Synthesis of Petri Nets from Finite Partial Languages

Robert Lorenz, Robin Bergenthum, Jörg Desel, Sebastian Mauser
Department of Applied Computer Science
Catholic University of Eichstätt-Ingolstadt
85072 Eichstätt, Germany
firstname.lastname@ku-eichstaett.de

Abstract

In this paper we present an algorithm to synthesize a finite place/transition Petri net (p/t-net) from a finite set of labeled partial orders (a finite partial language). This p/t-net has minimal non-sequential behavior including the specified partial language. Consequently, either this net has exactly the non-sequential behavior specified by the partial language, or there is no such p/t-net. We finally develop an algorithm to test whether the synthesized net has exactly the non-sequential behavior specified by the partial language.

The algorithms are based on the theory of regions for partial languages developed by Lorenz and Juhás. Thus, this paper shows the applicability of this concept and, for the first time, provides an effective algorithm for the synthesis of system models from partial languages.

1 Introduction

Synthesis of Petri nets from behavioral descriptions has been a successful line of research since the 1990s. There is a rich body of nontrivial theoretical results, and there are important applications in industry, in particular in hardware system design [3, 9], and recently also in workflow design [16]. Moreover, there are several synthesis tools that are based on the theoretical results [2].

Originally, synthesis means algorithmic construction of a Petri net from sequential observations. It can be applied to various classes of Petri nets, including elementary nets [6, 7] and place/transition nets (p/t-nets) [1]. Synthesis can start with a transition system representing the sequential behavior of a system or with a step transition system which additionally represents steps of concurrent events [1]. Synthesis can also be based on a language (a set of occurrence sequences or step sequences [4]). The *synthesis problem* is the problem to decide whether, for a given behavioral specification (transition system, language), there exists a Petri net of the respective class such that the behavior of this net coincides with the specified behavior. The aim of this paper is to solve the synthesis problem for p/t-nets where the behavior is given in terms of a finite partial language, i.e., as a finite set of labeled partial orders (LPOs – also known as *partial words* [8] or *pomsets* [15]). Moreover, we provide a synthesis algorithm. In contrast to previous work on the synthesis problem, we consider partial order behavior of Petri nets, truly representing the concurrency of events, which is often considered the most appropriate representation of behavior of Petri net models of concurrent systems.

As mentioned, we start with a finite set of partially ordered sets of events together with a labeling function associating a transition to each event. Thus, a single possible run (of the unknown p/t-net) is represented by an LPO of events. The ordering relation defines a *possible ordering of the transition occurrences*, i.e., if events e and e' are ordered ($e < e'$), and if moreover e is labeled by t and e' is labeled by t', then in this run t' can occur after the occurrence of t. If two events e and e' are not ordered (neither $e < e'$ nor $e' < e$), then the respective transitions can occur concurrently. This interpretation of a partial order semantics is different to the so-called process semantics, where $e < e'$ means that the respective occurrences of the labels, t and t', have to be causally ordered (and cannot be concurrent) whereas in our semantics $e < e'$ means that the respective transitions t and t' can either occur concurrently (and thus in any order) or in the order t earlier than t'. LPOs representing an order between specified transition occurrences, which is in the above described sense possible in a p/t-net, we formally call enabled w.r.t. this net.

Like previous results, our approach is based on the notion of regions. All approaches to Petri net synthesis based on regions roughly follow the same idea:

- Instead of solving the synthesis problem first (is there a net with the specified behavior?) and then – in the positive case – synthesizing the net, a net is synthesized for any specification.
- The construction starts with the transitions taken from the behavioral specification. In our case, transitions

are the labels of the events of the LPOs. So we start with a net with many transitions and without places.

- Since this net has too much concurrency in general, its behavior will be restricted by the addition of places. In particular, a place constitutes a dependency relation between the occurrences of the transitions in its pre-set and the occurrences of the transitions in its post-set.

- A single region identifies a dependency between two sets of transitions. Regions are defined for the behavioral description (in our case, for a partial language). Each region yields a corresponding place, together with its initial marking, in the constructed net. A region is defined in such a way that the behavior of the net with its corresponding place still includes the specified behavior. The same holds for any net with many places corresponding to regions.

- When all, or sufficiently many, regions are identified, all places of the synthesized net are constructed. The crucial point for this step is that the set of all regions can be very large or even infinite whereas in most cases finite, smaller sets of regions suffice to represent all relevant dependencies.

- If the behavior of the synthesized net coincides with the specified behavior (where coincide is defined by an appropriate notion of isomorphism), then the synthesis problem has a positive solution; otherwise there is no Petri net with the specified behavior and therefore the synthesis problem has a negative solution.

The notion of region employed in this work was already introduced in our previous work [12]. We showed that each region defines a place and that addition of all such places yields a net such that the behavior of this net includes the specified behavior. Moreover, there is no p/t-net with this property which enables fewer LPOs. If we could effectively check whether the behavior of the constructed net coincides with the specified behavior, we would be finished. However, this simple approach is infeasible for two reasons:

First, the set of regions defined previously is infinite, and so is the set of places of the synthesized net. In other words, this net can be defined mathematically, but it can never be constructed effectively. The first main result of this paper provides a solution to this problem. It shows that a finite subset of regions (places) suffices for generating the same behavior as all regions (places). It is based on a linear algebraic representation of regions and employs finiteness results from convex geometry.

The second problem addressed in this paper is concerned with the problem to check whether the behavior of this finite net coincides with the specified behavior. One possibility is to compute the (complete) behavior of the finite net (which itself turns out to be finite) and to compare it with the specified behavior. The behavior of the synthesized net can be computed through the set of its process nets (implemented in our tool VipTool [5]) considering the LPOs underlying process nets. Another possibility is to check for all LPOs not specified whether they do not belong to the behavior of the net. Since the set of not specified LPOs is infinite, we construct a finite representation of it. This way, the problem reduces to the problem of checking whether these finitely many LPOs are runs of the synthesized net. This can be solved using the verification result from [10].

The rest of the paper is organized as follows: We start with brief introduction to LPOs, partial languages, p/t-nets and enabled LPOs in Section 2. In Section 3 we recall definitions and main results from [12] on the theory of regions for partial languages. In the subsequent sections we develop the new results of this paper: In Section 4 we show how to compute regions as integer solutions of an homogenous linear inequation system (Subsection 4.1), and we prove that a finite set of basis solutions generating the set of all solutions already appropriately represents the set of all regions (Subsection 4.2). Finally, in Section 5 we present methods to test whether the synthesized finite p/t-net has exactly the specified non-sequential behavior.

2 Preliminaries

In this Section we recall the definitions of *labeled partial orders* (LPOs), *partial languages*, *place/transition nets* (p/t-nets) and LPOs *enabled w.r.t. p/t-nets*. We start with basic mathematical notations: By $\mathbb{N}$ we denote the *nonnegative integers*. $\mathbb{N}^+$ denotes the positive integers. Given a function f from A to B and a subset C of A, we write $f|_C$ to denote the *restriction* of f to the set C. Given a finite set A, the symbol $|A|$ denotes the *cardinality* of A. The set of all *multi-sets* over a set A is the set $\mathbb{N}^A$ of all functions $f : A \to \mathbb{N}$. Addition $+$ on multi-sets is defined as usual by $(m + m')(a) = m(a) + m'(a)$. We also write $\sum_{a\in A} m(a)a$ to denote a multi-set m over A and $a \in m$ to denote $m(a) > 0$. Given a binary relation $R \subseteq A \times A$ over a set A, the symbol R^+ denotes the *transitive closure* of R. We write aRb to denote $(a, b) \in R$. A *directed graph* is a pair $(V, \to)$, where V is a finite *set of vertices* and $\to \subseteq V \times V$ is a binary relation over V, called the *set of edges*. All graphs considered in this paper are finite.

Definition 1 (Partial order). *A* partial order *is a directed graph* $\mathrm{po} = (V, <)$*, where* $<$ *is a binary relation on* V *which is irreflexive (*$\forall v \in V : v \not< v$*) and transitive (*$<=<^+$*).*

Two nodes $v, v' \in V$ of a partial order $(V, <)$ are called *independent* if $v \not< v'$ and $v' \not< v$. By $\mathrm{co} \subseteq V \times V$ we denote the set of all pairs of independent nodes of V. A *co-set* is a subset $C \subseteq V$ fulfilling: $\forall x, y \in C: \ x \operatorname{co} y$. A *cut* is a maximal co-set. For a co-set C of a partial order $(V, <)$ and a node $v \in V \setminus C$ we write $v < C$, if $v < s$ for an element $s \in C$ and $v \operatorname{co} C$, if $v \operatorname{co} s$ for all elements $s \in C$. A partial order $(V', <')$ is a *prefix* of another partial order $(V, <)$ if $V' \subseteq V$ with $(v' \in V' \wedge v < v') \Longrightarrow (v \in V')$ and $<' = < \cap V' \times V'$. Given partial orders $\mathrm{po}_1 = (V, <_1)$ and $\mathrm{po}_2 = (V, <_2)$, po_2 is a *sequentialization* of po_1 if $<_1 \subseteq <_2$. We use partial orders with nodes *labeled by action names* to specify scenarios describing the behavior of systems.

Definition 2 (Labeled partial order). *A* labeled partial order *(LPO) is a triple* $\mathrm{lpo} = (V, <, l)$*, where* $(V, <)$ *is a partial order, and* $l : V \to T$ *is a* labeling function *with* set of labels T.

We use the above notations defined for partial orders also for LPOs. We will often consider LPOs only up to isomorphism. Two LPOs $(V, <, l)$ and $(V', <', l')$ are called *isomorphic*, if there is a bijective mapping $\psi : V \to V'$ such that $l(v) = l'(\psi(v))$ for $v \in V$, and $v < w \Longleftrightarrow \psi(v) <' \psi(w)$ for $v, w \in V$. By $[\mathrm{lpo}]$ we will denote the set of all LPOs isomorphic to lpo. The LPO lpo is said to *represent* the isomorphism class $[\mathrm{lpo}]$. The behavior of systems is formally specified by sets of (isomorphism classes of) LPOs. Such sets are also called *partial languages*.

Definition 3 (Partial language). *Let T be a set. A set $\mathcal{L} \subseteq \{[\mathrm{lpo}] \mid \mathrm{lpo}$ is an LPO with set of labels $T\}$ is called* partial language over T.

Usually, partial languages are given by sets of concrete LPOs representing isomorphism classes. We always assume that each label from T occurs in a partial language over T. Figure 1 shows a partial language represented by the set of LPOs $L = \{\mathrm{lpo}_1, \mathrm{lpo}_2\}$, which we will use as a running example.

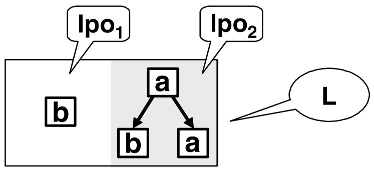

Figure 1. A partial language.

A *net* is a triple (P, T, F), where P is a (possibly infinite) set of *places*, T is a finite set of *transitions* satisfying $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*.

Definition 4 (Place/transition net). *A* place/transition-net *(*p/t-net*)* N *is a quadruple* (P, T, F, W)*, where* (P, T, F) *is a net, and* $W : F \to \mathbb{N}^+$ *is a* weight function.

We extend the weight function W to pairs of net elements $(x, y) \in (P \times T) \cup (T \times P)$ with $(x, y) \notin F$ by $W(x, y) = 0$. A *marking* of a net $N = (P, T, F, W)$ is a function $m : P \to \mathbb{N}$ assigning $m(p)$ tokens to a place $p \in P$, i.e. a multi-set over P. A *marked p/t-net* is a pair (N, m_0), where N is a p/t-net, and m_0 is a marking of N, called *initial marking*. Figure 2 shows a marked p/t-net (N, m_0). As usual, places are drawn as circles including tokens representing the initial marking, transitions as rectangles and the flow relation as arcs which have annotated the values of the weight function (the weight 1 is not shown).

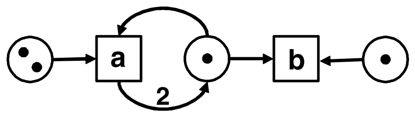

Figure 2. A marked p/t-net (N, m_0)**.**

A multi-set of transitions $\tau \in \mathbb{N}^T$ is called a *step* of N. A step τ is *enabled to occur* (concurrently) in a marking m if and only if $m(p) \geq \sum_{t \in \tau} \tau(t) W(p, t)$ for each place $p \in P$. In this case, its occurrence leads to the marking $m'(p) = m(p) + \sum_{t \in \tau} \tau(t)(W(t, p) - W(p, t))$. In the marked p/t-net (N, m_0) from Figure 2 only the steps a and b are enabled to occur in the initial marking. In the marking reached after the occurrence of a the step $a + b$ is enabled to occur. There are two equivalent formal notions of runs of p/t-nets defining non-sequential semantics based on [11, 17]. We only introduce the notion of enabled LPOs here: An LPO is enabled w.r.t. a marked p/t-net, if for each cut of the LPO the marking reached by firing all transitions corresponding to events smaller than the cut enables the step (of transitions) given by the cut.

Definition 5 (Enabled LPO). *Let* (N, m_0) *be a marked p/t-net,* $N = (P, T, F, W)$*. An LPO* $\mathrm{lpo} = (V, <, l)$ *with* $l : V \to T$ *is called* enabled (to occur) in (N, m_0) *if* $m_0(p) + \sum_{v \in V \wedge v < C}(W(l(v), p) - W(p, l(v))) \geq \sum_{v \in C} W(p, l(v))$ *for every cut* C *of* lpo *and every* $p \in P$*. Its* occurrence *leads to the* final marking m' *given by* $m'(p) = m_0(p) + \sum_{v \in V}(W(l(v), p) - W(p, l(v)))$.

The set of all isomorphism classes of LPOs enabled in (N, m_0) *is denoted by* $\mathfrak{Lpo}(N, m_0)$*.* $\mathfrak{Lpo}(N, m_0)$ *is called the* partial language of runs *of* (N, m_0)*. Enabled LPOs are also called* runs*.*

Observe that $\mathfrak{Lpo}(N, m_0)$ is always sequentialization and prefix closed, i.e. every sequentialization and every prefix of an enabled LPO is again enabled w.r.t. (N, m_0). Moreover, the set of labels of $\mathfrak{Lpo}(N, m_0)$ is always finite. Therefore, when specifying the non-sequential behavior of a searched p/t-net by a partial language, this partial language must be necessarily sequentialization and prefix closed and must have a finite set of labels. We assume that such a partial language $\mathcal{L}$ is given by a set of concrete LPOs L representing $\mathcal{L}$ in the sense that $[\mathrm{lpo}] \in \mathcal{L} \Longleftrightarrow \exists \mathrm{lpo}' \in$

L : $[\mathrm{lpo}] = [\mathrm{lpo}']$. Usually, we specify the non-sequential behavior by a set of concrete LPOs L which is *not* sequentialization and prefix closed and then consider the partial language which emerges by adding all prefixes of sequentializations of LPOs in L. In this sense, the partial language L given in Figure 1 specifies the non-sequential behavior of a searched p/t-net. Both LPOs shown in this Figure are enabled w.r.t. the marked p/t-net (N, m_0) shown in Figure 2. It even holds $\{[\mathrm{lpo}] \mid \mathrm{lpo} \text{ is a prefix of a sequentialization of an LPO in } L\} = \mathfrak{Lpo}(N, m_0)$. That means, (N, m_0) solves the synthesis problem w.r.t. L.

3 Region-based Synthesis

We consider the problem of synthesizing a p/t-net from a partial language specifying its non-sequential behavior. As mentioned, such a partial language $\mathcal{L}$ is represented by a set of concrete LPOs L (which is not necessarily prefix or sequentialization closed). That means we develop an algorithm to compute a marked p/t-net (N, m_0) from a given set of LPOs L such that the partial language $\mathcal{L}$ emerging from L satisfies $\mathcal{L} = \mathfrak{Lpo}(N, m_0)$ (if such a net exists). In this section we recall the definitions and main results on region based synthesis from [12] in a consolidated version, which is better structured and easier to understand: We explain the ideas of region based synthesis in two independent parts, first defining axiomatically the so called saturated feasible net as the best upper approximation to a p/t-net having the specified behavior and second introducing the notion of regions for the computation of this net.

3.1 Saturated Feasible Net

The idea to construct a net (N, m_0) solving the synthesis problem is as follows: The set of transitions of the searched net is the finite set of labels of L. Then each LPO in L is enabled w.r.t. the marked p/t-net consisting only of these transitions (having an empty set of places), because there are no causal dependencies between transitions. This net in general has many runs not specified by L. Thus, one restricts the behavior of this net by creating causal dependencies between the transitions through adding places. Such places are defined by their initial marking and the weights on the arcs connecting them to each transition (Figure 3).

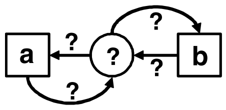

Figure 3. An unknown place of a p/t-net.

Two kinds of such places can be distinguished. In the case that there is an LPO in L which is not a run of the corresponding "one place"-net, this place restricts the behavior too much. Such a place is *non-feasible*. In the other case, the considered place is *feasible*.

Definition 6 (Feasible place). *Let $\mathcal{L}$ be a partial language over the finite set of labels T and let (N, m_p), $N = (\{p\}, T, F_p, W_p)$ be a marked p/t-net with only one place p (F_p, W_p, m_p are defined according to the definition of p). The place p is called* feasible (w.r.t. $\mathcal{L}$)*, if $\mathcal{L} \subseteq \mathfrak{Lpo}(N, m_p)$, otherwise* non-feasible (w.r.t. $\mathcal{L}$).

Figure 4 shows on the left side a place which is feasible w.r.t. the partial language specified by L in Figure 1. This is because, after the occurrence of a, the place is marked by 2 tokens. In this marking the step $a + b$ is enabled to occur (as specified by lpo_2). The place shown on the right side is non-feasible because, after the occurrence of a, the place is again marked by only 1 token (in this marking the step $a+b$ is not enabled to occur). Thus lpo_2 is not enabled w.r.t. the one-place-net shown on the right side.

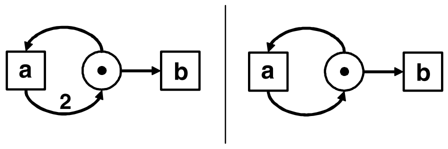

Figure 4. left part: a feasible place; right part: a place which is not feasible.

If we add all feasible places to the searched net, then the partial language of runs of the resulting net includes $\mathcal{L}$, and it is minimal with this property. We call this net the *saturated feasible net (w.r.t. $\mathcal{L}$)*. In general, the partial language of runs of the saturated feasible net is not necessarily equal to $\mathcal{L}$. If it is not equal to $\mathcal{L}$, there does not exist a marked p/t-net whose partial language of runs equals $\mathcal{L}$. That means the synthesis problem has a solution if and only if the partial language of runs of the saturated feasible net equals $\mathcal{L}$.

Definition 7 (Saturated feasible p/t-net). *Let $\mathcal{L}$ be a partial language over the finite set of labels T. The marked p/t-net (N, m_0), $N = (P, T, F, W)$, such that P is the set of all places feasible w.r.t. $\mathcal{L}$ is called* saturated feasible (w.r.t. $\mathcal{L}$) *(F, W, m_0 are defined according to the definitions of the feasible places).*

Theorem 8. *Let (N, m_0) be saturated feasible w.r.t. a partial language $\mathcal{L}$. Then it holds:*

(i) $\mathcal{L} \subseteq \mathfrak{Lpo}(N, m_0)$.

(ii) The behavior of (N, m_0) is minimal with property (i): $\forall (N' m_0') : (\mathfrak{Lpo}(N', m_0') \subsetneq \mathfrak{Lpo}(N, m_0)) \Longrightarrow (\mathcal{L} \not\subseteq \mathfrak{Lpo}(N', m_0'))$.

(iii) Either $\mathfrak{Lpo}(N, m_0) = \mathcal{L}$ or the synthesis problem has a negative answer.

Altogether, the saturated feasible net is a solution of the synthesis problem or there is no solution. Note that there are always infinitely many feasible places. For example, each place p_n with $W(a, p_n) = 2n$, $W(p_n, a) = n$, $W(p_n, b) = n$, $W(b, p_n) = 0$ and $m_0(p_n) = n$ is feasible w.r.t. the partial language given by L in Figure 1. Therefore, in particular the problem of representing the infinite set of feasible places by a finite subset (restricting the behavior in the same way) must be solved.

3.2 Regions

By so called *regions* of partial languages it is possible to define the set of all feasible places structurally on the level of the partial language given by L. The idea of defining regions of L is as follows: If two events x and y are ordered in an LPO $\mathrm{lpo} = (V, <, l) \in L$ – that means $x < y$ – this specifies that the corresponding transitions $l(x)$ and $l(y)$ are causally dependent. Such a causal dependency arises exactly if the occurrence of transition $l(x)$ produces tokens in a place, and some of these tokens are consumed by the occurrence of the other transition $l(y)$. Such a place can be defined as follows: Assign to every edge (x, y) of an LPO in L a natural number representing *the number of tokens which are produced by the occurrence of $l(x)$ and consumed by the occurrence of $l(y)$ in the place to be defined*. Then the number of tokens consumed overall by a transition $l(y)$ in this place is given as the sum of the natural numbers assigned to ingoing edges (x, y) of y. This number can then be interpreted as the weight of the arc connecting the new place with the transition $l(y)$. Similarly, the number of tokens produced overall by a transition $l(x)$ in this place is given as the sum of the natural numbers assigned to outgoing edges (x, y) of x, and this number can then be interpreted as the weight of the arc connecting the transition $l(x)$ with the new place. Moreover, transitions can also

- consume tokens from the initial marking of the new place (tokens which are not produced by another transition): In order to specify the number of such tokens, we extend an LPO by an *initial event* v_0 representing a transition producing the initial marking.
- produce tokens in the new place which remain in the final marking after the occurrence of all transitions (tokens which are not consumed by some subsequent transition): In order to specify the number of such tokens, we extend an LPO by a *final event* v_{max} representing a transition consuming the final marking.

The sum of the natural numbers assigned to outgoing edges (v_0, y) of the initial event v_0 can be interpreted as the initial marking of the new place.

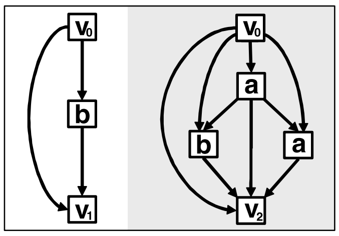

Figure 5. $\star$-extensions of LPOs.

Figure 5 shows the LPOs lpo_1 and lpo_2 from Figure 1 extended by an initial and a final event. Such extensions we call $\star$-extensions of LPOs.

Definition 9 ($\star$-extension)**.** *For a set of LPOs L we denote $W_L = \bigcup_{(V,<,l)\in L} V$, $E_L = \bigcup_{(V,<,l)\in L} <$ and $l_L = \bigcup_{(V,<,l)\in L} l$. A* $\star$-extension $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ *of* $\mathrm{lpo} = (V, <, l)$ *is defined by*

(i) $V^\star = (V \cup \{v_0^{lpo}, v_{\max}^{lpo}\})$ *with* $v_0^{lpo}, v_{\max}^{lpo} \notin V$*,*

(ii) $\prec^\star = \prec \cup (\{v_0^{lpo}\} \times V) \cup (V \times \{v_{\max}^{lpo}\}) \cup \{(v_0^{lpo}, v_{\max}^{lpo})\}$*,*

(iii) $l^\star(v_0^{lpo}), l^\star(v_{\max}^{lpo}) \notin l(V)$*,* $l^\star(v_0^{lpo}) \neq l^\star(v_{\max}^{lpo})$ *and* $l^\star|_V = l$*.*

v_0^{lpo} *is called* initial event of lpo *and* $v_{\max}^{lpo}$ maximal event of lpo. *Let* $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ *be a $\star$-extension of each* $\mathrm{lpo} \in L$ *such that:*

(iv) For each two LPOs $(V, <, l), (V', <', l') \in L$*:* $l^\star(v_0^{lpo}) = (l')^\star(v_0^{lpo'})$ *.*

(v) For each two distinct LPOs $(V, <, l), (V', <', l') \in L$*:* $l^\star(v_{\max}^{lpo}) \neq (l')^\star(v_{\max}^{lpo'})$ *(*$\notin l_L(W_L)$*).*

Then the set $L^\star = \{\mathrm{lpo}^\star \mid \mathrm{lpo} \in L\}$ *is called* $\star$-extension of L. *We denote* $W_L^\star = W_{L^\star}$*,* $E_L^\star = E_{L^\star}$ *and* $l_L^\star = l_{L^\star}$*.*

According to the above explanation, we can define a new place p_r by assigning in each LPO $\mathrm{lpo} = (V, <, l) \in L$ a natural number $r(x, y)$ to each edge (x, y) of a $\star$-extension of lpo through a function $r : E_L^\star \to \mathbb{N}_0$:

- The sum of the natural numbers $In_{\mathrm{lpo}}(y, r) = \sum_{x <^\star y} r(x, y)$ assigned to ingoing edges (x, y) of a node $y \in W_L$ defines $W(p_r, l(y)) = In_{\mathrm{lpo}}(y, r)$. We call $In_{\mathrm{lpo}}(y, r)$ the *intoken flow* of y.
- The sum of the natural numbers $Out_{\mathrm{lpo}}(x, r) = \sum_{x <^\star y} r(x, y)$ assigned to outgoing edges (x, y) of a node $x \in W_L$ defines $W(l(x), p_r) = Out_{\mathrm{lpo}}(x, r)$. We call $Out_{\mathrm{lpo}}(x, r)$ the *outtoken flow* of x.
- the sum of the natural numbers assigned to outgoing edges (v_0, y) of an initial node v_0^{lpo} (the outtoken flow of v_0^{lpo}) defines $m_0(p_r) = Out_{\mathrm{lpo}}(v_0^{\mathrm{lpo}}, r)$. We call $Out_{\mathrm{lpo}}(v_0^{\mathrm{lpo}}, r)$ the *initial token flow* of lpo.

The value $r(x, y)$ we call the *token flow* between x and y. Since equally labeled nodes formalize occurrences of the same transition, this is well-defined only if equally labeled events have equal intoken flow and equal outtoken flow. In particular all LPOs must have the same initial token flow. We say that $r : E^\star_L \to \mathbb{N}_0$ fulfills the properties (IN) and (OUT) on L if for all $\mathrm{lpo} = (V, <, l), \mathrm{lpo}' = (V', <', l') \in L$ and for all $v \in V^\star, v' \in (V')^\star$ holds

(IN) $l(v) = l'(v') \Longrightarrow In_{\mathrm{lpo}}(v, r) = In_{\mathrm{lpo}'}(v', r)$.

(OUT) $l(v) = l'(v') \Longrightarrow Out_{\mathrm{lpo}}(v, r) = Out_{\mathrm{lpo}'}(v', r)$.

Observe that (OUT) in particular ensures that all LPOs have the same initial token flow. Altogether, each such function r fulfilling (IN) and (OUT) on L defines a place p_r. We call p_r *corresponding place* of r.

Definition 10 (Region)**.** *Let L be a set of LPOs which is sequentialization and prefix closed. Let further $\mathcal{L}$ be the partial language represented by L. A* region *of $\mathcal{L}$ is a function $r : E^\star_L \to \mathbb{N}_0$ fulfilling (IN) and (OUT) on L.*

If we define a function r fulfilling (IN) and (OUT) on a set of LPOs L which is not sequentialization and prefix closed, then this function is easily extended to a region of the partial language defined by the set of all prefixes of sequentializations of LPOs in L as follows:

- Assign the value 0 to each additional edge within a sequentialization of an LPO in L and keep the values of r on all other edges.

- Define r on a prefix of an LPO in L by gluing all nodes subsequent to the prefix to a maximal node of the prefix. If several edges are glued to one edge, then sum up the values of r on the glued edges. Keep the values of r on all remaining edges.

Thus, it is enough to specify a function fulfilling (IN) and (OUT) on some set of LPOs L to define a region of the partial language $\mathcal{L}$ defined by L. Figure 6 shows a function r fulfilling (IN) and (OUT) on the set L of LPOs given in Figure 1, which in this sense can be extended to a region of the partial language defined by L. The corresponding place p_r is defined by $W(p_r, a) = 1$, $W(a, p_r) = 2$, $W(p_r, b) = 1$, $W(b, p_r) = 0$ and $m_0(p_r) = 1$ (p_r is the middle place of the p/t-net in Figure 2).

As the main result we showed in [12] that the set of places corresponding to regions of a partial language equals the set of feasible places w.r.t. this partial language.[1]

[1]In [12] we assumed that the set of LPOs L representing $\mathcal{L}$ fulfills some technical requirements. These will be automatically fulfilled for all such sets L we consider in the following. Thus, we omit their detailed presentation here.

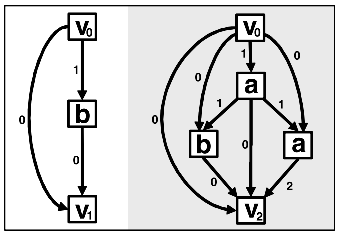

Figure 6. Region of a partial language.

Theorem 11 ([12])**.** *Let $\mathcal{L}$ be a partial language. Then it holds (i) that each place corresponding to a region of $\mathcal{L}$ is feasible w.r.t. $\mathcal{L}$ and (ii) that each place feasible w.r.t. to $\mathcal{L}$ corresponds to a region of $\mathcal{L}$.*

Thus the saturated feasible net can be given by the set of places corresponding to regions:

Corollary 12. *Let $\mathcal{L}$ be a partial language represented by the set of LPOs L. Denote $P = \{p_r \mid r \text{ is a region of } \mathcal{L}\}$, T the set of labels of $\mathcal{L}$, $W(p_r, l_L(v)) = In_{\mathrm{lpo}}(v, r)$ and $W(l_L(v), p_r) = Out_{\mathrm{lpo}}(v, r)$ for $p_r \in P$ and some $\mathrm{lpo} = (V, <, l) \in L$ with $v \in V$, $F = \{(x, y) \mid W(x, y) > 0\}$ and $m_{sat}(p_r) = Out_{\mathrm{lpo}}(v_0^{\mathrm{lpo}}, r)$ for $p_r \in P$ (and some $\mathrm{lpo} \in L$). Then the p/t-net (N_{sat}, m_{sat}), $N_{sat} = (P, T, F, W)$, is the* saturated feasible p/t-net (w.r.t. $\mathcal{L}$).

Remember that the saturated feasible net has infinitely many places, i.e. there are infinite many regions of $\mathcal{L}$. Moreover, even the description of one region may be infinite, since there may exist infinitely many edges in $E^\star_L$. Therefore we restrict ourselves in the following to finite partial languages, i.e. to partial languages which are represented by a finite set of LPOs L.

4 Computing a Finite Representation of all Regions

For finite partial languages we show in this section, that the set of regions can be computed as the set of non-negative integer solutions of a homogenous linear equation system $\mathbf{A} \cdot \mathbf{x} = \mathbf{0}$ (Subsection 4.1). It is well known that there is a finite set of basis-solutions, such that every solution is generated as a non-negative linear sum of basis-solutions. In Subsection 4.2 we prove that the set of places corresponding to basis-solutions already restricts the behavior of the searched net in the same way as the set of all feasible places. That means there is a representation of the saturated feasible net by a net with finitely many places having the same partial language of runs. For this finite net it can be tested effectively if it has $\mathcal{L}$ as its partial language of runs (Section 5).

4.1 Computing Regions

In this subsection we show how to compute regions (and thus feasible places) of a partial language $\mathcal{L}$ represented by a finite set of LPOs L. For this, we rewrite the properties (IN) and (OUT) as a homogenous linear equation system $\mathbf{A}_L \cdot \mathbf{x} = \mathbf{0}$. The LPOs in L are assumed to have pairwise disjoint node sets.[2] To compute a region r, we need to assign a value $r(x, y)$ to every edge $e = (x, y)$ in the finite set of edges $E_L^\star$. We interpret r as a $|E_L^\star|$-dimensional vector $\mathbf{x}_r = (x_1, \dots, x_n)$, $n = |E_L^\star|$. Considering a fixed numbering of the edges in $E_L^\star = \{e_1, \dots, e_n\}$, a value $r(e_i)$ equals x_i. Figure 7 shows a numbering of the edges of the $\star$-extension of the set of LPOs L given in Figure 1.

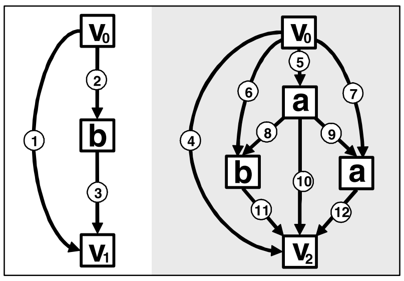

Figure 7. A numbering of edges.

Now, we encode the properties (IN) and (OUT) by a homogenous linear equation system $\mathbf{A}_L \cdot \mathbf{x} = \mathbf{0}$ in the sense that $r : E_L^\star \to \mathbb{N}_0$ fulfills (IN) and (OUT) on L if and only if $\mathbf{A}_L \cdot \mathbf{x}_r = \mathbf{0}$. This can be done by, loosely speaking, defining for pairs of equally labeled nodes a row $\mathbf{a}$ of $\mathbf{A}_L$ counting the token flow on ingoing edges of one node positively and of the other node negatively. Similarly, a row $\mathbf{b}$ of $\mathbf{A}_L$ counting the token flow on outgoing edges of one node positively and of the other node negatively can be defined. It is enough for each label t to ensure, that the intoken (outtoken) flow of the first and second node with label t are equal, that the intoken (outtoken) flow of the second and third node with label t are equal, and so on.

Formally, we denote $W_t = \{v \in W_L^\star \mid l_L^\star(v) = t\} = \{v_1^t, v_2^t, \dots\}$ for all labels $t \in T$ and denote

$$\begin{aligned}\mathbf{a}_m^t &= (a_{m,1}^t, \dots, a_{m,n}^t)\\ \mathbf{a}_{m,j}^t &= \begin{cases} 1 & \text{if } e_j \text{ is an ingoing edge of } v_m^t,\\ -1 & \text{if } e_j \text{ is an ingoing edge of } v_{m+1}^t\\ 0 & \text{else.}\end{cases}\end{aligned}$$

for $1 \leqslant m \leqslant |W_t| - 1$. Clearly, $\mathbf{a}_m^t \cdot \mathbf{x}_r = \mathbf{0}$ if and only if $In_{\text{lpo}}(v_m^t, r) = In_{\text{lpo}'}(v_{m+1}^t, r)$ for the LPOs $\text{lpo} = (V, <, l)$ and $\text{lpo}' = (V', <', l')$ with $v_m^t \in V$ and $v_{m+1}^t \in V'$.

[2]This ensures that L requires all technical requirements used in [12] to prove Theorem 11.

Similarly, we set

$$\begin{aligned}\mathbf{b}_m^t &= (b_{m,1}^t, \dots, b_{m,n}^t)\\ \mathbf{b}_{m,j}^t &= \begin{cases} 1 & \text{if } e_j \text{ is an outgoing edge of } v_m^t,\\ -1 & \text{if } e_j \text{ is an outgoing edge of } v_{m+1}^t\\ 0 & \text{else.}\end{cases}\end{aligned}$$

for $1 \leqslant m \leqslant |W_t| - 1$. Clearly, $\mathbf{b}_m^t \cdot \mathbf{x}_r = \mathbf{0}$ if and only if $Out_{\text{lpo}}(v_m^t, r) = Out_{\text{lpo}'}(v_{m+1}^t, r)$ for the LPOs $\text{lpo} = (V, <, l)$ and $\text{lpo}' = (V', <', l')$ with $v_m^t \in V$ and $v_{m+1}^t \in V'$.

Finally, to ensure that all LPOs have the same initial token flow, we denote $L = \{\text{lpo}_1, \text{lpo}_2, \dots\}$ and add rows

$$\begin{aligned}\mathbf{c}_m &= (c_{m,1}, \dots, c_{m,n})\\ \mathbf{c}_{m,j} &= \begin{cases} 1 & \text{if } e_j \text{ is an outgoing edge of } v_0^{\text{lpo}_m},\\ -1 & \text{if } e_j \text{ is an outgoing edge of } v_0^{\text{lpo}_{m+1}}\\ 0 & \text{else.}\end{cases}\end{aligned}$$

for $1 \leqslant m \leqslant |L| - 1$. Clearly, $\mathbf{c}_m \cdot \mathbf{x}_r = \mathbf{0}$ if and only if $Out_{\text{lpo}_m}(v_0^{\text{lpo}_m}, r) = Out_{\text{lpo}_{m+1}}(v_0^{\text{lpo}_{m+1}}, r)$.

$$\begin{array}{cccccccccccc} ① & ② & ③ & ④ & ⑤ & ⑥ & ⑦ & ⑧ & ⑨ & ⑩ & ⑪ & ⑫ \end{array}$$
$$\begin{pmatrix} 1 & 1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & -1\\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \bullet \begin{pmatrix} ①\\ ②\\ ③\\ ④\\ ⑤\\ ⑥\\ ⑦\\ ⑧\\ ⑨\\ ⑩\\ ⑪\\ ⑫ \end{pmatrix} = \begin{pmatrix} \mathbf{0}\\ \mathbf{0}\\ \cdots\\ \mathbf{0}\\ \mathbf{0} \end{pmatrix}$$

Figure 8. Equation system defining regions.

Figure 8 shows the described homogenous linear equation system $\mathbf{A}_L \cdot \mathbf{x} = \mathbf{0}$ for the numbering of edges given in Figure 7. The first row of the matrix ensures that both LPOs have the same initial token flow. Therefore, the sum of the values on all outgoing edges of $v_0^{\text{lpo}_1}$ (namely e_1 and e_2) must equal the sum of the values on all outgoing edges of $v_0^{\text{lpo}_2}$ (namely e_4, e_5, e_6 and e_7). We get the corresponding equation $x_1 + x_2 - x_4 - x_5 - x_6 - x_7 = 0$ (this equation corresponds to the first row $\mathbf{c}_1$ of $\mathbf{A}_L$). Moreover, there exist two pairs of equally labeled nodes, and we need to ensure that each pair has the same intoken and outtoken flow. Row number two $\mathbf{a}_1^a$ ensures for every function r given by a solution x_r that both a-labeled nodes have the same intoken flow, row number three $\mathbf{b}_1^a$ guarantees equal outtoken flow of the a-labeled nodes. Rows number four $\mathbf{a}_1^b$ and five $\mathbf{b}_1^b$ do the same for both nodes labeled by b. A possible non-negative integer solution would be $\mathbf{x}_r = (0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 2)$ corresponding to the region drawn in Figure 6 (and to the middle place shown in Figure 2).

By the above considerations the set of regions r is in one-to-one-correspondence to the set of non-negative integer solutions $\mathbf{x} = (x_1, \ldots, x_n)$ of $\mathbf{A}_L \cdot \mathbf{x} = \mathbf{0}$ via $r(e_i) = x_i$. That means, every feasible place can be computed by such a solution. The place corresponding to a solution $\mathbf{x}$ we denote by $p_{\mathbf{x}}$. Note that the number of rows N of $\mathbf{A}_L$ linearly depends on the number of nodes $|W_L|$ and the number of LPOs $|L|$.

4.2 Finite Representation

The homogenous linear equation system developed in the last section is in fact an inequation system, since we search for non-negative solutions, i.e. we require $\mathbf{x} \geqslant 0$ for solutions $\mathbf{x}$. Thus we compute regions of a finite partial language $\mathcal{L}$ and subsequently places of the searched saturated feasible p/t-net by solving the finite homogenous linear inequation system $\mathbf{A}_L \cdot \mathbf{x} \leqslant \mathbf{0}$, $-\mathbf{A}_L \cdot \mathbf{x} \leqslant \mathbf{0}$, $-\mathbf{x} \leqslant \mathbf{0}$ with $n + 2N$ rows (N is the number of rows, n the number of columns of $\mathbf{A}_L$). The set of solutions of such a system is called a *polyhedral cone*. According to a theorem of Minkowski [13] polyhedral cones are finitely generated, i.e. there are finitely many vectors $\mathbf{y}_1, \ldots, \mathbf{y}_k$ (also called *basis solutions*) such that each element $\mathbf{x}$ of the polyhedral cone is a non-negative linear sum $\mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{y}_i$ for some $\lambda_1, \ldots, \lambda_k \geqslant 0$. Such basis solutions $\mathbf{y}_1, \ldots, \mathbf{y}_k$ can be effectively computed from $\mathbf{A}_L$ (see for example [14]). If all entries of $\mathbf{A}_L$ are integral, then also the entries of all $\mathbf{y}_i$ can be chosen to be integral. The time complexity of the computation essentially depends on the number k of basis solution which is bounded by $k \leqslant \binom{n+2N}{n-1}$. That means, in the worst case the time complexity is exponential in the number of nodes, whereas in most practical examples of polyhedral cones there are only few basis solutions. It is a topic of further research to evaluate k for typical instances of polyhedral cones in our setting.

We finally claim that all places which do not correspond to basis solutions can be deleted from the saturated feasible p/t-net without changing its partial language of runs. Thus, the saturated feasible p/t-net has a finite representation. Consider places $p, p_1, \ldots, p_k$ of some marked p/t-net (N, m_0), $N = (P, T, F, W)$, and non-negative real numbers $\lambda_1, \ldots, \lambda_k$ ($k \in \mathbb{N}$) such that (i) $m_0(p) = \sum_{i=1}^{k} \lambda_i m_0(p_i)$, (ii) $W(p, t) = \sum_{i=1}^{k} \lambda_i W(p_i, t)$ for all transitions t and (iii) $W(t, p) = \sum_{i=1}^{k} \lambda_i W(t, p_i)$ for all transitions t. In such a case we write $p = \sum_{i=1}^{k} \lambda_i p_i$. Figure 9 shows the p/t-net N from Figure 2 extended to a net N' by adding the two places p_4 and p_5. Neither p_4 nor p_5 restrict the behavior of N' more then $\{p_1, p_2, p_3\}$. In other words each LPO enabled in N is also enabled in N'. That is because the places p_4 and p_5 are positive linear combinations of the other three places. It holds $p_5 = 2p_3$ and $p_4 = \frac{1}{2}p_1 + \frac{1}{2}p_2 + \frac{1}{2}p_3$.

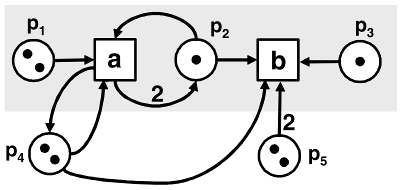

Figure 9. Summing places.

Lemma 13. *Let (N, m_0), $N = (P, T, F, W)$, be a marked p/t-net with $P = \{p_1, \ldots, p_k, p\}$ and $p = \sum_{i=1}^{k} \lambda_i p_i$ for non-negative real numbers $\lambda_1, \ldots, \lambda_k$ ($k \in \mathbb{N}$). Denote $P' = \{p_1, \ldots, p_k\}$, $m_0' = m_0|_{P'}$ and $N' = (P', T, F|_{(P' \times T) \cup (T \times P')}, W|_{(P' \times T) \cup (T \times P')})$. Then each LPO enabled w.r.t. (N', m_0') is enabled w.r.t. (N, m_0).*

Proof. Let lpo be enabled w.r.t. (N', m_0'), $\mathrm{lpo} = (V, <, l)$. According to Definition 5, for a cut C of lpo and $i \in \{1, \ldots, k\}$ it holds $m_0(p_i) + \sum_{v \in V \wedge v < C}(W(l(v), p_i) - W(p_i, l(v))) \geq \sum_{v \in C} W(p_i, l(v))$. This implies for an arbitrary cut C of lpo and the place p: $m_0(p) + \sum_{v \in V \wedge v < C}(W(l(v), p) - W(p, l(v))) = \sum_{i=1}^{k} \lambda_i (m_0(p_i) + \sum_{v \in V \wedge v < C}(W(l(v), p_i) - W(p_i, l(v)))) \geqslant \sum_{i=1}^{k} \lambda_i \sum_{v \in C} W(p_i, l(v)) = \sum_{v \in C} W(p, l(v))$. Thus, lpo is enabled w.r.t. (N, m_0). □

If $\mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{y}_i$ for basis solutions $\mathbf{y}_1, \ldots, \mathbf{y}_k$ of $\mathbf{A}_L \cdot \mathbf{x} = \mathbf{0}$, then $p_{\mathbf{x}} = \sum_{i=1}^{k} \lambda_i p_{\mathbf{y}_i}$. Thus, the finite net (N, m) having the places $p_{\mathbf{y}_1}, \ldots, p_{\mathbf{y}_k}$ satisfies $\mathfrak{Lpo}(N_{sat}, m_{sat}) = \mathfrak{Lpo}(N, m)$. That means, (N, m) generates the smallest partial language of runs including L. To compute (N, m), we compute such a finite set of integer basis solutions $\mathbf{y}_1, \ldots, \mathbf{y}_k$ (Algorithm 1).

```
1: $A \leftarrow EmptyMatrix$
2: for all $t \in T$ do
3:    $W_t \leftarrow \{v \in W_L^\star \mid l_L^\star(v) = t\}$
4:    for $m = 1$ to $|W_t| - 1$ do
5:       $A.addRow(a_m^t)$
6:       $A.addRow(b_m^t)$
7:    end for
8: end for
9: for $m = 1$ to $|L| - 1$ do
10:   $A.addRow(c_m)$
11: end for
12: $Solutions \leftarrow A.getBasisSolutions$
13: $(N, m) \leftarrow (\emptyset, T, \emptyset, \emptyset, \emptyset)$
14: for all $r \in Solutions$ do
15:   $(N, m).addCorrespondingPlace(r)$
16: end for
```

Algorithm 1: Calculates a net (N, m) from a partial language over T given by L, such that (N, m) generates the smallest partial language of runs including L.

5 Equality Test

Up to now, we have shown how to compute from a finite set of LPOs L a finite marked p/t-net (N, m) which has the smallest partial language of runs $\mathfrak{Lpo}(N, m)$ including the specified partial language $\mathcal{L} = \{[\text{lpo}] \mid \text{lpo is a prefix of a sequentialization of an LPO in } L\}$. This net either solves the synthesis problem ($\mathfrak{Lpo}(N, m) = \mathcal{L}$) or there is no solution. In this section we develop two possibilities to test whether $\mathfrak{Lpo}(N, m) = \mathcal{L}$.

Let L_{sp} be the set of all sequentilizations of prefixes of LPOs in L. Since we already know $\mathfrak{Lpo}(N, m) \supseteq \mathcal{L}$, in order to test $\mathfrak{Lpo}(N, m) = \mathcal{L}$, we (1) either have to check if each enabled lpo of (N, m) is isomorphic to an LPO in L_{sp} (optimistic equality test), or (2) to test that no LPO lpo which is not isomorphic to an LPO in L_{sp} is enabled w.r.t. (N, m) (pessimistic equality test).

5.1 Optimistic Equality Test

In the first case (1), we calculate all enabled LPOs of (N, m). The set of (pairwise non-isomorphic) enabled LPOs of a p/t-net in general can be infinite, but $\mathfrak{Lpo}(N, m)$ is always finite. This can be proven for $\mathfrak{Lpo}(N_{sat}, m_{sat})$ $(= \mathfrak{Lpo}(N, m))$ as follows: For every transition t and every LPO $\text{lpo} = (V, <, l) \in L$ there is a finite number $n_{\text{lpo},t}$ of nodes $v \in V$ labeled by t. Since L is finite we get a finite upper bound $n_t = max(\{n_{\text{lpo},t} \mid \text{lpo} \in L\})$ for the maximal number of occurrences of t in an LPO $\text{lpo} \in L$. Consequently the place p_t with the initial marking $m_0(p_t) = n_t$, an empty pre-set and t as the only transition in its post-set with $W(p_t, t) = 1$ is feasible w.r.t. $\mathcal{L}$. That means, that each transition t can maximally occur n_t-times, and thus every LPO in $\mathfrak{Lpo}(N_{sat}, m_{sat})$ has at most $\sum_{t \in T} n_t$ nodes.

Since $\mathfrak{Lpo}(N, m)$ is finite, it can be calculated. In principle, we have to check if each run $\text{lpo} \in \mathfrak{Lpo}(N, m)$ is isomorphic to an LPO in L_{sp}. But for a run lpo' which is a sequentialization of a prefix of another run lpo, it is enough to consider only lpo, because if lpo' is not isomorphic to an LPO in L_{sp}, then the same holds for lpo. Therefore, we only regard runs which are not sequentializations of prefixes of other runs. The set of all such runs can be computed through the (finite) set of process nets with maximal length of (N, m) [17]: Omitting conditions in a process net and only keeping the ordering between events yields an LPO, and it is well known that each such LPO underlying a process net is a run. Moreover, each run is a sequentialization of a prefix of an LPO underlying a process net with maximal length. Thus, it is enough to regard the LPOs underlying such process nets of (N, m). The synthesis problem has a solution if and only if each such LPO is isomorphic to some LPO in L_{sp}. For example, the runs of the p/t-net shown in Figure 2, which are not sequentializations of prefixes of other runs, are exactly the LPOs shown in Figure 1.

An algorithm that calculates the set of maximal process nets of a p/t-net is for example implemented in our tool VipTool [5]. In general, the number of process nets is exponential in the size of the p/t-net, and the calculation of the process nets requires an exponential runtime. But in our special situation we expect that the number of process nets of (N, m) roughly coincides with the size of L, because in the case that there is a positive solution of the synthesis problem there holds $\mathfrak{Lpo}(N, m) = \mathcal{L}$, and in the negative case $\mathfrak{Lpo}(N, m)$ is the best upper approximation to $\mathcal{L}$. There can easily be developed heuristics to find (in the negative case) enabled LPOs not in L_{sp} before the whole set of process nets of (N, m_0) is constructed.

5.2 Pessimistic Equality Test

The alternative possibility (2) to test $\mathfrak{Lpo}(N, m) = \mathcal{L}$ is to check if no LPO lpo not isomorphic to some LPO in L_{sp} is in $\mathfrak{Lpo}(N, m)$. For one such LPO lpo this can be tested in polynomial time in the number of nodes of lpo using the algorithm we presented in [10]. The problem is, that there are infinite many such LPOs. Therefore, we define a finite set L^c_{fin} of LPOs representing the set of all LPOs L^c not specified by L in the following sense: if no LPO in L^c_{fin} is enabled in (N, m) then also no LPO in L^c is enabled in (N, m). The idea for the construction of L^c_{fin} is to extend each $\text{lpo} \in L_{sp}$ in all possible ways by one event, such that the resulting LPO lpo' is not isomorphic to an LPO in L_{sp}. That means, L^c_{fin} consists of all LPOs lpo' not isomorphic to an LPO in L_{sp} defined by $\text{lpo}' = (V \cup \{v_t\}, < \cup <_t, l \cup (v_t, t))$, where $(V, <, l) \in L_{sp}$, $t \in T$, $v_t \notin V$ and $<_t = \{v' \mid v' \in V' \vee v' < V'\} \times \{t\}$ for a co-set V' of $(V, <, l)$ (V' may be empty, which means that v_t becomes an additional minimal event). Figure 10 shows some of the LPOs in L^c_{fin} for the set of LPOs L shown in Figure 1. The most left LPO is constructed from the empty co-set V'.

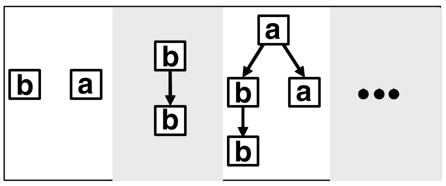

Figure 10. Some LPOs in L^c_{fin}.

If there exists an LPO $\text{lpo}' \in L^c_{fin}$ which is enabled in (N, m), then obviously $\mathcal{L} \neq \mathfrak{Lpo}(N, m)$. On the other hand, if every such LPO lpo' is not enabled in (N, m), we conclude that $\mathcal{L} = \mathfrak{Lpo}(N, m)$. This can be proven as follows by contradiction: Assume that there exists an $\text{lpo} \in L^c$

which is enabled in (N, m). Then, there is a maximal prefix lpo_{pre} of lpo (possibly empty) isomorphic to an LPO in L_{sp}. Let lpo'_{pre} be a further prefix of lpo having one additional node (such lpo'_{pre} exists because lpo is not isomorphic to an LPO in L_{sp}). The maximality of lpo_{pre} implies that lpo'_{pre} is not isomorphic to an LPO in L_{sp}. By construction of L^c_{fin}, we conclude that lpo'_{pre} is isomorphic to an LPO in L^c_{fin}. Since lpo'_{pre} is a prefix of an LPO enabled in (N, m), it is also enabled in (N, m). This is a contradiction. Note that the set L^c_{fin} in general can have exponentially many LPOs in the size of L. We are currently working on methods to reduce the set L^c_{fin}.

6 Conclusion

In this paper we presented, given a finite set of LPOs representing a partial language, how to compute a (finite) marked p/t-net with minimal set of runs, such that each specified LPO is a run of the net. Finally, we presented two effective methods to test, whether the computed net has more runs than specified or not. This decides the synthesis problem, since the synthesis problem has a solution if and only if the computed net does not have more runs than specified.

The restriction to finite partial languages need not be a problem in practise. An application field of synthesis methods is the discovery of workflow processes (given as Petri nets) from finite event logs [16]. A finite event log is a finite set of finite sequences of observed actions, thus a special type of a finite partial language (without concurrency). Currently there is research effort to deduce information about concurrency from event logs. Another natural application field is the specification of system behavior via message sequence charts, which are special types of LPOs. In both cases, the synthesized net can be used for analysis purposes.

The next steps of research are the implementation of Algorithm 1 into VipTool [5] extended by different versions of the equality test, evaluation of its performance and examination of the special instances of polyhedral cones used in the algorithm in view of a better upper bound for the number of basis solutions. We also work on a generalization of the presented results to infinite partial languages which allow a finite representation (for example a term-based representation).

References

[1] E. Badouel and P. Darondeau. On the synthesis of general petri nets. Technical Report 3025, Inria, 1996.

[2] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Trans. of Informations and Systems*, E80-D(3):315–325, 1997.

[3] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Hardware and petri nets: Application to asynchronous circuit design. In M. Nielsen; D. Simpson, editor, *ICATPN*, volume 1825 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[4] P. Darondeau. Deriving unbounded petri nets from formal languages. In D. Sangiorgi; R. de Simone, editor, *CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 533–548. Springer, 1998.

[5] J. Desel, G. Juhás, and R. Lorenz. Viptool-homepage., 2003. http://www.informatik.ku-eichstaett.de/projekte/vip/.

[6] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. part i: Basic notions and the representation problem. *Acta Inf.*, 27(4):315–342, 1989.

[7] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. part ii: State spaces of concurrent systems. *Acta Inf.*, 27(4):343–368, 1989.

[8] J. Grabowski. On partial languages. *Fundamenta Informaticae*, 4(2):428–498, 1981.

[9] M. B. Josephs and D. P. Furey. A programming approach to the design of asynchronous logic blocks. In J. Cortadella, A. Yakovlev, and G. Rozenberg, editors, *Concurrency and Hardware Design*, volume 2549 of *Lecture Notes in Computer Science*, pages 34–60. Springer, 2002.

[10] G. Juhás, R. Lorenz, and J. Desel. Can i execute my scenario in your net?. In G. Ciardo and P. Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 289–308. Springer, 2005.

[11] A. Kiehn. On the interrelation between synchronized and non-synchronized behaviour of petri nets. *Elektronische Informationsverarbeitung und Kybernetik*, 24(1/2):3–18, 1988.

[12] R. Lorenz and G. Juhás. Towards synthesis of petri nets from scenarios. In S. Donatelli and P. S. Thiagarajan, editors, *ICATPN*, volume 4024 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2006.

[13] H. Minkowski. *Geometrie der Zahlen*. Teubner, 1896.

[14] T. Motzkin. *Beiträge zur Theorie der linearen Ungleichungen*. PhD thesis, Jerusalem, 1936.

[15] V. Pratt. Modelling concurrency with partial orders. *Int. Journal of Parallel Programming*, 15:33–71, 1986.

[16] W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[17] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets.*, volume 625 of *Lecture Notes in Computer Science*. Springer, 1992.