}(\mu'|P_{pre}) + \pi^{-1}_{cut}(\mu'|P_{cut})$.*

2.2 Proof of Proposition 2.2

For each rule sequence $u = r_1 ... r_n \in R^\star$, the cost of u is the vector $\text{cost}(u) \in \mathbb{Z}^P$ such that $\text{cost}(u)(p) = \sum_{i=1}^n {}^\bullet r_i(p) - r_i{}^\bullet(p)$ 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 $\mu - \text{cost}(u)$. Let $\mathcal{K}$ be a process of a rule sequence u firable from μ. Then $\mu - \text{cost}(u)$ is the marking reached by $\mathcal{K}$ from μ. Moreover the marking $\mu - \text{cost}(u)$ 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 $\mu - \text{cost}(u) = \sum_{p \in \max(\mathcal{K}) \cap P} \pi(p)$.

For each rule r such that ${}^\bullet r \leqslant \mu - \text{cost}(u)$, 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 2.3. *Let $\mu \in \mathbb{N}^P$. The class of processes of a rule sequence $u \in R^\star$ satisfies the two following properties:*

– *If u is empty, i.e. $u = \varepsilon$, then the processes of $[\![\varepsilon]\!]_\mu$ consist 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* ${}^\bullet r > \mu - cost(u)$.
– *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 1.4, 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 2.4. *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 2.5. *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. 2.3, 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 Proposition 2.3, 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 $\mathcal{N}$ after the firing of s.

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 $\mathcal{K}_v \in [\![v]\!]_{\mu_{\text{in}}}$. 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 2.6. *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 $\mathcal{K} \in [\![v.w]\!]_{\mu_{\text{in}}} \cap [\![u]\!]_{\mu_{\text{in}}}$. By Proposition 2.3, $\mathcal{K}$ may be built from a causal net $\mathcal{K}_v \in [\![v]\!]_{\mu_{\text{in}}}$ 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 $\mathcal{K} \in [\![u]\!]_{\mu_{\text{in}}}$ with $u \in \text{CS}(\mathcal{S})$ 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 $\mathcal{K}' \in [\![v]\!]_{\mu_{\text{in}}}$ 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 $\mathcal{K} \in [\![v.w]\!]_{\mu_{\text{in}}}$. Therefore (u, v, w) is a partial computation. ∎

LEMMA 2.7. *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. 2.3). 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 Proposition 2.2 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 Petri net $\mathcal{N}$ 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 represented by a single token in P_Q and to check that all events from $\mathcal{K}_u$ that do not occur in $\mathcal{K}_v$ are performed by transitions from T_{suf}. To do so, we have to guide $\mathcal{N}$ to transfer *exactly* the required number of tokens from P_{pre} to P_{suf}, which corresponds to the marking of P_{cut}.

LEMMA 2.8. *Let (u, v, w) be a partial computation in $\mathcal{S}$ and q be some state such that $\iota \xrightarrow{u} q$ in $\mathcal{S}$. There exists some firable rule sequence s in $\mathcal{N}$ 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) $\pi_Q^{-1}(\mu_s^\circ | P_Q) = \{q\}$.

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 $\mathcal{N}$ satisfies the four above properties because $\mu^\circ_{\text{in}} = \pi_{\text{pre}}(\mu_{\text{in}}) \cup \pi_Q(\{\imath\})$. 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 $\mathcal{N}$ such that the four above properties are satisfied. In particular, $\pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) = \text{req}(w)$. Moreover Proposition 2.5 ensures that $\text{req}(w.a) \leqslant \mu_{v'}$ because $\mathcal{K}_{w'} \in [\![w.a]\!]_{\mu_{v'}}$. Furthermore we have on one hand $\pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) = \text{req}(w) \leqslant \text{req}(w.a)$; and on the other hand $\text{req}(w.a) \leqslant \mu_{v'}$ i.e. $\text{req}(w.a) \leqslant \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}})$. Therefore

$$\pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) \leqslant \text{req}(w.a) \leqslant \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}})$$

   It follows that there is some multiset of places $\hat{\mu} \in \mathbb{N}^P$ such that $\pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) + \hat{\mu} = \text{req}(w.a)$ with $\pi_{\text{pre}}(\hat{\mu}) \leqslant \mu^\circ_s | P_{\text{pre}}$. We consider a sequence of transitions $A \in T^\star_{\text{cut}}$ in $\mathcal{N}$ 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

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

   Hence $\pi^{-1}_{\text{suf}}(\mu^\circ_{s.A} | P_{\text{suf}}) = \pi^{-1}_{\text{suf}}(\mu^\circ_s | P_{\text{suf}}) + \hat{\mu} \geqslant {}^\bullet a$. It follows that $s.A.\tau_{\text{suf}}(a)$ is a firable computation sequence of $\mathcal{N}$. The latter satisfies the required conditions:
   (a) $\pi^{-1}_{\text{pre}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{pre}}) = \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}}) - \hat{\mu}$ and $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{cut}}) = \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) + \hat{\mu}$ hence $\pi^{-1}_{\text{pre}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{pre}}) + \pi^{-1}_{\text{cut}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{cut}}) = \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}}) + \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) = \mu_{v'}$.
   (b) $\pi^{-1}_{\text{pre}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{pre}}) = \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}}) - \hat{\mu}$, $\pi^{-1}_{\text{suf}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{suf}}) = \pi^{-1}_{\text{suf}}(\mu^\circ_s | P_{\text{suf}}) + \hat{\mu} - {}^\bullet a + a^\bullet$ therefore $\pi^{-1}_{\text{pre}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{pre}}) + \pi^{-1}_{\text{suf}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{suf}}) = \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}}) + \pi^{-1}_{\text{suf}}(\mu^\circ_s | P_{\text{suf}}) - {}^\bullet a + a^\bullet = \mu_{u.a} = \mu_{u'}$.
   (c) $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_{\text{cut}}) = \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) + \hat{\mu} = \text{req}(w.a)$.
   (d) $\pi^{-1}_Q(\mu^\circ_{s.A.\tau_{\text{suf}}(a)} | P_Q) = \{q'\}$.
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 $\mathcal{N}$ such that the four above properties are satisfied. Then we have

$$^\bullet a \leqslant \mu_v - \text{req}(w') = \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}}) - \pi^{-1}_{\text{cut}}(\mu^\circ_s | P_{\text{cut}}) = \pi^{-1}_{\text{pre}}(\mu^\circ_s | P_{\text{pre}}).$$

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

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

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

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

(d) $\pi_Q^{-1}(\mu^\circ_{s.\tau_{\mathrm{pre}}(a)}|P_Q) = \{q'\}$.

■

LEMMA 2.9. *Let (u, v, w) be a partial computation and $a \in R$ be a rule such that ${}^\bullet a + \mathrm{req}(w) \leqslant \mu_v$. If $u.a \in CS(\mathcal{S})$ then $(u.a, v.a, w)$ is a partial computation.*

Proof. Let $\mathcal{K}_u \in [\![u]\!]_{\mu_{\mathrm{in}}}$ and let $\mathcal{K}_v \in [\![v]\!]_{\mu_{\mathrm{in}}}$ be a prefix of $\mathcal{K}_u$. Since w is firable from $\mathrm{req}(w)$, there exists some labeled causal net $\mathcal{K}_w \in [\![w]\!]_{\mathrm{req}(w)}$. 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 of $\mathcal{N}$ 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. At each step the token in P_Q describes some state $q \in Q$ reached by u. When $\mathcal{N}$ fires an additional transition t, the corresponding partial computation is either (u, v, w) if $t \in T_{\mathrm{cut}}$; or $(u.r, v, w.r)$ if $t \in T_{\mathrm{suf}}$ and t corresponds to an occurrence of rule r; or $(u.r, v.r, w)$ if $t \in T_{\mathrm{pre}}$ and t corresponds to an occurrence of rule r. 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. $\pi_{\mathrm{cut}}(\mathrm{req}(w)) \leqslant \mu_s^\circ|P_{\mathrm{cut}}$.

LEMMA 2.10. *Let s be a firable rule sequence in $\mathcal{N}$ leading to the marking μ_s°. There exists some partial computation (u, v, w) of $\mathcal{S}$ 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) $\pi_{cut}^{-1}(\mu_s^\circ|P_{cut}) \geqslant req(w)$, *and*

(d) $\pi_Q^{-1}(\mu_s^\circ|P_Q)$ *consists of a single state $q \in Q$, and moreover $\imath \xrightarrow{u} q$.*

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 $\mu_{\mathrm{in}}^\circ = \pi_{\mathrm{pre}}(\mu_{\mathrm{in}}) + \pi_Q(\{\imath\})$. 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. $a \in T_{\text{cut}}$. Then we can check that (u, v, w) satisfies the four requirements for $s.a$.
   (a) $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.a}|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_{s.a}|P_{\text{pre}}) = \pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) = \mu_v$.
   (b) $\pi^{-1}_{\text{suf}}(\mu^\circ_{s.a}|P_{\text{suf}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_{s.a}|P_{\text{pre}}) = \pi^{-1}_{\text{suf}}(\mu^\circ_s|P_{\text{suf}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) = \mu_u$.
   (c) We have $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.a}|P_{\text{cut}}) > \pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) \geqslant \text{req}(w)$.
   (d) Finally $\pi^{-1}_Q(\mu^\circ_{s.a}|P_Q) = \pi^{-1}_Q(\mu^\circ_s|P_Q)$.
2. $a \in T_{\text{suf}}$. Then $\tau^{-1}_{\text{suf}}(a)$ is an arc $q \xrightarrow{r}_{\mathcal{S}} q'$ in $\mathcal{S}$ and $u.r$ is a computation sequence of $\mathcal{S}$. Moreover $\pi^{-1}_{\text{suf}}({}^\bullet a|P_{\text{suf}}) = {}^\bullet r$ and $\pi^{-1}_{\text{suf}}(a^\bullet|P_{\text{suf}}) = r^\bullet$. Furthermore ${}^\bullet a \leqslant \mu^\circ_s$, hence ${}^\bullet r \leqslant \pi^{-1}_{\text{suf}}(\mu^\circ_s|P_{\text{suf}}) \leqslant \mu_u$. By Lemma 2.7 we know that $(u.r, v, w.r)$ is a partial computation of $\mathcal{S}$. Moreover
   (a) $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.a}|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_{s.a}|P_{\text{pre}}) = \pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) = \mu_v$.
   (b) $\pi^{-1}_{\text{suf}}(\mu^\circ_{s.a}|P_{\text{suf}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_{s.a}|P_{\text{pre}}) = \pi^{-1}_{\text{suf}}(\mu^\circ_s|P_{\text{suf}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) + \pi^{-1}_{\text{suf}}(a^\bullet - {}^\bullet a|P_{\text{suf}}) = \mu_u - {}^\bullet r + r^\bullet = \mu_{u.r}$.
   (c) We have $\mu_{s.a}|P_{\text{cut}} = \mu_s|P_{\text{cut}}$. Moreover ${}^\bullet a \leqslant \mu^\circ_s$, hence

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

   Then $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.a}|P_{\text{cut}}) = \pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) \geqslant \max(\text{req}(w), {}^\bullet r + \text{cost}(w)) = \text{req}(w.r)$.
   (d) $\pi^{-1}_Q(\mu^\circ_{s.a}|P_Q)$ consists of a single state q' and $\imath \xrightarrow{u} q \xrightarrow{r} q'$.
3. $a \in T_{\text{pre}}$. Then $\tau^{-1}_{\text{pre}}(a)$ is an arc $q \xrightarrow{r}_{\mathcal{S}} q'$. Moreover $\pi^{-1}_{\text{pre}}({}^\bullet a|P_{\text{pre}}) = {}^\bullet r$ and $\pi^{-1}_{\text{pre}}(a^\bullet|P_{\text{pre}}) = r^\bullet$. We observe first that
   (a) $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.a}|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_{s.a}|P_{\text{pre}}) = \mu_v - \pi^{-1}_{\text{pre}}({}^\bullet a|P_{\text{pre}}) + \pi^{-1}_{\text{pre}}(a^\bullet|P_{\text{pre}}) = \mu_{v.r}$
   (b) $\pi^{-1}_{\text{suf}}(\mu^\circ_{s.a}|P_{\text{suf}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_{s.a}|P_{\text{pre}}) = \mu_u - \pi^{-1}_{\text{pre}}({}^\bullet a|P_{\text{pre}}) + \pi^{-1}_{\text{pre}}(a^\bullet|P_{\text{pre}}) = \mu_{u.r}$
   (c) $\pi^{-1}_{\text{cut}}(\mu^\circ_{s.a}|P_{\text{cut}}) = \pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) \geqslant \text{req}(w)$
   (d) $\pi^{-1}_Q(\mu^\circ_{s.a}|P_Q) = \{q'\}$.
   Since $\mu^\circ_s \geqslant {}^\bullet a$, we have $\pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) \geqslant \pi^{-1}_{\text{pre}}({}^\bullet a|P_{\text{pre}})$. On the other hand, $\pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) \geqslant \text{req}(w)$, hence $\pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) + \pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) \geqslant \pi^{-1}_{\text{pre}}({}^\bullet a|P_{\text{pre}}) + \text{req}(w)$, i.e $\mu_v \geqslant {}^\bullet r + \text{req}(w)$. By Lemma 2.9, $(u.r, v.r, w)$ is a partial computation. ∎

We are now ready to prove Proposition 2.2. Let μ be the marking reached by a prefix $\mathcal{K}'$ of a process $\mathcal{K} \in [\![\mathcal{S}]\!]$. According to Proposition 2.6, there exists some partial computation (u, v, w) such that $\mu_v = \mu$. By Lemma 2.8, there exists some firable rule sequence s in $\mathcal{N}$ such that $\pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) = \mu_v = \mu$. Conversely if $\pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) = \mu$ for some firable rule sequence s in $\mathcal{N}$ then Lemma 2.10 ensures that there exists some partial computation (u, v, w) such that $\pi^{-1}_{\text{cut}}(\mu^\circ_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}(\mu^\circ_s|P_{\text{pre}}) = \mu_v$. Moreover Proposition 2.6 asserts that μ_v is the marking reached by some prefix $\mathcal{K}'$ of some process $\mathcal{K} \in [\![\mathcal{S}]\!]$.

2.3 Analysis of prefix-reachable markings

At present we make use of the properties of the Petri net $\mathcal{N}$, in particular Prop. 2.2, 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 Petri net with states $\mathcal{S}$ is finite. It is easy to prove that $\mathcal{S}$ is prefix-bounded if and only if the Petri net $\mathcal{N}$ is bounded. 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 2.11. *The prefix-boundedness problem of Petri nets with states is computationally equivalent to the boundedness problem of Petri nets.*

Proof. It is clear from Proposition 2.2 that if $\mathcal{N}$ 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 $\mu_v(m) \leqslant M$ and $\mu_u(m) \leqslant M$ for all partial computations (u, v, w) and all places $m \in P$. Then Lemma 2.10 ensures that if a firable rule sequence of $\mathcal{N}$ leads to some marking μ then $\mu(p) \leqslant M + 1$ for each place $p \in P_{\mathcal{N}}$. ∎

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 $\pi_{\text{cut}}(\mu)$ is covered by some reachable marking of $\mathcal{N}$. Thus,

THEOREM 2.12. *The prefix-covering problem for Petri nets with states 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 Petri nets with states is as difficult as the covering problem of Petri nets.

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: $\mu_0 \leqslant \mu$. By Proposition 2.2, there exists some reachable marking μ' in $\mathcal{N}$ such $\mu = \pi_{\text{pre}}^{-1}(\mu'|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu'|P_{\text{cut}})$. Due to the structure of $\mathcal{N}$, we can assume that $\mu'|P_{\text{pre}} = 0$ hence $\pi_{\text{cut}}(\mu) = \mu'|P_{\text{cut}}$ and $\pi_{\text{cut}}(\mu_0) \leqslant \mu'|P_{\text{cut}}$.

Conversely, assume now that $\pi_{\text{cut}}(\mu_0) \leqslant \mu'$ for some reachable marking μ' of $\mathcal{N}$. Then, by Proposition 2.2, $\mu_0 \leqslant \pi_{\text{cut}}^{-1}(\mu'|P_{\text{cut}}) \leqslant \mu$ 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 $\mathcal{N}'$ of $\mathcal{N}$ where each place $p \in P_{\text{suf}} \cup P_Q$ is provided with an additional clearing transition 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 $\pi_{\text{cut}}(\mu)$ is reachable in $\mathcal{N}'$. Thus,

THEOREM 2.13. *The prefix-reachability problem of Petri nets with states 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 Petri nets with states is as difficult as the reachability problem of Petri nets.

Let $\mu \in \mathbb{N}^P$ be a multiset of places. Assume first that μ is prefix-reachable in $\mathcal{S}$. By Proposition 2.2, there exists some reachable marking μ' in $\mathcal{N}$ such $\mu = \pi_{\text{pre}}^{-1}(\mu'|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu'|P_{\text{cut}})$. Due to the structure of $\mathcal{N}$, we can assume that $\mu'|P_{\text{pre}} = 0$ hence $\pi_{\text{cut}}(\mu) = \mu'|P_{\text{cut}}$. Then $\pi_{\text{cut}}(\mu)$ is reachable in $\mathcal{N}'$ because the clearing transitions allow to remove all tokens in all places from P_{suf} and P_Q.

Conversely, assume that $\pi_{\text{cut}}(\mu)$ is reachable in $\mathcal{N}'$. Then there exists some reachable marking μ' in $\mathcal{N}$ such that $\mu'|P_{\text{pre}} = 0$ and $\pi_{\text{cut}}(\mu) = \mu'|P_{\text{cut}}$. It follows from Proposition 2.2, that μ is prefix-reachable in $\mathcal{S}$. ∎

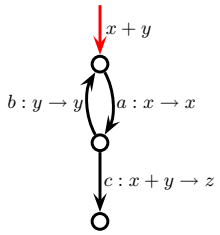

FIG. 9. A non prefix-realizable PNS

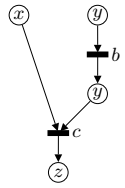

FIG. 10. Some implied process

3 Two synthesis problems for bounded Petri nets

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 mean 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 3.1. *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 1.6) whereas the set of processes 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 3.2. Consider the PNS $\mathcal{S}$ from Figure 9. Any Petri net $\mathcal{N}$ such that $[\![\mathcal{N}]\!] = \mathrm{Pref}([\![\mathcal{S}]\!])$ would accept the causal net $\mathcal{K}$ from Figure 10 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 mean 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 3.3. *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 $\mathrm{Pref}(\llbracket \mathcal{S}_1 \rrbracket) \neq \llbracket \mathcal{S}_2 \rrbracket$ 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 $\mathrm{Pref}(\llbracket \mathcal{S}_1 \rrbracket) = \llbracket \mathcal{S}' \rrbracket$. 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. 1.7). Clearly $\mathrm{Pref}(\llbracket \mathcal{S}_1 \rrbracket) \subseteq \llbracket \mathcal{S}_2 \rrbracket$. Thus the difference between the specification $\mathcal{S}_1$ and the canonical implementation $\mathcal{S}_2$ stems from processes built upon 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].

3.1 An undecidable problem with Mazurkiewicz traces

The results presented in this section rely on the universality problem in the setting of Mazurkiewicz trace theory [8] 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 3.4. *[33, Theorem IV.4.3] (see also [32, Th. 2]) It is undecidable whether $[L] = \Sigma^\star$ for a given independence alphabet $(\Sigma, \|)$ and a given regular language $L \subseteq \Sigma^\star$.*

We shall use actually the immediate slightly stronger next statement.

COROLLARY 3.5. *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 3.4 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

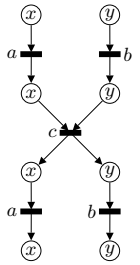

FIG. 11. Some process corresponding to the rule sequence $\rho(abcab)$ with $a \| b$

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 $[L] = \Sigma^\star$. ∎

3.2 From Mazurkiewicz traces to causal nets

To show that realizability is undecidable we reduce the decision problem from Corollary 3.5 to this problem. To do so we fix some independence alphabet $(\Sigma, \|)$ and some regular and prefix-closed language $L \subseteq \Sigma^\star$. The regular and prefix-closed language L is recognized by a finite automaton $\mathcal{A} = (Q, \imath, \longrightarrow_{\mathcal{A}})$ whose states are all accepting. We may assume that each state of $\mathcal{A}$ is reachable from the initial state and each action of Σ appears on a labeled arc of $\mathcal{A}$. We build a Petri net with states $\mathcal{S}$ which is realizable if and only if $[L] = \Sigma^\star$.

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 3.6. *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 3.7. 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 11 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. 2.3 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 3.8. *For all $u, v \in \Sigma^\star$: $u \sim v$ if and only if $[\![\rho(u)]\!]_{\mu_{\text{in}}} = [\![\rho(v)]\!]_{\mu_{\text{in}}}$.*

Proof. Let $u \in \Sigma^\star$ and $a, b \in \Sigma$ such that $a \neq b$. If $u.ab \sim u.ba$ then $a \| b$, $\text{Loc}(a) \cap \text{Loc}(b) = \emptyset$, and $[\![\rho(u.ab)]\!]_{\mu_{\text{in}}} = [\![\rho(u.ba)]\!]_{\mu_{\text{in}}}$ by Proposition 2.3. Therefore $u \sim v$ implies $[\![\rho(u)]\!]_{\mu_{\text{in}}} = [\![\rho(v)]\!]_{\mu_{\text{in}}}$ for all $u, v \in \Sigma^\star$ (again with the help of Prop. 2.3). 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 $[\![\rho(u)]\!]_{\mu_{\text{in}}} = [\![\rho(v)]\!]_{\mu_{\text{in}}}$. 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 Proposition 2.3 we have $[\![\rho(u')]\!]_{\mu_{\text{in}}} = [\![\rho(v')]\!]_{\mu_{\text{in}}}$. 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 $[\![\rho(u)]\!]_{\mu_{\text{in}}} = [\![\rho(v)]\!]_{\mu_{\text{in}}}$ 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 $[\![\rho(w')]\!]_{\mu_{\text{in}}}$. Moreover $[\![\rho(w'.a)]\!]_{\mu_{\text{in}}} = [\![\rho(v')]\!]_{\mu_{\text{in}}}$ and $[\![\rho(w'.b)]\!]_{\mu_{\text{in}}} = [\![\rho(u')]\!]_{\mu_{\text{in}}}$. 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 build from the automaton $\mathcal{A}$ the Petri net with states $\mathcal{S} = (Q, \imath, \longrightarrow_{\mathcal{S}}, \mu_{\text{in}})$ 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 $q_1 \xrightarrow{r}_{\mathcal{S}} q_2$ if $q_1 \xrightarrow{a}_{\mathcal{A}} q_2$. Observe here the multiset of tokens is left unchanged by each rule. Consequently the set of markings reached by prefixes of $[\![\mathcal{S}]\!]$ is finite, i.e. $\mathcal{S}$ is prefix-bounded. We can prove that $\mathcal{S}$ is realizable if and only if $[L] = \Sigma^\star$, which leads to the next result.

THEOREM 3.9. *It is undecidable whether a given prefix-bounded Petri net with states is realizable.*

Proof. We show that $\mathcal{S}$ is realizable if and only if $[L] = \Sigma^\star$. Assume first that $\mathcal{S}$ is realizable. Then Prop. 3.3 ensures that $[\![\mathcal{S}]\!] = [\![R_\Sigma^\star]\!]$. Let $u \in \Sigma^\star$. We have $[\![\rho(u)]\!]_{\mu_{\text{in}}} = [\![w]\!]_{\mu_{\text{in}}}$ for some $w \in \text{CS}(\mathcal{S})$. Let $v = \rho^{-1}(w)$. Clearly $v \in L$. Since $[\![\rho(u)]\!]_{\mu_{\text{in}}} = [\![\rho(v)]\!]_{\mu_{\text{in}}}$ we get $u \sim v$ by Lemma 3.8. Hence $\Sigma^\star = [L]$.

Assume now that $[L] = \Sigma^\star$. Let $w \in R_\Sigma^\star$. We have $\rho^{-1}(w) \in \Sigma^\star$. Then $\rho^{-1}(w) \sim u$ for some $u \in L$. It follows from Lemma 3.8 that $[\![w]\!]_{\mu_{\text{in}}} = [\![\rho(u)]\!]_{\mu_{\text{in}}}$. Moreover $\rho(u) \in \text{CS}(\mathcal{S})$. Therefore $[\![R_\Sigma^\star]\!]_{\mu_{\text{in}}} = [\![\text{CS}(\mathcal{S})]\!]_{\mu_{\text{in}}}$, i.e. $\mathcal{S}$ is realizable. ∎

3.3 Realizability and prefix-realizability are undecidable for VASSs

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

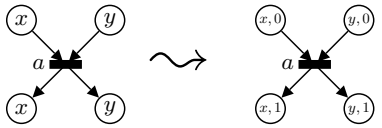

FIG. 12. Building the pure rule corresponding to a synchronisation rule

FIG. 13. 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 traces but we keep the same set of rule names Σ. Let 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 12 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 one as depicted in Figure 13. Formally 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 3.10. *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 3.11. *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 3.12. *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 3.13. *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. 3.3, $\mathcal{S}$ 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. 3.3 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. 3.10 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. 3.11. 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. 3.12, 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 3.14. *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 $\mathcal{K}_\bot \in \mathrm{Pref}(\llbracket \mathcal{S}^\circ_\bot \rrbracket)$. Thus $\mathcal{K}_\bot$ is a prefix of some process $\mathcal{K}' \in \llbracket u' \rrbracket_{\mu_{\mathrm{in}}}$ 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 $\mathcal{K} \in \llbracket u \rrbracket_{\mu_{\mathrm{in}}}$ and $\mathcal{K} \in \llbracket \mathcal{S}^\circ \rrbracket$.

Assume now that $\mathcal{S}^\circ$ is realizable. Then $\llbracket \mathcal{S}^\circ \rrbracket = \llbracket \mathcal{N}^\circ \rrbracket$ 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 $\llbracket \mathcal{S}^\circ_\bot \rrbracket = \llbracket \mathcal{N}^\circ_\bot \rrbracket$, so $\mathcal{S}^\circ_\bot$ is prefix-realizable. It is clear that $\llbracket \mathcal{S}^\circ_\bot \rrbracket \subseteq \llbracket \mathcal{N}^\circ_\bot \rrbracket$. Let $\mathcal{K} \in \llbracket \mathcal{N}^\circ_\bot \rrbracket$ 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 $\llbracket \mathcal{S}^\circ \rrbracket = \llbracket \mathcal{N}^\circ \rrbracket$, we have $\llbracket v \rrbracket_{\mu^\circ_{\mathrm{in}}} = \llbracket v' \rrbracket_{\mu^\circ_{\mathrm{in}}}$ for some $v' \in \mathrm{FCS}(\mathcal{S}^\circ)$. Then $\llbracket u \rrbracket_{\mu^\circ_{\mathrm{in}}} = \llbracket u' \rrbracket_{\mu^\circ_{\mathrm{in}}}$ for some $u' \in \mathrm{FCS}(\mathcal{S}^\circ_\bot)$ obtained from v' by adding possibly an occurrence some terminating rule. Therefore $\mathcal{K} \in \llbracket \mathcal{S}^\circ_\bot \rrbracket$. ∎

3.4 Gap between Petri nets and vector addition systems with states

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

Corollary 3.15. *Given two prefix-bounded vector addition systems with states $\mathcal{S}_1$ and $\mathcal{S}_2$, it is undecidable whether $\llbracket \mathcal{S}_1 \rrbracket = \llbracket \mathcal{S}_2 \rrbracket$ (resp. $\mathrm{Pref}(\llbracket \mathcal{S}_1 \rrbracket) = \llbracket \mathcal{S}_2 \rrbracket$, $\mathrm{Pref}(\llbracket \mathcal{S}_1 \rrbracket) = \mathrm{Pref}(\llbracket \mathcal{S}_2 \rrbracket)$).*

Proof. By Proposition 3.3, a VASS $\mathcal{S}$ is realizable if and only if $\llbracket \mathcal{S} \rrbracket = \llbracket \mathcal{S}' \rrbracket$ 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 1.7, we can effectively build $\mathcal{S}'$ from $\mathcal{S}$. By Theorem 3.13, $\llbracket \mathcal{S} \rrbracket = \llbracket \mathcal{S}' \rrbracket$ is undecidable. Therefore $\llbracket \mathcal{S}_1 \rrbracket = \llbracket \mathcal{S}_2 \rrbracket$ is undecidable for two vector addition systems with states $\mathcal{S}_1$ and $\mathcal{S}_2$.

Observe now that $\llbracket \mathcal{S}' \rrbracket = \mathrm{Pref}(\llbracket \mathcal{S}' \rrbracket)$. It follows that $\llbracket \mathcal{S}_1 \rrbracket = \mathrm{Pref}(\llbracket \mathcal{S}_2 \rrbracket)$ 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 $\mathrm{Pref}(\llbracket \mathcal{S} \rrbracket) = \mathrm{Pref}(\llbracket \mathcal{S}' \rrbracket)$. It follows from Theorem 3.14 that $\mathrm{Pref}(\llbracket \mathcal{S}_1 \rrbracket) = \mathrm{Pref}(\llbracket \mathcal{S}_2 \rrbracket)$ 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 3.16. *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two Petri nets. The property $\llbracket \mathcal{N}_1 \rrbracket \subseteq \llbracket \mathcal{N}_2 \rrbracket$ 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. 1.7). Then $\llbracket \mathcal{N}_1 \rrbracket \subseteq \llbracket \mathcal{N}_2 \rrbracket$ if and only if $R_1 \subseteq R_2$. ∎

4 Conclusion

We have studied a natural partial order semantics for vector addition systems with states (and Petri nets with states) which extends the classical non-branching process semantics of Place/Transition nets. We have shown how basic problems about the set of markings reached along concurrent executions, such as boundedness, covering and reachability, can be solved similarly to the usual analogous problems for Petri nets. However 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.

The synthesis problems considered in this paper are similar to the problems of realizability and safe realizability [1] in the setting of message sequence charts. They are not so close to the classical synthesis problems of Petri nets [7, 10, 19, 25] because the set of places and the set of transition rules are given. Thus there is no need of the notion of region to define an adequate set of places from the specification. The synthesis problems for asynchronous automata [8, 9, 35] are even more complicated since the set of rules and the set of places need to be computed from the specification.

Introduced as a formal model for the semantics of elementary Petri nets, Mazurkiewicz traces [24] are particular labeled partial orders that benefit from a rich and still growing theory [8]. 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 [18]). 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. F. Avellaneda and R. Morin. Checking non-divergence, channel-bound and global cooperation using SAT-solvers. In Benoît Caillaud, Josep Carmona, and Kunihiko Hiraishi, editors, *ACSD*, pages 19–28. IEEE, 2011.
4. 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.
5. E. Best and R. R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.

6. F. Bracho, M. Droste, and D. Kuske. Representation of computations in concurrent automata by dependence orders. *Theoretical Computer Science*, 174:67–96, 1997.
7. 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.
8. V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific Publ. Co., 1995.
9. M. Droste, P. Gastin, and D. Kuske. Asynchronous cellular automata for pomsets. *Theoretical Computer Science*, 247(1-2):1–38, 2000.
10. A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. Part II: State spaces of concurrent systems. *Acta Informatica*, 27(4):343–368, 1989.
11. J. Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28:575–591, 1991.
12. J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *Bulletin of the EATCS*, 52:244–262, 1994.
13. J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20:285–310, 2002.
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. 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.
19. 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.
20. J.E. Hopcroft and J-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979.
21. R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.
22. 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.
23. 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.
24. A. Mazurkiewicz. Concurrent program schemes and their interpretation. Technical report, DAIMI Report PB-78, Aarhus University, 1977.
25. M. Mukund. Petri nets and step transition systems. *International Journal of Foundations of Computer Science*, 3(4):443–478, 1992.
26. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.
27. J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1981.
28. 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.
29. C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.
30. W. Reisig. *Petri Nets, An Introduction.* EATCS Monographs, vol. 4. Springer, 1985.
31. C. Reutenauer. *The Mathematics of Petri Nets.* New York, USA: Prentice-Hall, 1990.
32. 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.
33. J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.

34. W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*, volume 625 of *Lecture Notes in Computer Science*. Springer, 1992.
35. W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. - Informatique Théorique et Applications*, 21:99–135, 1987.