# Synthesis of Petri nets from infinite partial languages

**Robin Bergenthum, Jorg Desel, Robert Lorenz, Sebastian Mauser**

**Angaben zur Veröffentlichung / Publication details:**

Bergenthum, Robin, Jorg Desel, Robert Lorenz, and Sebastian Mauser. 2008. "Synthesis of Petri nets from infinite partial languages." In *2008 8th International Conference on Application of Concurrency to System Design, 23-27 June 2008, Xian, China*, 170–79. Piscataway, NJ: IEEE. https://doi.org/10.1109/acsd.2008.4574609.

# Synthesis of Petri Nets from Infinite Partial Languages

Robin Bergenthum, Jörg Desel, Robert Lorenz and 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 unlabeled place/transition Petri net (p/t-net) from a possibly infinite partial language, which is given by a term over a finite set of labeled partial orders using operators for union, iteration, parallel composition and sequential composition. The synthesis algorithm is based on the theory of regions for partial languages presented in [18] and produces a p/t-net having minimal net behavior including the given partial language. The algorithm uses linear programming techniques that were already successfully applied in [17] for the synthesis of p/t-nets from finite partial languages.*

## 1 Introduction

Synthesis of Petri nets from behavioural 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, 11], and recently also in workflow design [21]. 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] and place/transition nets (p/t-nets) [1]. Synthesis can start with a transition system representing the sequential behaviour of a system as well as with a step transition system which additionally represents steps of concurrent events [1]. Synthesis can also be based on a language, i.e., on a set of occurrence sequences or step sequences [4, 1].

Recently, we solved the synthesis problem for p/t-nets with behaviour given in terms of a finite partial language, i.e., as a finite set of labelled partial orders (LPOs) [17]. LPOs are also known as partial words [8] or pomsets [19]. In contrast to previous work on the synthesis problem, we considered partial order behaviour of Petri nets, truly representing the concurrency of events. Partial orders are often considered the most appropriate representation of behaviour of concurrent systems modelled by Petri nets.

Based on our previous work, this paper tackles synthesis of Petri nets from infinite partial languages. More precisely, we introduce terms built from LPOs and composition operators including iteration. The semantics of an iterated finite LPO is an infinite set of LPOs. Moreover, we consider operators for sequential and parallel composition as well as a union operator. Given a term constructed this way from a finite set of LPOs, we show in this paper how to synthesize a finite p/t-net from this term such that the behaviour of the net coincides with the set of LPOs represented by the term – if such a net exists. The synthesis approach is based on the so called theory of regions. Each transition of the synthesized net is given by a label appearing in the term, and each place of the net is given by a region. The synthesized p/t-net has minimal net behaviour including the behaviour specified by the given term.

In contrast to [17], in this paper we only give the construction of the p/t-net, but not an algorithm to decide if the behaviour of the synthesized net coincides with the specified behaviour. That means we do not characterize partial languages generated by (unlabeled) p/t-nets. This is out of scope of this paper and a topic of further research.

We emphasize at this point that we aim at the synthesis of *unlabeled* Petri nets (i.e. Petri nets with unique transition names) and concentrate on the algorithmic solution. In contrast, in [14] partial languages generated by safe labeled p/t-nets are characterized. In [15, 16] partial languages which can be generated from singletons via operators for union, iteration, parallel composition and sequential composition (so called *series-rational sp-languages*) are characterized through so called *branching automata*, which can be interpreted as a restricted class of labeled p/t-nets. Other papers

*This paper was supported by the *German Research Council*, project *SYNOPS*.

consider unlabeled nets, but do not consider algorithmic aspects, such as [10] (characterizing the branching behaviour of p/t-nets without auto-concurrency by event structures) and [9] (proposing a trace semantics for p/t-nets).

The remainder of the paper is organized as follows: We start with a brief introduction to the behavioural model considered in this paper: We define the so called partial language of runs of a p/t-net in Section 2. In Section 3 the term based representation of infinite partial languages is introduced, generalizing regular expressions of sequential languages in two ways: a single partial word generalizes a sequential word, and we have a parallel composition operator. The latter only makes sense for partial words which can express independent, parallel execution of events. In Section 4 we first recall definitions and main results from [18] and [17] on the theory of regions for partial languages. Then we introduce regions of terms. Finally, the last Section 5 shows that, although the set of regions of a term is infinite in general, finitely many regions suffice for our construction, yielding a finite Petri net. This finite set is effectively constructed using concepts of linear programming.

## 2 The Partial Language of Runs of a P/t-net

In this section we introduce the behavioural model considered in this paper. By $\mathbb{N}$ we denote the *nonnegative integers*. $\mathbb{N}^+$ denotes the positive integers. 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 $f : A \to \mathbb{N}$. Given a binary relation $R \subseteq A \times A$, we write $aRb$ to denote $(a, b) \in R$. A *directed graph* is a pair $(V, \to)$, where $V$ is a finite *set of nodes* and $\to \subseteq V \times V$ is called the *set of arcs*. A *partial order* is a directed graph $\mathrm{po} = (V, <)$, where $< \subseteq V \times V$ is irreflexive and transitive.

**Definition 1** (Labelled partial order)**.** *A* labelled partial order *(LPO) is a triple* $\mathrm{lpo} = (V, <, l)$*, where* $(V, <)$ *is a partial order and* $l : V \to T$ *is a* labelling function *with* set of labels $T$.

In our context, a node $v$ of an LPO $(V, <, l)$ is called *event*, representing an occurrence of $l(v)$. Two nodes $v, v' \in V$ are called *independent* if $v \not< v'$ and $v' \not< v$. Notice that by this definition, independence is reflexive. 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 $C \subseteq V$ satisfying $\forall x, y \in C : x \,\mathrm{co}\, y$. A *cut* is a maximal co-set (w.r.t. set inclusion). 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 $v \,\mathrm{co}\, C$, if $v \,\mathrm{co}\, s$ for all elements $s \in C$. A partial order $(V', <')$ is a *prefix* of a partial order $(V, <)$ if $V' \subseteq V$, $<' = < |_{V' \times V'}$ and $(v' \in V' \wedge v < v') \Longrightarrow (v \in V')$. Given two partial orders $\mathrm{po}_1 = (V, <_1)$ and $\mathrm{po}_2 = (V, <_2)$, we say that $\mathrm{po}_2$ *is a sequentialization of* $\mathrm{po}_1$ if $<_1 \subseteq <_2$.

We use the notations defined for partial orders also for LPOs. If $T$ is the set of labels of $\mathrm{lpo} = (V, <, l)$ then for a set $V' \subseteq V$, we define the multi-set $|V'|_l \subseteq \mathbb{N}^T$ by $|V'|_l(t) = |\{v \in V' \mid l(v) = t\}|$. We consider LPOs only up to isomorphism. As usual, 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 each $v \in V$, and $v < w \Longleftrightarrow \psi(v) <' \psi(w)$ for each $v, w \in V$. By $[\mathrm{lpo}]$ we denote the set of all LPOs isomorphic to $\mathrm{lpo}$.

**Definition 2** (Partial language)**.** *Let $T$ be a set. A set* $\mathcal{L} \subseteq \{[\mathrm{lpo}] \mid \mathrm{lpo} = (V, <, l)$ *is an LPO,* $l(V) \subseteq T\}$ *with* $\bigcup_{[(V,<,l)] \in \mathcal{L}} l(V) = T$ *is called* partial language over $T$.

A partial language is given by a set of concrete LPOs $L$ representing $\mathcal{L}$ in the sense that $[\mathrm{lpo}] \in \mathcal{L} \Longleftrightarrow \exists \mathrm{lpo}' \in L : [\mathrm{lpo}] = [\mathrm{lpo}']$.

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 3** (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 $m : P \to \mathbb{N}$, 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*. The occurrence rule of p/t-nets is defined as usual [20]. The non-sequential semantics of a p/t-net can be given by *enabled LPOs*, also called *runs*. An LPO is enabled in a net if the events of the LPO can occur in the net respecting the concurrency relation of the LPO [22].

**Definition 4** (Enabledness)**.** *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 w.r.t. $(N, m_0)$ *if for every cut* $C$ *of* $\mathrm{lpo}$ *and every* $p \in P$ *there holds* $m_0(p) + \sum_{v \in V \wedge v < C}(W(l(v), p) - W(p, l(v))) \geq \sum_{v \in C} W(p, l(v))$*. Its* occurrence *leads to the marking* $m'$ *given by* $m'(p) = m_0(p) + \sum_{v \in V}(W(l(v), p) - W(p, l(v)))$ *for each* $p \in P$.

**Definition 5** (Partial language of runs)**.** *The set of all isomorphism classes of LPOs enabled w.r.t. a given marked p/t-net* $(N, m_0)$ *is denoted by* $\mathcal{L}(N, m_0)$*.* $\mathcal{L}(N, m_0)$ *is called the* partial language of runs *of* $(N, m_0)$.

Given a partial language $\mathcal{L}$, we are interested in algorithms to calculate a marked p/t-net $(N, m_0)$ with $\mathcal{L}(N, m_0) = \mathcal{L}$, if such a net exists. Observe that $\mathcal{L}(N, m_0)$ 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 $\mathcal{L}(N, m_0)$ is finite by definition. Therefore, when specifying the behaviour of a net by a partial language, this partial language must necessarily be sequentialization and prefix closed, and it must have a finite set of labels.

## 3 Term Based Finite Representation of Infinite Partial Languages

When specifying a partial language as the input for a synthesis algorithm, this specification has to be finite. In [17] we developed an algorithm to solve the synthesis problem for finite partial languages. We consider infinite partial languages in this paper. Consequently, we finitely represent infinite partial languages. More precisely we consider term-based finite representations of infinite partial languages. This approach was already successfully applied for the synthesis of nets from languages of occurrence sequences given by regular expressions over a finite alphabet of transitions [4]. In this paper, the alphabet is a finite set of LPOs. The considered terms extend regular expressions by a parallel composition operator representing concurrency. Thus we consider a class of partial languages specified by terms over a given finite set of LPOs $\mathcal{A}$, where terms are constructed by iteration, parallel and sequential composition and union. For $A \in \mathcal{A}$ we write $A = (V_A, <_A, l_A)$, and we denote by $\lambda = (\emptyset, \emptyset, \emptyset)$ the empty LPO.

**Definition 6** (LPO-term)**.** *The set of* LPO-terms *over a finite set of LPOs $\mathcal{A}$ is inductively defined as follows: The characters $A \in \mathcal{A}$ and $\lambda$ are LPO-terms. Let $\alpha_1$ and $\alpha_2$ be LPO-terms. Then $\alpha = \alpha_1;\alpha_2$ (sequential composition), $\alpha = \alpha_1 + \alpha_2$ (union), $\alpha = (\alpha_1)^*$ (iteration) and $\alpha = \alpha_1 \parallel \alpha_2$ (parallel composition) are LPO-terms.*

If each LPO in $\mathcal{A}$ is a singleton, an LPO-term defines a so called *series rational sp-language* [15, 16]. In a similar way, we assign to an arbitrary LPO-term $\alpha$ a possibly infinite set of LPOs $L(\alpha)$ representing a partial language. Given an LPO-term $\alpha$, we first inductively define a set of LPOs $K(\alpha)$ represented by $\alpha$. The set $L(\alpha)$ is the prefix and sequentialization closure of $K(\alpha)$. To define $K(\alpha)$, we define the sequential composition of LPOs $A, B \in \mathcal{A}$ by $AB = (V_A \cup V_B, <_A \cup <_B \cup (V_A \times V_B), l_A \cup l_B)$, the parallel composition of LPOs $A, B \in \mathcal{A}$ by $A \parallel B = (V_A \cup V_B, <_A \cup <_B, l_A \cup l_B)$, and denote $A^0 = \lambda$ and $A^n = A^{n-1}A$ for $n \in \mathbb{N}^+$ (we can assume that $A, B$ have disjoint sets of nodes).

**Definition 7** (Partial language of an LPO-term)**.** *We set $K(\lambda) = \{\lambda\}$ and $K(A) = \{A\}$. We further define inductively for LPO-terms $\alpha_1$ and $\alpha_2$:*
$K(\alpha_1 + \alpha_2) = K(\alpha_1) \cup K(\alpha_2)$
$K(\alpha_1;\alpha_2) = \{A_1A_2 \mid A_1 \in K(\alpha_1),\ A_2 \in K(\alpha_2)\}$
$K((\alpha_1)^*) = \{A_1 \ldots A_n \mid A_1, \ldots, A_n \in K(\alpha_1)\} \cup \{\lambda\}$
$K(\alpha_1 \parallel \alpha_2) = \{A_1 \parallel A_2 \mid A_1 \in K(\alpha_1),\ A_2 \in K(\alpha_2)\}$
*$L(\alpha)$ is the set of sequentializations of prefixes of LPOs in $K(\alpha)$. $\mathcal{L}(\alpha) = \{[\mathrm{lpo}] \mid \mathrm{lpo} \in L(\alpha)\}$ is the partial language of $\alpha$.*

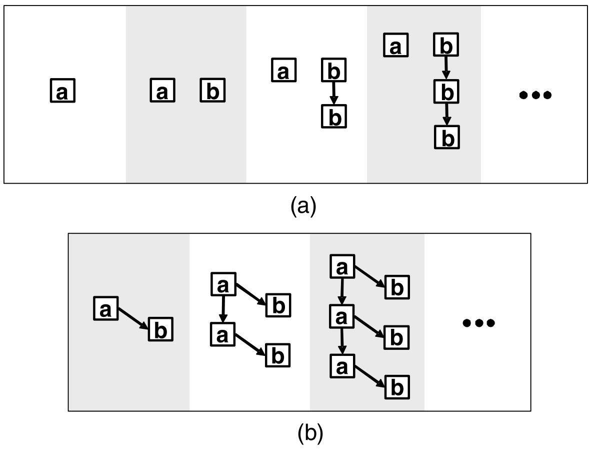

**Figure 1. Representations of partial languages.**

An example for a partial languages of an LPO-term is depicted in Figure 1, part (a), showing the set of LPOs $K(a \parallel b^\star)$. The set of LPOs from part (b) cannot be generated by an LPO-term (this is also the case if we consider its prefix-closure), because by sequential composition and iteration it is not possible to append an LPO only to a part of another LPO. Note that this set of LPOs could be defined through a recursive expression of the from $A = a(A \parallel b) + \lambda$. Synthesis from such expressions is a topic of further research. It is also possible to consider further composition operators (see [19] for an overview). In [12] an operator for synchronous composition of single actions is used, leading to terms which cannot longer be represented by (sets of) LPOs but by causal structures extending LPOs.

Altogether, the partial languages of LPO-terms (over finite sets of LPOs) form a certain class of infinite partial languages. Note that not each partial language of runs of a p/t-net can be described through LPO-terms.[1] It is easy to see that also not each partial language of runs of an elementary net or even of a marked graph can be described through LPO-terms.[2]

Note that, for simplicity of figures, in all examples we only consider LPO-terms constructed from singletons.

[1]The partial language of runs of the p/t-net having the two places $p$ and $q$ defined by $W(p,a) = W(a,p) = W(a,q) = W(q,b) = 1$ equals the partial language depicted in part (b) of Figure 1.

[2]Consider the marked graph having transitions $a, b, c, d$ and four places $p, q, r, s$ defined by $W(p,a) = W(a,q) = W(q,b) = W(b,p) = W(a,r) = W(r,c) = W(b,s) = W(s,d) = 1$.

## 4 Regions of LPO-terms

The synthesis problem tackled in this paper is as follows:
**Given:** An LPO-term $\alpha$.
**Searched:** A marked p/t-net $(N, m_0)$ with $\mathcal{L}(N, m_0) = \mathcal{L}(\alpha)$ if such $(N, m_0)$ exists.
We use the so called theory of regions to solve the synthesis problem. Like the synthesis algorithm for finite partial languages in [17], the synthesis algorithm in this paper is based on the notion of regions of partial languages introduced in [18]. Transitions of the synthesized net are given by the labels of the partial language and places are given by regions. In the case of infinite partial languages, a region according to [18] is a function with an infinite number of variables that has to fulfill an infinite number of constraints. Such regions are not computable. The aim of this section is to define computable regions of an LPO-term $\alpha$ which define the same places as the regions of $\mathcal{L}(\alpha)$ from [18].

We first recall the general ideas of region based synthesis. The basic approach is the construction of a marked p/t-net from a given partial language $\mathcal{L}$ according to the following strategy: The set of transitions of the synthesized net is the finite set of labels of $\mathcal{L}$. Clearly, each LPO specified in $\mathcal{L}$ is a run of the marked p/t-net consisting only of these transitions (with empty set of places), because there are no causal dependencies between the transitions. Therefore, this net in general has many runs not specified in $\mathcal{L}$. Thus, one restricts the behaviour of this net by creating causal dependencies between the transitions through addition of places. Places are defined by their initial marking and the weights on the arcs connecting them to transitions. Two kinds of places can be distinguished. In the case that there is an LPO specified in $\mathcal{L}$ which is no run of the net which has only the one considered place, this place restricts the behaviour too much. Such places are *non-feasible (w.r.t. $\mathcal{L}$)*. In the other case, the considered place is *feasible (w.r.t. $\mathcal{L}$)*. Every feasible place is added to the net to be constructed.

**Definition 8** (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 $\mathcal{L} \subseteq \mathcal{L}(N, m_p)$, otherwise* non-feasible (w.r.t. $\mathcal{L}$).

Examples of a feasible place and a non-feasible place are depicted in Figure 2, showing the set of LPOs $K(b+(a; (a \parallel b)^\star))$ in part (a): Using Definition 4 one can easily verify that all LPOs in $L(b + (a; (a \parallel b)^\star))$ are enabled in the one-place net depicted in part (b). The third LPO of part (a) is not enabled in the one-place net depicted in part (c).

**Definition 9** (Saturated feasible p/t-net)**.** *Let $\mathcal{L}$ be a partial language over the finite set of labels $T$. The marked p/t-net*

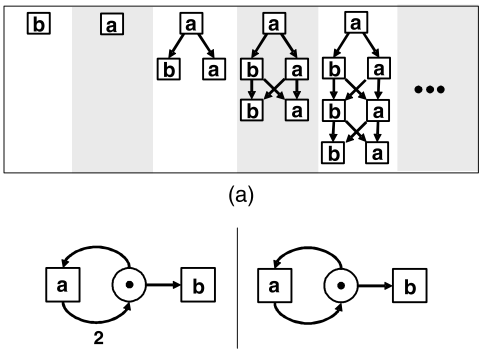

**Figure 2. (a)** $K(b + (a; (a \parallel b)^\star))$**, (b) feasible place, (c) non-feasible place.**

*$(N, m_0)$, $N = (P, T, F, W)$, such that $P$ is the set of all places feasible w.r.t. $\mathcal{L}$ is called* saturated feasible (w.r.t. $\mathcal{L}$)

**Theorem 10** ([18])**.** *Let $(N, m_0)$ be the saturated feasible p/t-net w.r.t. a partial language $\mathcal{L}$. Then $\mathcal{L} \subseteq \mathcal{L}(N, m_0)$ and $\mathcal{L}(N, m_0)$ is minimal with this property.*

Altogether, given an LPO-term $\alpha$, the saturated feasible net $(N, m_0)$ w.r.t. $\mathcal{L}(\alpha)$ solves the formulated synthesis problem, i.e. $\mathcal{L}(N, m_0) = \mathcal{L}(\alpha)$, or there is no such net. A problem that we solve in Section 5 is that there are infinitely many feasible places.

By *regions* of partial languages it is possible to define the set of all feasible places on the level of the partial language. Given a partial language $\mathcal{L}$ over $T$ represented by a set of LPOs $L$, the idea of defining regions of $\mathcal{L}$ was developed in [18]: If two events $x$ and $y$ satisfy $x < y$ in an LPO $\mathrm{lpo} = (V, <, l) \in L$, this specifies that the corresponding transitions $l(x)$ and $l(y)$ may be causally dependent. Such a causal dependency arises exactly if the occurrence of the transition $l(x)$ produces one or more 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*. We extend each LPO $\mathrm{lpo} \in L$ by an initial and a final event, representing transitions producing the initial marking and consuming the final marking (after the occurrence of lpo).

**Definition 11** ($\star$-extension)**.** *A* $\star$-extension *of* $\mathrm{lpo} = (V, <, l) \in L$ *is an LPO* $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ *satisfying: There is an* initial node $v_{init} \in V^\star$ *smaller than all other nodes and a* final node $v_{final} \in V^\star$ *bigger than all other nodes, both with new labels, and there holds $V = V^\star \setminus \{v_{init}, v_{final}\}$.*

*For each* $\mathrm{lpo} \in L$*, let* $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ *be a $\star$-extension of* lpo *such that $\star$-extensions have disjoint node sets and all initial and final nodes have different labels.*

*Then the set* $L^\star = \{\text{lpo}^\star \mid \text{lpo} \in L\}$ *is called* $\star$-extension of $L$. *We denote* $E_L^\star = \bigcup_{(V,<,l)\in L^\star} <$.

Assume we have fixed a $\star$-extension of $L$. According to the above explanation, we define a place $p_r$ by assigning for each LPO $\text{lpo} = (V,<,l) \in L$ a natural number $r(x,y)$ to each edge $(x,y)$ of the $\star$-extension of lpo through a function $r : E_L^\star \to \mathbb{N}$. The sum of the natural numbers assigned to ingoing edges $(x,y)$ of a node $y \in V^\star$ is denoted by $In(y,r) = \sum_{x<^\star y} r(x,y)$. We call $In(y,r)$ the *intoken flow* of $y$. If $y$ is no initial or final node, the intoken flow of $y$ is interpreted as the weight of the arc connecting the new place $p_r$ with the transition $l(y)$, i.e. we define $W(p_r, l(y)) = In(y,r)$. The sum of the natural numbers assigned to outgoing edges $(x,y)$ of a node $x \in V^\star$ is denoted by $Out(x,r) = \sum_{x<^\star y} r(x,y)$. We call $Out(x,r)$ the *outtoken flow* of $x$. If $x$ is no initial or final node, the outtoken flow of $x$ is interpreted as the weight of the arc connecting the transition $l(x)$ with the new place $p_r$, i.e. we define $W(l(x), p_r) = Out(x,r)$. If $z$ is the initial node of the $\star$-extension of lpo, then the outtoken flow of $z$ is interpreted as the initial marking of the new place $p_r$, i.e. we define $m_0(p_r) = Out(z,r)$. We also denote $Init(\text{lpo}, r) = Out(z,r)$ and call $Init(\text{lpo}, r)$the *initial token flow* of lpo. The value $r(x,y)$ is called the *token flow* between $x$ and $y$. Since equally labeled nodes formalize occurrences of the same transition, $p_r$ is well-defined only if equally labeled nodes have equal intoken flow and equal outtoken flow. Since the initial token flow of all LPOs formalizes the initial marking, $p_r$ is well-defined only if all LPOs have equal initial token flow. In general we say that a function $r : E_L^\star \to \mathbb{N}$ *fulfills the property* $(*)$ *w.r.t.* $L$ if for all LPOs $\text{lpo} = (V,<,l), \text{lpo}' = (V',<',l') \in L$ and for all $v \in V, v' \in V'$ holds

$(*)$ $Init(\text{lpo}, r) = Init(\text{lpo}', r) \wedge (l(v) = l'(v') \implies In(v,r) = In(v',r) \wedge Out(v,r) = Out(v',r))$.

Every function $r$ fulfilling $(*)$ for a set of LPOs $L$ defines a place $p_r$ as shown above. The place $p_r$ is said to be *corresponding* to $r$. If $z$ is the final node of the $\star$-extension of $\text{lpo} \in L$, we denote $Final(\text{lpo}, r) = In(z,r)$. Moreover, we write $Init(p_r) = Init(\text{lpo}, r)$ and $Final(\text{lpo}, p_r) = Final(\text{lpo}, r)$. For a function $r$ on the edges of a $\star$-extension of a single LPO lpo we say that it fulfills $(*)$ if $r$ fulfills $(*)$ w.r.t. $\{\text{lpo}\}$.

**Definition 12** (Region). *A* region *of a partial language* $\mathcal{L}$ *is a function* $r : E_L^\star \to \mathbb{N}$ *fulfilling* $(*)$.

The main result of [18] is that the set of places corresponding to regions of a partial language equals the set of feasible places w.r.t. this partial language.[3] Thus the saturated feasible net can be given by the set of places corresponding to regions.

[3] In [18] it is assumed that the set of LPOs $L$ representing $\mathcal{L}$ fulfills some technical requirements. Since such representation is always possible we omit a detailed presentation here.

**Theorem 13.** *Let* $\mathcal{L}$ *be a partial language. Then each place corresponding to a region of* $\mathcal{L}$ *is feasible w.r.t.* $\mathcal{L}$ *and each place feasible w.r.t.* $\mathcal{L}$ *corresponds to a region of* $\mathcal{L}$.

In this paper we deal with partial languages $\mathcal{L}(\alpha)$ given by an LPO-term $\alpha$. According to property $(*)$ of regions it is enough to define regions of a partial language $\mathcal{L}(\alpha)$ on the edges of LPOs in $K(\alpha)^\star$ because such regions can be extended to edges of LPOs in $L(\alpha)^\star \setminus K(\alpha)^\star$: If $\text{lpo} = (V,<,l)$ is a prefix of $\text{lpo}' = (V',<',l')$ and a region is defined on $(<')^\star$ then merge the nodes of $(V')^\star \setminus V^\star$ to one node representing the maximal node of $\text{lpo}^\star$ thus defining a region on $<^\star$. If $\text{lpo} = (V,<,l)$ is a sequentialization of $\text{lpo}' = (V',<',l')$ and a region is defined on $(<')^\star$ then assign the value 0 to edges in $<^\star \setminus (<')^\star$.

An example region of the partial language $\mathcal{L}(\alpha)$ introduced in Figure 2 is depicted in Figure 3. The feasible place in Figure 2, part (b), corresponds to this region. The non-zero values of $r$ are assigned to the arcs of the LPOs in $K(\alpha)$. Initial and final nodes are not drawn. The non-zero values of $r$ assigned to edges starting from an initial node respectively ending in a final node are depicted with small ingoing resp. outgoing arrows. The intoken flow of $a$ and $b$ equals 1, the outtoken flow of $a$ equals 2, the outtoken flow of $b$ equals 0. The initial token flow equals 1.

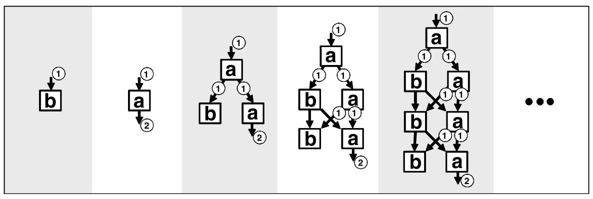

**Figure 3. A region of a partial language.**

We now describe a technique to represent the regions of a possibly infinite partial language $\mathcal{L}(\alpha)$, defined on the edges of $K(\alpha)$, through regions of a finite representation of $K(\alpha)$. By now $K(\alpha)$ may contain infinitely many LPOs, caused by the iteration operator. An LPO $A$ can occur arbitrary often consecutively in a certain marking if and only if it consumes in every place at most as many tokens as it produces in this place (then an occurrence of $A$ does not reduce the number of tokens in this place). Consequently, if $A$ can occur iterated in a certain marking $m$, then another LPO $B$ can occur after the occurrence of $A^n$ for each $n \in \mathbb{N}$ if and only if it can occur in $m$, since an occurrence of $A$ does not reduce the number of tokens in a place. This principle can be used to represent the infinite set $K(\alpha)$ by two finite sets of LPOs $R(\alpha)$ and $I(\alpha)$. We define regions by these two sets. This approach is similar to the ideas in [4] where the authors define regions by two finite sets representing a regular expression. Our sets differ from these sets because of two reasons. First, [4] deals with occurrence sequences of pure nets instead of LPOs of p/t-nets. Second, a region in [4] does not include the value of the initial marking of the corresponding place.

For arbitrary LPO-terms $\alpha$ we define inductively the finite representation set $R(\alpha)$ consisting of, roughly speaking, all LPOs in $K(\alpha)$ neglecting iterations. To ensure that all LPOs in $R(\alpha)$ are enabled w.r.t the place defined by the region, we require that regions satisfy $(*)$ w.r.t. $R(\alpha)$. It remains to ensure that certain LPOs can occur iterated. For this we define inductively the second finite iteration set $I(\alpha)$ of LPOs consisting of, roughly speaking, all LPOs associated to iterated subterms of $\alpha$. We require that the LPOs in $I(\alpha)$ produce at least as many tokens as they consume in the place to be defined by the region. This ensures that the place defined by the region is feasible w.r.t. $\mathcal{L}(\alpha)$.

**Definition 14** (Representation/Iteration set)**.** *The* representation set $R(\alpha)$ *and the* iteration set $I(\alpha)$ *of a partial language $\mathcal{L}(\alpha)$ are defined inductively for LPO-terms $\alpha_1$ and $\alpha_2$ as follows:*
$R(\lambda) = \{\lambda\}$, $I(\lambda) = \emptyset$,
$R(A) = \{A\}$, $I(A) = \emptyset$ *for* $A \in \mathcal{A}$,
$R(\alpha_1+\alpha_2) = R(\alpha_1) \cup R(\alpha_2)$, $I(\alpha_1+\alpha_2) = I(\alpha_1) \cup I(\alpha_2)$,
$R(\alpha_1;\alpha_2) = \{A_1A_2 \mid A_1 \in R(\alpha_1),\ A_2 \in R(\alpha_2)\}$,
$I(\alpha_1;\alpha_2) = I(\alpha_1) \cup I(\alpha_2)$,
$R((\alpha_1)^*) = R(\alpha_1) \cup \{\lambda\}$, $I((\alpha_1)^*) = I(\alpha_1) \cup R(\alpha_1)$,
$R(\alpha_1 \parallel \alpha_2) = \{A_1 \parallel A_2 \mid A_1 \in R(\alpha_1),\ A_2 \in R(\alpha_2)\}$,
$I(\alpha_1 \parallel \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$.
*We denote* $W_\alpha = \bigcup_{(V,<,l)\in R(\alpha)^\star} V$, $E_\alpha = \bigcup_{(V,<,l)\in R(\alpha)^\star} <$, $l_\alpha = \bigcup_{(V,<,l)\in R(\alpha)^\star} l$.

Figure 4, part (a), shows the representation set $R(b + (a;(a \parallel b)^\star))$ and the iteration set $I(b + (a;(a \parallel b)^\star))$ of the partial language $\mathcal{L}(b + (a;(a \parallel b)^\star))$ introduced in Figure 2, together with an annotated region (defined later).

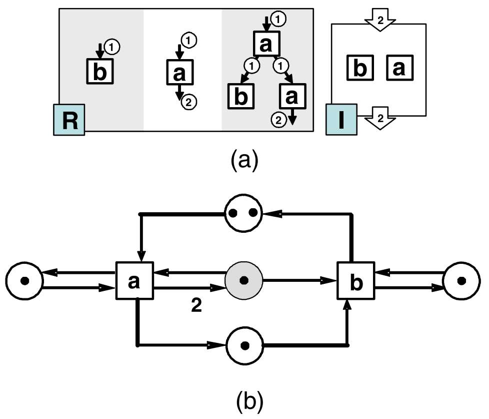

**Figure 4. (a) Region of representation and iteration set, (b) A part of the synthesized net.**

The requirement, that every LPO in $I(\alpha)$ produces at least as many tokens as it consumes, corresponds to the requirement, that the final token flow of each LPO in $I(\alpha)$ exceeds its initial token flow. Given an LPO $\mathrm{lpo} = (V,<,l)$ and some place $p$, the sum

$$\mathit{Prod}(\mathrm{lpo}, p) := \sum_{t \in l(V)} |V|_l(t)(W(t,p) - W(p,t))$$

equals the difference of the final and the initial token flow.

**Lemma 15.** *Let $(V^\star, <^\star, l^\star)$ be a $\star$-extension of* $\mathrm{lpo} = (V,<,l)$. *If $r : <^\star \to \mathbb{N}$ satisfies $(*)$ then there holds $\mathit{Prod}(\mathrm{lpo}, p_r) = \mathit{Final}(\mathrm{lpo}, p_r) - \mathit{Init}(p_r)$.*

*Proof.* It holds $\sum_{v\in V^\star} \mathit{Out}(v,r) = \sum_{e\in<^\star} r(e) = \sum_{v\in V^\star} \mathit{In}(v,r)$. Let $v_{init}$ be the initial node and $v_{final}$ be the final node of the $\star$-extension of lpo. Then $\sum_{v\in V} \mathit{Out}(v,r) + \mathit{Out}(v_{init},r) + \mathit{Out}(v_{final},r) = \sum_{v\in V} \mathit{In}(v,r) + \mathit{In}(v_{init},r) + \mathit{In}(v_{final},r)$. Since $\mathit{Out}(v_{final},r) = 0 = \mathit{In}(v_{init},r)$, we get $\mathit{In}(v_{final},r) - \mathit{Out}(v_{init},r) = \sum_{v\in V}(\mathit{Out}(v,r) - \mathit{In}(v,r)) = \sum_{t\in l(V)} |V|_l(t)(W(t,p_r) - W(p_r,t))$. □

We define a region of an LPO-term as a function on the edges of $R(\alpha)^\star$.

**Definition 16** (Region of an LPO-term)**.** *A* region $s$ *of an LPO-term $\alpha$ is a function $s : E_\alpha \to \mathbb{N}$ satisfying $(*)$ w.r.t. $R(\alpha)$, which additionally fulfills for all LPOs* $\mathrm{lpo} \in I(\alpha)$*:*

$$(**) \qquad \mathit{Prod}(\mathrm{lpo}, p_s) \geqslant 0.$$

Each region $s$ of an LPO-term defines a corresponding place $p_s$ in an analogous way as regions of partial languages. Figure 4, part (a), shows a region $s$ of the LPO-term $b + (a;(a \parallel b)^\star)$ (illustrated analogously as in Figure 3). It defines the feasible place $p_s$ shown grey in Figure 4, part (b). For the single LPO $\mathrm{lpo} \in I(\alpha)$ the value $\mathit{Final}(\mathrm{lpo}, p_s) = 2$ ($\mathit{Init}(p_s) = 2$) is attached to a big outgoing (ingoing) arc.

Finally, we will prove that the places defined by regions of $\alpha$ and the places defined by regions of $\mathcal{L}(\alpha)$ coincide. For this we need three lemmas. Two of these regard technical constructions. These two are arranged at the end of the section after the main theorem. The relationship between $K(\alpha)$ and $R(\alpha)$ gets clear in the following lemma.

**Lemma 17.** *Given an LPO-term $\alpha$ and a region $s$ of $\alpha$, for each* $\mathrm{lpo} \in K(\alpha)$ *there is* $\mathrm{lpo}^R \in R(\alpha)$ *such that $\mathit{Prod}(\mathrm{lpo}, p_s) \geq \mathit{Prod}(\mathrm{lpo}^R, p_s)$.*

*Proof by induction.* Given $A \in \mathcal{A}$ it holds $K(A) = \{A\} = R(A)$, i.e. we can set $A^R = A$. Assume the statement holds for $\alpha_1$ and $\alpha_2$:

*ad* $(+)$: Given $\mathrm{lpo} \in K(\alpha_1 + \alpha_2)$, then $\mathrm{lpo} \in K(\alpha_1)$ or $\mathrm{lpo} \in K(\alpha_2)$. Let $\mathrm{lpo} \in K(\alpha_1)$. By assumption there is $\mathrm{lpo}^R \in R(\alpha_1) \subseteq R(\alpha_1 + \alpha_2)$ such that $\mathit{Prod}(\mathrm{lpo}, p_s) \geq \mathit{Prod}(\mathrm{lpo}^R, p_s)$.

*ad* $(;)$: Given $\text{lpo} \in K(\alpha_1;\alpha_2)$, there is $\text{lpo}_1 \in K(\alpha_1)$ and $\text{lpo}_2 \in K(\alpha_2)$ such that $\text{lpo}_1;\text{lpo}_2 = \text{lpo}$. By assumption there are $\text{lpo}_1^R \in R(\alpha_1)$, $\text{lpo}_2^R \in R(\alpha_2)$ such that $\mathit{Prod}(\text{lpo}_i^R, p_s) \leq \mathit{Prod}(\text{lpo}_i, p_s)$, $i = 1, 2$. It holds $\text{lpo}_1^R;\text{lpo}_2^R \in R(\alpha_1;\alpha_2)$ with $\mathit{Prod}(\text{lpo}_1^R;\text{lpo}_2^R, p_s) = \mathit{Prod}(\text{lpo}_1^R, p_s) + \mathit{Prod}(\text{lpo}_2^R, p_s) \leq \mathit{Prod}(\text{lpo}_1, p_s) + \mathit{Prod}(\text{lpo}_2, p_s) = \mathit{Prod}(\text{lpo}_1;\text{lpo}_2, p_s)$.

*ad* $(\|)$: Given $\text{lpo} \in K(\alpha_1 \parallel \alpha_2)$, there is $\text{lpo}_1 \in K(\alpha_1)$ and $\text{lpo}_2 \in K(\alpha_2)$ such that $\text{lpo}_1 \parallel \text{lpo}_2 = \text{lpo}$. By assumption there is $\text{lpo}_1^R \in R(\alpha_1)$, $\text{lpo}_2^R \in R(\alpha_2)$ such that $\mathit{Prod}(\text{lpo}_i^R, p_s) \leq \mathit{Prod}(\text{lpo}_i, p_s)$, $i = 1, 2$. It holds $\text{lpo}_1^R \parallel \text{lpo}_2^R \in R(\alpha_1 \parallel \alpha_2)$ with $\mathit{Prod}(\text{lpo}_1^R \parallel \text{lpo}_2^R, r) = \mathit{Prod}(\text{lpo}_1^R, p_s) + \mathit{Prod}(\text{lpo}_2^R, p_s) \leq \mathit{Prod}(\text{lpo}_1, p_s) + \mathit{Prod}(\text{lpo}_2, p_s) = \mathit{Prod}(\text{lpo}_1 \parallel \text{lpo}_2, p_s)$.

*ad* $(*)$: Given $\text{lpo} \in K(\alpha_1^*)$, there are $\text{lpo}_1, \ldots, \text{lpo}_n \in K(\alpha_1)$ such that $\text{lpo}_1; \ldots ; \text{lpo}_n = \text{lpo}$. By assumption and (**), for each $\text{lpo}_i$ ($i \in \{1, \ldots, n\}$) there is $\text{lpo}_i^R \in R(\alpha) \subseteq I(\alpha^*)$ with $\mathit{Prod}(\text{lpo}_i, p_s) \geq \mathit{Prod}(\text{lpo}_i^R, p_s) \geq 0$. It holds $\text{lpo}_1^R \in R(\alpha_1^*)$ with $\mathit{Prod}(\text{lpo}_1^R, p_s) \leq \mathit{Prod}(\text{lpo}_1, p_s) + \ldots + \mathit{Prod}(\text{lpo}_n, p_s) = \mathit{Prod}(\text{lpo}_1; \ldots ; \text{lpo}_n, p_s)$. $\square$

The following theorem proves the correspondence between regions of LPO-terms and regions of partial languages corresponding to LPO-terms.

**Theorem 18.** *Let $\alpha$ be an LPO-term. It holds:*
*(i) Let $s$ be a region of $\alpha$ with corresponding place $p$. Then there is a region $r$ of $\mathcal{L}(\alpha)$ with corresponding place $p$.*
*(ii) Let $r$ be a region of $\mathcal{L}(\alpha)$ with corresponding place $p$. Then there is a region $s$ of $\alpha$ with corresponding place $p$.*

*Proof by induction.* As described, it is enough to consider $K(\alpha)$ instead of $L(\alpha)$.

*ad (i)*: Given $A \in \mathcal{A}$ we have $R(A) = \{A\} = K(A)$. That means we can chose $r = s$. Assume the statement holds for $\alpha$, $\alpha_1$ and $\alpha_2$:

*ad* $(;)$: Let $s$ be a region of $\alpha_1;\alpha_2$ with corresponding place $p$. First we construct a region $s_1$ of $\alpha_1$ and a region $s_2$ of $\alpha_2$ from $s$. For the construction of $s_1$ fix $\text{lpo}_2 \in R(\alpha_2)$. For each $\text{lpo}_1 \in R(\alpha_1)$ there holds $\text{lpo}_1;\text{lpo}_2 \in R(\alpha_1;\alpha_2)$. In Lemma 19 we describe how to construct a function $s|_{lpo_1}$ which satisfies $(*)$ on $\{\text{lpo}_1\}$ and corresponds to the place $p$. On every $\text{lpo}_1 \in R(\alpha_1)$ define $s_1 = s|_{lpo_1}$. Since $I(\alpha_1) \subseteq I(\alpha_1;\alpha_2)$, $s_1$ satisfies $(**)$, i.e. is a region of $\alpha_1$. For the construction of $s_2$ fix $\text{lpo}_1^{min} \in R(\alpha_1)$ such that for all $\text{lpo} \in R(\alpha_1)$ there holds $\mathit{Prod}(\text{lpo}_1^{min}, p) \leq \mathit{Prod}(\text{lpo}, p)$. For each $\text{lpo}_2 \in R(\alpha_2)$ there holds $\text{lpo}_1^{min};\text{lpo}_2 \in R(\alpha_1;\alpha_2)$. In Lemma 19 we describe how to construct a function $s|_{lpo_2}$ which satisfies $(*)$ on $\{\text{lpo}_2\}$ and corresponds to the place $p_2$ which differs from $p$ only in its initial marking. It holds $\mathit{Init}(p_2) = \mathit{Init}(p) + \mathit{Prod}(\text{lpo}_1^{min}, p)$. Define $s_2 = s|_{lpo_2}$ on every $\text{lpo} \in R(\alpha_2)$. Since $I(\alpha_2) \subseteq I(\alpha_1;\alpha_2)$, $s_2$ satisfies $(**)$, i.e. is a region of $\alpha_2$. By assumption there is a region $r_1$ of $K(\alpha_1)$ with corresponding place $p$ and a region $r_2$ of $K(\alpha_2)$ with corresponding place $p_2$. Given $\text{lpo} \in K(\alpha_1;\alpha_2)$ there is $\text{lpo}_1^K \in K(\alpha_1)$ and $\text{lpo}_2^K \in K(\alpha_2)$ such that $\text{lpo}_1^K;\text{lpo}_2^K = \text{lpo}$. Observe that $\mathit{Final}(\text{lpo}_1^K, p) = \mathit{Init}(p) + \mathit{Prod}(lpo_1^K, p) \geq \mathit{Init}(p) + \mathit{Prod}(\text{lpo}_{min}, p) = \mathit{Init}(p_2)$. This is the precondition for Lemma 20 and we are able to construct a function $r_{\text{lpo}}$ fulfilling $(*)$ and corresponding to $p$. Define $r = r|_{\text{lpo}}$ on each $lpo \in K(\alpha_1;\alpha_2)$.

*ad* $(+)$: Let s be a region of $\alpha_1 + \alpha_2$ with corresponding place $p$. Since $R(\alpha_1 + \alpha_2) = R(\alpha_1) \cup R(\alpha_2)$ and $I(\alpha_1 + \alpha_2) = I(\alpha_1) \cup I(\alpha_2)$ we directly get regions $s_1$ of $\alpha_1$ and $s_2$ of $\alpha_2$ both corresponding to $p$ just by restricting $s$ onto $R(\alpha_1)$ resp. $R(\alpha_2)$. By assumption there is a region $r_1$ of $K(\alpha_1)$ and a region $r_2$ of $K(\alpha_2)$ corresponding to $p$. Given $\text{lpo} \in K(\alpha_1 + \alpha_2)$, there holds $\text{lpo} \in K(\alpha_1)$ or $\text{lpo} \in K(\alpha_2)$. Therefore $r_1 \cup r_2$ is a region of $K(\alpha_1 + \alpha_2)$ with corresponding place $p$.

*ad* $(\|)$: Let $s$ be a region of $\alpha_1 \parallel \alpha_2$ with corresponding place $p$. First we construct a region $s_1$ of $\alpha_1$. Fix an $\text{lpo}_2 \in R(\alpha_2)$. For every $\text{lpo}_1 \in R(\alpha_1)$ there holds $\text{lpo}_1 \parallel \text{lpo}_2 \in R(\alpha_1 \parallel \alpha_2)$. To define $s_1$ on $\text{lpo}_1$ we start with $s$ on $(\text{lpo}_1 \parallel \text{lpo}_2)^\star$ and just delete all nodes of $lpo_2$ and adjacent edges. There results a function $s_1$ on a $\star$-extension of $\text{lpo}_1$ which fulfills $(*)$ and corresponds to a place $p'$, where $p'$ equals $p$ except for its initial marking. On every $\text{lpo}_1 \in R(\alpha_1)$ define $s_1 = s|_{lpo_1}$. Since $I(\alpha_1) \subseteq I(\alpha_1 \parallel \alpha_2)$, $s_1$ satisfies $(**)$, i.e. is a region of $\alpha_1$. Analogously we fix some $\text{lpo}_1 \in R(\alpha_1)$ and construct a region $s_2$ of $\alpha_2$ with corresponding place $p''$ equal to $p$ except for its initial marking. It holds $\mathit{Init}(p') + \mathit{Init}(p'') = \mathit{Init}(p)$. By assumption there is a region $r_1$ of $K(\alpha_1)$ with corresponding place defining $p'$ and a region $r_2$ of $K(\alpha_2)$ with corresponding place $p''$. Given $\text{lpo} \in K(\alpha_1 \parallel \alpha_2)$, there is $\text{lpo}_1^K \in K(\alpha_1)$ and $\text{lpo}_2^K \in K(\alpha_2)$ such that $\text{lpo}_1^K \parallel \text{lpo}_2^K \in K(\alpha_1 \parallel \alpha_2)$. To get a function on the edges of $(\text{lpo}_1^K \parallel \text{lpo}_2^K)^\star$ fulfilling $(*)$ take $(\text{lpo}_1)^\star$ and $(\text{lpo}_2)^\star$ with the annotated regions $r_1$ and $r_2$ and join the two initial and final nodes. The resulting function corresponds to $p$.

*ad* $(*)$: Let $s$ be a region of $\alpha^*$ with corresponding place $p$. $s$ directly defines a region $s_1$ of $\alpha$ with corresponding place $p$ since $R(\alpha) \subseteq R(\alpha^*)$ and $I(\alpha) \subseteq I(\alpha^*)$. By assumption there is a region $r_1$ of $K(\alpha)$ with corresponding place $p$. Given $\text{lpo} \in K(\alpha^*)$ there is $\text{lpo}_i \in K(\alpha)$, $i = 1, \ldots, n$, such that $\text{lpo}_1; \ldots ; \text{lpo}_n = \text{lpo}$. As in the case (;), we apply Lemma 20 to get a function $r_2$ on the edges of $(\text{lpo}_1;\text{lpo}_2)^\star$ satisfying $(*)$ with corresponding place $p$. For this we need to show that the preconditions of lemma 20 are fulfilled. For $\text{lpo}_1 \in K(\alpha)$ there is $\text{lpo}_1^R \in R(\alpha)$ with $\mathit{Prod}(\text{lpo}_1, p) \geq \mathit{Prod}(\text{lpo}_1^R, p)$ (Lemma 17). Because of $R(\alpha) \subseteq I(\alpha^*)$ it holds $\mathit{Prod}(\text{lpo}_1^R, p) \geq 0$. We get $\mathit{Prod}(\text{lpo}_1, p) \geq 0$ and it holds $\mathit{Init}(\text{lpo}_1, r_1) = \mathit{Init}(\text{lpo}_2, r_1)$ since $r_1$ is a region of $K(\alpha_1)$. Finally it holds $\mathit{Final}(\text{lpo}_1, p) = \mathit{Init}(p) + \mathit{Prod}(\text{lpo}_1, p) \geq$

$Init(p)$. Inductively we get a function $r_n$ on the edges of $lpo^\star$ fulfilling $(*)$ with corresponding place $p$.

*ad (ii)*: $r|_{E_\alpha}$ is a function on the edges of $R(\alpha)$ satisfying $(*)$ corresponding to $p$ since $R(\alpha) \subseteq K(\alpha)$. It remains to show $(**)$ for $s = r|_{E_\alpha}$. We assume that there is $\mathrm{lpo} \in I(\alpha)$ such that $Prod(\mathrm{lpo}, p) < 0$. From the definition of $I(\alpha)$ there are subterms $\beta$ and $\gamma$ of $\alpha$ with $\beta = \gamma^*$ and $\mathrm{lpo} \in R(\gamma) \subseteq I(\beta) \subseteq I(\alpha)$. It holds $(\mathrm{lpo})^n \in L(\beta)$ and $Prod((\mathrm{lpo})^n, p) = n \cdot Prod(\mathrm{lpo}, p) \leqslant -n$ for each $n \in \mathbb{N}$. That means $\beta$ satisfies the following property

$(+)$: For each $n$ there is an LPO $\mathrm{lpo}_n \in L(\beta)$ such that $Prod(lpo_n, p) \leqslant -n$.

It is easy to observe that if an LPO-term $\alpha_1$ satifies property $(+)$, then for arbitrary LPO-terms $\alpha_2$ also the LPO-terms $\alpha_1 + \alpha_2$, $\alpha_1 \parallel \alpha_2$, $\alpha_1;\alpha_2$, $\alpha_2;\alpha_1$ and $\alpha_1^*$ satisfy property $(+)$. Since $\beta$ is a subterm of $\alpha$ this implies that also $\alpha$ satisfies property $(+)$. Therefore there is an LPO $\mathrm{lpo}_N \in L(\alpha)$ such that $Prod(\mathrm{lpo}_N, p) < -Init(p)$. This gives $Final(\mathrm{lpo}_N, p) = Prod(\mathrm{lpo}_N, p) + Init(p) < 0$, a contradiction. □

**Lemma 19.** *Let $s$ be a function on the edges of $(lpo_1; lpo_2)^\star$ satisfying $(*)$ with corresponding place $p_s$. Then there is a function $s|_{lpo_1}$ on the edges of $lpo_1^\star$ satisfying $(*)$ with corresponding place $p_s$ and a function $s|_{lpo_2}$ on the edges of $lpo_2^\star$ satisfying $(*)$ with corresponding place $p'_s$ such that $W(p'_s, t) = W(p_s, t)$ and $W(t, p'_s) = W(t, p_s)$ for every $t \in T$ and $Init(p'_s) = Init(p_s) + Prod(lpo_1, p_s)$.*

*Proof.* Denote $\mathrm{lpo}_1 = (V_1, <_1, l_1)$, $\mathrm{lpo}_2 = (V_2, <_2, l_2)$ and $\mathrm{lpo} = \mathrm{lpo}_1; \mathrm{lpo}_2$. Let $v^i_{init}$ and $v^i_{final}$ be the initial and final nodes of the $\star$-extension of $\mathrm{lpo}_i$, $i = 1, 2$, and let $v_{init}$ and $v_{final}$ be the initial and final node of the $\star$-extension of lpo. First define $s|_{\mathrm{lpo}_1}$ on the edges of $\mathrm{lpo}_1^\star$ by:
(i) $\forall e \in V_1 \times V_1 : \; s|_{\mathrm{lpo}_1}(e) = s(e)$,
(ii) $\forall v \in V_1 : \; s|_{\mathrm{lpo}_1}(v^1_{init}, v) = s(v_{init}, v)$,
(iii) $\forall v \in V_1 : s|_{\mathrm{lpo}_1}(v, v^1_{final}) = \sum_{w \in V_2 \cup \{v_{final}\}} s(v, w)$,
(iv) $s|_{\mathrm{lpo}_1}(v^1_{init}, v^1_{final}) = \sum_{w \in V_2 \cup \{v_{final}\}} s(v_{init}, w)$.
Since by this construction the intoken and outtoken flow of nodes in $V_1$ and of the initial node are not changed, $s|_{\mathrm{lpo}_1}$ fulfills $(*)$ and corresponds to $p_s$. We define $s|_{\mathrm{lpo}_2}$ on the edges of $lpo_2^\star$ by
(i) $\forall e \in V_2 \times V_2 : \; s|_{\mathrm{lpo}_2}(e) = s(e)$,
(ii) $\forall v \in V_2 : \; s|_{\mathrm{lpo}_2}(v, v^2_{final}) = s(v, v_{final})$,
(iii) $\forall v \in V_2 : \; s|_{\mathrm{lpo}_2}(v^2_{init}, v) = \sum_{w \in V_1 \cup \{v_{init}\}} s(w, v)$,
(iv) $s|_{\mathrm{lpo}_2}(v^2_{init}, v^2_{final}) = \sum_{w \in V_1 \cup \{v_{init}\}} s(w, v_{final})$.
Since by this construction the intoken and outtoken flow of nodes in $V_2$ and of the final node are not changed, $s|_{lpo_2}$ fulfills $(*)$ and corresponds to a place $p'_s$ with $Init(p'_s) = Init(p_s) + Prod(\mathrm{lpo}_1, p_s)$. □

**Lemma 20.** *Consider two LPOs $\mathrm{lpo}_1$ and $\mathrm{lpo}_2$, a function $r_1$ on the edges of $\mathrm{lpo}_1^\star$ fulfilling $(*)$ with corresponding place $p$ and a function $r_2$ on the edges of $\mathrm{lpo}_2^\star$ fulfilling $(*)$ with corresponding place $p_2$, such that $W(p_2, t) = W(p, t)$ and $W(t, p_2) = W(t, p)$ for every $t \in T$ and $Final(\mathrm{lpo}_1, p) \geq Init(p_2)$ for a place $p$. Then there is a function $r$ on the edges of $\mathrm{lpo}^\star = (\mathrm{lpo}_1; \mathrm{lpo}_2)^\star$ fulfilling $(*)$ with corresponding place $p$.*

*Proof.* Let $v^i_{init}$ and $v^i_{final}$ be the initial and final nodes of the $\star$-extension of $\mathrm{lpo}_i$, $i = 1, 2$, and let $v_{init}$ and $v_{final}$ be the initial and final node of the $\star$-extension of lpo. We define a function $r$ on the edges of $\mathrm{lpo}^\star$ by:
(i) $\forall e \in V_1 \times V_1 : \; r(e) = r_1(e)$,
(ii) $\forall e \in V_2 \times V_2 : \; r(e) = r_2(e)$,
(iii) $\forall v \in V_1 : \; r(v_{init}, v) = r_1(v^1_{init}, v)$,
(iv) $\forall v \in V_2 : \; r(v, v_{final}) = r_2(v, v^2_{final})$.
By now all nodes in $V_1$ got the *right* intoken flow, all nodes in $V_2$ got the *right* outtoken flow. It remains to distribute the final token flow of $\mathrm{lpo}_1$ onto the edges in $(\{v_{init}\} \cup V_1\}) \times (\{V_2 \cup \{v_{final}\})$, such that $r$ fulfills $(*)$ on the edges of $\mathrm{lpo}^\star$, i.e. such that each node in $(\{V_2 \cup \{v_{final}\})$ gets enough intoken flow. This is possible since $Final(\mathrm{lpo}_1, p) \geq Init(p_2)$. By construction the function $r$ satisfies $(*)$ and corresponds to $p$. □

## 5 Computing Regions of Term-based Partial Languages

In this section we develop an algorithm to calculate a finite representation of the saturated feasible net w.r.t. a partial language $\mathcal{L}(\alpha)$ given by an LPO-term $\alpha$. This algorithm solves the formulated synthesis problem.

The algorithm is based on a method to compute regions of an LPO-term as solutions of a finite homogenous linear inequation system. More precisely, we calculate a basis of the solution space of such a system. The places corresponding to such a basis span the set of all feasible places. That means the calculated net has the same partial language of runs as the saturated feasible net.

To construct a region of an LPO-term $\alpha$ we first need to construct a function $r$ on the set of edges $E_\alpha$ of $R(\alpha) = \{\mathrm{lpo}_1, \ldots, \mathrm{lpo}_{|R(\alpha)|}\}$ fulfilling $(*)$. Such a function can be found by solving a corresponding equation system $\mathbf{A}_R \cdot \mathbf{x} = \mathbf{0}$ ($\mathbf{A}_R$ finite). The vector $\mathbf{x}$ contains $|E_\alpha| = n$ entries. Considering a fixed numbering of the edges in $E_\alpha = \{e_1, \ldots, e_n\}$, a solution $\mathbf{x} = (x_1, \ldots, x_n)$ of $\mathbf{A}_R \cdot \mathbf{x} = \mathbf{0}$ defines a function $r$ on $E_\alpha$ via $r(e_i) := x_i$. The rows of $\mathbf{A}_R$ encode the property $(*)$ as follows: We order the events of $W_\alpha$ with the same label $t \in T$ in a set $W_t = \{v \in W_\alpha \mid l_\alpha(v) = t\} = \{v^t_1, \ldots, v^t_{|W_t|}\}$. For each set $W_t$ we define rows $\mathbf{a}^t_i$ (and rows $\mathbf{b}^t_i$) of $\mathbf{A}_R$ to ensure that the intoken (outtoken) flow of the first node $v^t_1$ and the $i$-th node $v^t_i$ with label $t$ are equal, $i = 2, \ldots, |W_t|$. We define $\mathbf{a}^t_i = (a^t_{i,1}, \ldots, a^t_{i,n})$, where $a^t_{i,j}$ equals $1$ if $e_j$ is an ingoing edge of $v^t_1$, equals $-1$ if $e_j$ is an ingoing edge of

$v_i^t$ and equals 0 otherwise. We define $\mathbf{b}_i^t = (b_{i,1}^t, \dots, b_{i,n}^t)$, where $b_{i,j}^t$ equals 1 if $e_j$ is an outgoing edge of $v_1^t$, equals $-1$ if $e_j$ is an outgoing edge of $v_i^t$ and equals 0 otherwise. Then it holds $\mathbf{a}_i^t \cdot \mathbf{x} = 0$ ($\mathbf{b}_i^t \cdot \mathbf{x} = 0$) if and only if the intoken (outtoken) flows of $v_1^t$ and $v_i^t$ are equal. Finally, to ensure that all LPOs have the same initial token flow, we add rows $c_i$, $2 \leqslant i \leqslant |R(\alpha)|$ to $\mathbf{A}_R$. We define $\mathbf{c}_i = (c_{i,1}, \dots, c_{i,n})$, where $c_{i,j}$ equals 1 if $e_j$ is an outgoing edge of the initial node of $\mathrm{lpo}_1^\star$, equals $-1$ if $e_j$ is an outgoing edge of the initial node of $\mathrm{lpo}_i^\star$ and equals 0 otherwise. Then it holds $\mathbf{c}_i \cdot \mathbf{x} = 0$ if and only if the initial token flows of $\mathrm{lpo}_1$ and $\mathrm{lpo}_i$ are equal. Each non-negative integer solution of $\mathbf{A}_R \cdot \mathbf{x} = \mathbf{0}$ corresponds to a function $r : E_\alpha \to \mathbb{N}$ given by $r(e_i) = x_i$ fulfilling $(*)$ w.r.t. $R(\alpha)$ and vice versa.

Given a non-negative integer solution $\mathbf{x}$ of $\mathbf{A}_R \cdot \mathbf{x} = \mathbf{0}$, the set of LPOs $I(\alpha)$ should additionally fulfill $(**)$. For each $\mathrm{lpo}_i = (V_i, <_i, l_i) \in I(\alpha) = \{\mathrm{lpo}_1, \dots, \mathrm{lpo}_{|I(\alpha)|}\}$, we define a row $\mathbf{d}_i = (d_{i,1}, \dots, d_{i,n})$ of a second matrix $\mathbf{A}_I$ such that the inequation $\mathbf{d}_i \cdot \mathbf{x} \geq \mathbf{0}$ ensures $Prod(\mathrm{lpo}_i, p_r) \geq 0$. For each $t \in T$, we fix one node $v_t \in W_\alpha$ with $l(v_t) = t$ (such a node always exists by construction of $R_\alpha$). We define for each edge $e_j$ the entries $d_{i,j} = d_{i,j}^{out} - d_{i,j}^{in}$ of the vector $\mathbf{d}_i$. The value $d_{i,j}^{out}$ equals $|V_i|_l(t)$ if $e_j$ is an outgoing edge of $v_t$ ($t \in T$) and 0 otherwise. The value $d_{i,j}^{in}$ equals $|V_i|_l(t)$ if $e_j$ is an ingoing edge of $v_t$ ($t \in T$) and 0 otherwise. Then it holds $\mathbf{d}_i \cdot \mathbf{x} \geq \mathbf{0}$ if and only if $Prod(\mathrm{lpo}_i, p_r) \geq 0$.

**Theorem 21.** *Given an LPO-term $\alpha$ there is a one-to-one correspondence between the regions $r$ of $\alpha$ and the non-negative integer solutions of* $\mathbf{A}_R \cdot \mathbf{x} = \mathbf{0}, \mathbf{A}_I \cdot \mathbf{x} \geq \mathbf{0}$.

Thus we can compute regions of $\alpha$ and subsequently places of the searched saturated feasible p/t-net by solving a finite linear homogenous inequation system. The set of solutions of such a system is called a *polyhedral cone*. According to a theorem of Minkowski polyhedral cones are finitely generated, that means there are finite solutions $\mathbf{y}_1, \dots, \mathbf{y}_n$ (also called *basis solutions*) such that each element $\mathbf{x}$ of the polyhedral cone is a non-negative linear sum $\mathbf{x} = \sum_{i=1}^n \lambda_i \mathbf{y}_i$ for some $\lambda_1, \dots, \lambda_n \geqslant 0$. Such basis solutions $\mathbf{y}_1, \dots, \mathbf{y}_n$ can be effectively computed. Since all inequations contain only integer coefficients, the entries of all $\mathbf{y}_i$ can be chosen as integers.

Lemma 13 in [17] shows that all places which do not correspond to a basis solution can be deleted from the saturated feasible p/t-net without changing its partial language of runs. That means computing such a basis yields a finite representation of the saturated feasible p/t-net, called *basis representation*. Altogether to solve the formulated synthesis problem, we add to the set of all transitions (given by the labels appearing in $\alpha$) the places corresponding to basis solutions of the constructed inequation system. The described synthesis algorithm is shown in Algorithm 1.

Figure 4, part (b), shows a part of the net synthesized from the LPO-term $b + (a; (a \parallel b)^\star)$ by this algorithm. Altogether, the algorithm computes 12 basis regions (places) for this example, where those not shown in the figure are less restrictive than the drawn places. It is easy to see, that the net still generates the occurrence sequences $abb, ababb, abababb, \dots$ which do not belong to the partial language specified by $b + (a; (a \parallel b)^\star)$. That means, this partial language is not a net language. In general the runtime of the presented synthesis algorithm (Algorithm 1) is not polynomial. In particular the set of basis solutions can be exponential in the worst case, but in practice the number of basis solutions of such inequation systems is often small. We are currently working on an implementation of the algorithm.

```
1: $A_R, A_I \leftarrow EmptyMatrix$
2: for all $t \in T$ do
3:     for $m = 1$ to $|W_t| - 1$ do
4:         $A_R.addRows(a_m^t, b_m^t)$
5:     end for
6: end for
7: for $m = 1$ to $|R(\alpha)| - 1$ do
8:     $A_R.addRow(c_m)$
9: end for
10: for $m = 1$ to $|I(\alpha)|$ do
11:     $A_I.addRow(d_m)$
12: end for
13: $Solutions \leftarrow getBasisSolutions(A_R, A_I)$
14: $(N, m_0) \leftarrow (\emptyset, T, \emptyset, \emptyset, \emptyset)$
15: for all $r \in Solutions$ do
16:     $(N, m_0).addCorrespondingPlace(r)$
17: end for
18: return $(N, m_0)$
```

**Algorithm 1:** Calculates a net $(N, m_0)$ from an LPO-term $\alpha$ which solves the formulated synthesis problem.

## 6 Conclusion

In contrast to [17] we did not present an algorithm to decide if $\mathcal{L}(N, m_0) = \mathcal{L}(\alpha)$ for the computed p/t-net $(N, m_0)$. Such an algorithm cannot be developed similarly as in [17]. On contrary, it is an open question whether this equality problem is actually decidable. We believe that there is an algorithm deciding the equality of those language, but to prove this, an elaborated examination of the class of partial languages generated by LPO-terms is needed. One possibility is to consider so called *wrong continuations*. A wrong continuation is a not specified LPO minimally extending a specified LPO. The basic idea is to calculate a finite set of *wrong continuations* of LPOs specified by an LPO-term satisfying that the synthesized net exactly generates the specified partial language if and only if none of these wrong continuations is enabled in this net. The problem is that it is not clear whether such a finite set of wrong continuations exists in general, i.e. whether the (infinite) set of all wrong continuations is finitely generated. For sequential languages, it is proven that if a language satisfies certain conditions on *semi-linearity*, the set of wrong continuations is finitely generated and therefore the equality problem is decidable [5]. This is the case for example for regular and deterministic context free languages and can possibly be carried over to partial languages generated by LPO-terms. There are also several results concerning context-equivalence of LPOs w.r.t partial languages [7], which are perhaps useful to establish the result. Another, more indirect, approach is to translate a partial language given by an LPO-term into its corresponding step language. We believe that such a step language can be represented by a regular expression over a finite set of steps. Interpreting steps as singletons, existing techniques for sequential languages as described above can be applied to decide on language equality on the level of steps. Then it can be used that the synthesized net exactly generates the given partial language if and only if it exactly generates the corresponding step language and the partial language is *complete w.r.t. the step language* (this means that *each* LPO, whose corresponding step sequences belong to the step language, belongs to the partial language [13]). Thus, it remains to decide on completeness of the partial language w.r.t. the corresponding step language. Also this is an open problem for partial languages defined by LPO-terms. Nevertheless, Theorem 10 shows that the net $(N, m_0)$ computed from $\alpha$ in any case represents the best upper approximation to the behaviour given by the partial language $\mathcal{L}(\alpha)$.

## References

[1] E. Badouel and P. Darondeau. Theory of Regions. In W. Reisig; G. Rozenberg, editor, *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer, 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] P. Darondeau. Unbounded Petri Net Synthesis. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 413–438. Springer, 2003.

[6] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-Structures. Part I: Basic Notions and the Representation Problem / Part II: State Spaces of Concurrent Systems. *Acta Inf.*, 27(4):315–368, 1989.

[7] J. Fanchon and R. Morin. Regular Sets of Pomsets with Autoconcurrency. In L. Brim; P. Jancar; M. Kretínský; A. Kucera, editor, *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 402–417. Springer, 2002.

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

[9] P. Hoogers, H. Kleijn, and P. Thiagarajan. A Trace Semantics for Petri Nets. *Information and Computation*, 117(1):98–114, 1995.

[10] P. Hoogers, H. Kleijn, and P. Thiagarajan. An Event Structure Semantics for General Petri Nets. *Theoretical Computer Science*, 153(1&2):129–170, 1996.

[11] 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.

[12] G. Juhas, R. Lorenz, and S. Mauser. Causal Semantics of Algebraic Petri Nets distinguishing Concurrency and Synchronicity. *Fundam. Inform.*, page to appear, 2007.

[13] G. Juhás, R. Lorenz, and S. Mauser. Complete Process Semantics for Inhibitor Nets. In J. Kleijn and A. Yakovlev, editors, *ICATPN*, volume 4546 of *Lecture Notes in Computer Science*, pages 184–203. Springer, 2007.

[14] D. Kuske and R. Morin. Pomsets for Local Trace Languages - Recognizability, Logic & Petri Nets. In C. Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 426–441. Springer, 2000.

[15] K. Lodaya and P. Weil. Series-Parallel Posets: Algebra, Automata and Languages. In M. Morvan; C. Meinel; D. Krob, editor, *STACS*, volume 1373 of *Lecture Notes in Computer Science*, pages 555–565. Springer, 1998.

[16] K. Lodaya and P. Weil. Series-Parallel Languages and the Bounded-Width Property. *Theor. Comput. Sci.*, 237(1-2):347–380, 2000.

[17] R. Lorenz, R. Bergenthum, J. Desel, and S. Mauser. Synthesis of Petri Nets from Finite Partial Languages. In *ACSD*, pages 157–166. IEEE Computer Society, 2007.

[18] 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.

[19] V. Pratt. Modelling Concurrency with Partial Orders. *Int. Journal of Parallel Programming*, 15:33–71, 1986.

[20] W. Reisig. *Petrinetze - Eine Einführung.* Springer, 2 edition, 1986.

[21] 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.

[22] W. Vogler. Partial Words Versus Processes: a Short Comparison. In G. Rozenberg, editor, *Advances in Petri Nets: The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 292–303. Springer, 1992.