Towards Synthesis of Petri Nets from Scenarios

Robert Lorenz[1] and Gabriel Juhás[2]

[1] Lehrstuhl für Angewandte Informatik
Katholische Universität Eichstätt-Ingolstadt, Eichstätt, Germany
robert.lorenz@ku-eichstaett.de
[2] Faculty of Electrical Engineering and Information Technology
Slovak University of Technology, Bratislava, Slovakia
gabriel.juhas@stuba.sk

Abstract. Given a set of scenarios, we answer the question whether this set equals the set of all executions of a Petri net.

Formally, scenarios are expressed by (isomorphism classes of) labelled partial orders (LPOs), also known as pomsets or partial words. An LPO is an execution of a Petri net if it is a sequentialization of an LPO generated by a process of the net. We propose a definition of regions for a set of LPOs, i.e for a partial language. Given a partial language of scenarios, we prove a necessary and sufficient condition (based on regions) for the partial language of scenarios to be the partial language of executions of a place/transition Petri net. Finally, we prove our notion of regions to be consistent with the notion of regions of trace languages.

1 Introduction

Scenario based specifications of information systems become recently very popular both in theory and practice, see e.g. [16]. There are approaches which concentrate on scenario mining [1, 2]. There are also several methods for the synthesis of a system model from a set of scenarios (see e.g. [23, 15, 3, 28, 7, 8] for some of them), the scenario based verification of the system model ([21, 9]), and for test generation and validation purposes ([22]). They differ mostly in the formalisms used for scenarios and the system model. In this paper, we consider place/transition Petri nets (p/t-nets) as system models.

The simplest way to describe scenarios for p/t-nets is to use sequences of fired transitions. Thus, the set of scenarios specifying the system can be described by a formal language over the set of transitions or by a transition system. For this kind of specifications the synthesis problem is solved using the theory of regions [3, 6, 7, 8, 5, 4, 25]. However, sequences lack any information about independence between occurring events and Petri nets are popular mainly because they allow to easily describe their non-sequential behavior and concurrency of occurring events.

There are several different ways how to formalize the non-sequential behavior of Petri nets, most of them based on a partial order between events labelled by transitions. In such a partial order, an ordering of events a and b expresses that the occurrence of a precedes the occurrence of b. The absence of an ordering expresses their concurrent (independent) occurrence.

The probably most common way to express the partial order based behavior of Petri nets is by labelled causal (occurrence) nets and processes [12, 13]. Dropping the conditions and keeping the events in a process leads to a labelled partial order (LPO), called *run*. Runs capture the causal ordering of events. Naturally, events which are independent can occur sequentially in any order. Thus, adding order to a run still leads to a possible execution. For example, any occurrence sequence of transitions, which can be seen as a labelled total order, sequentializes a run. Generalizing this relationships, any LPO which sequentializes a run, is a possible *execution* of the net.

A partial order based semantics can also be defined through sequences of (concurrent) steps of transitions. A step of transitions is a multi-set of transitions. It is enabled in a marking, if there are enough tokens to fire all transitions concurrently, where a transition can be contained in a step more than once (auto-concurrency). The non-sequential behavior of p/t-nets can be described by step transition systems. A region based characterization of those step transition systems describing the non-sequential behavior of p/t-nets was given in [24]. The notion of enabled steps of transitions leads to the notion of *enabled LPOs*: Given a partial order of events labelled by transitions, it is said to be enabled to occur if the following condition is satisfied for every cut (i.e. for every maximal set of independent events) [14]: If all the events before the cut have occurred, then in the reached marking the step of transitions determined by the cut is enabled to occur. It was proven in [20, 29] that an LPO is an execution if and only if it is enabled.

There are also other descriptions of the non-sequential behavior of Petri nets. For example, one can take step sequences and relate them via an equivalence, getting so called *traces*. A region based characterization of trace languages which describe the non-sequential behavior of p/t-nets was given in [17].

Because the identity of vertices (events) of LPOs is not important, it is usual to distinguish LPOs only up to isomorphism. An isomorphism class of LPOs can be also understood as a *partial word* (over a set of labels), similarly as a labelled total order (a sequence) is understood as a (total) word [14]. A set of LPOs (over a set of labels) is called a *partial language* (over a set of labels). Another view is to understand isomorphism classes of LPOs as *pomsets* (partially ordered multi-sets (of labels)) [27].

In [14] Grabowski characterizes partial languages of executions of 1-safe Petri nets. In case of p/t-nets, there is no characterization of partial languages of executions similar to those based on regions for total languages or trace languages. In this paper, we fill this gap.

In [18] we gave a characterization of p/t-net executions based on so called *token flow functions*. Roughly speaking, a token flow function labels any arc connecting an event a with an event b in an LPO by the number of tokens produced by the occurrence of a which are consumed by the occurrence of b in a fixed place p. For each event the sum of token numbers annotated to its outgoing arcs defines the *outgoing token flow* (w.r.t. p). The sum of token numbers annotated to ingoing arcs defines its *ingoing token flow* (w.r.t. p). Given a p/t-net, we formulated a necessary and sufficient condition for the executability of an LPO, called the *token flow property*. An LPO fulfills the token flow property w.r.t. to a marked p/t-net if and only if for every place p there exists a token flow function of the LPO satisfying the following: for every event, its outgoing token flow (w.r.t. p) does not exceed the number of tokens produced by the corresponding transition, provided that for every event its ingoing token flow (w.r.t. p) equals the number of tokens consumed by the corresponding transition of the p/t-net.

The notion of regions presented in this paper is based on the extension of the notion of token flow functions from single LPOs to partial languages of LPOs. Regions are exactly those extended token flow functions, for which there is an upper bound for the outgoing token flows of events and equally labelled events have equal ingoing token flows. Given a set of scenarios in form of a partial language of LPOs, its regions are used to define its so called *associated p/t-net*: The regions form the set of places and the event labels form the set of transitions. Given a region r and a transition t, the ingoing token flow of events labelled by t defines the weight of the arc connecting r with t, and the maximum of all outgoing token flows of events labelled by t defines the weight of the arc connecting t with r.

As the main result of the paper, we show that a partial language $\mathcal{L}$ of LPOs equals the partial language of executions of a p/t-net if and only if all executions of the associated p/t-net belong to $\mathcal{L}$. We also prove that our notion of regions is consistent with the notion of regions for trace languages as given in [17].

The main result of this paper is a step towards the synthesis of p/t-nets from sets of scenarios. It is also a step towards the synthesis of p/t-nets from incomplete sets of scenarios $\mathcal{L}$, since the p/t-net associated to $\mathcal{L}$ can be considered as the net with the ”smallest” behavior including $\mathcal{L}$. The necessary steps to derive synthesis algorithms, namely a finite representation of an infinite set of scenarios and a necessary and sufficient condition for existence of a finite representation of the (possibly) infinite set of regions are discussed in the conclusion (Section 4).

The rest of the paper is organized as follows: In Section 2 we state basic mathematical notations and give the definitions of LPOs and of syntax and semantics of p/t-nets. In Section 3 we briefly restate the definitions and results from [18], define regions of partial languages, prove a characterization of partial languages of executions of p/t-nets based on this notion of regions and finally show that our notion of regions is consistent with the notion of regions of trace languages as presented in [17].

2 Preliminaries

2.1 Mathematical Notations

By $\mathbb{N}$ we denote the *nonnegative 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. By id_A we denote the *identity function* on a set A, and by 1_X we denote the *characteristic function* of a subset $X \subseteq A$ (given by $1_X(a) = 1$ for $a \in X$ and $1_X(a) = 0$ otherwise). Given a finite set A, the symbol $|A|$ denotes the *cardinality* of A. The set of all subsets of a set A we denote by 2^A. The set of all *multi-sets* over a set A is the set $\mathbb{N}^A$ of all functions $f : A \to \mathbb{N}$. We do not distinguish between a subset $X \subseteq A$ and the multi-set 1_X. 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. Given a binary relation $R \subseteq A \times A$ over a set A, the symbol R^+ denotes the *transitive closure* of R, and R^* the *reflexive transitive closure* of R.

2.2 Labelled Partial Orders

In this subsection we recall the definition of labelled partial orders (LPOs). It is based on the notion of directed graphs. A *directed graph* is a pair $(V, \rightarrow)$, where V is a finite *set of nodes* and $\rightarrow \subseteq V \times V$ is a binary relation over V called the *set of arcs*. As usual, given a binary relation $\rightarrow$, we write $a \rightarrow b$ to denote $(a, b) \in \rightarrow$.

Definition 1 ((Labelled) partial order)
A partial order *is a directed graph* $\mathrm{po} = (V, <)$*, where* $<$ *is an irreflexive and transitive binary relation on* V*. A* labelled partial order *(LPO) is a triple* $\mathrm{lpo} = (V, <, l)$*, where* $(V, <)$ *is a partial order, and* $l : V \rightarrow T$ *is a* labelling function *with* set of labels T.

Two nodes $v, v' \in V$, $v \neq v'$, 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: $\forall x, y \in C : x \,\mathrm{co}\, y$. A *cut* is a maximal co-set. If co is transitive, then the partial order $(V, <)$ is called *stepwise linear*.

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 $s \in C$. For two co-sets C', C we write $C' < C$, if $s' < C$ for all elements $s' \in C'$.

For each co-set C of $(V, <)$ the partial order $(V', <|_{V' \times V'})$ with $V' = \{v \in V \mid v < C\} \cup C$ is called *prefix* of $(V, <)$.

Definition 2 (Sequentialization, step-linearization)
Given two partial orders $\mathrm{po}_1 = (V, <_1)$ *and* $\mathrm{po}_2 = (V, <_2)$*, we say that* po_2 is a *sequentialization* of po_1 *if* $<_1 \subseteq <_2$*, and a* proper sequentialization *if additionally* $<_1 \neq <_2$.

If po_2 is stepwise linear and a sequentialization of po_1, the po_2 is called a step-linearization *of po_1.*

We use the above notations defined for partial orders also for labelled partial orders. If X is the set of labels of $\mathrm{lpo} = (V, <, l)$, i.e. $l : V \rightarrow X$, then for a set $V' \subseteq V$, we define the multi-set $|V'|_l \subseteq \mathbb{N}^X$ by $|V'|_l(x) = |\{v \in V \mid v \in V' \wedge l(v) = x\}|$.

In the case, an LPO $\mathrm{lpo} = (V, <, l)$ is stepwise linear, the relation $\mathrm{co} \cup \{(v, v) \mid v \in V\}$ is an equivalence relation on V. By $[v]_{\mathrm{co}}$ we denote the equivalence classes of this equivalence relation. Let $\{[v]_{\mathrm{co}} \mid v \in V\} = \{\nu_1, \ldots, \nu_k\}$ such that $i < j \Longrightarrow \nu_i < \nu_j$. Then $|\nu_1|_l \ldots |\nu_k|_l$ is the *step sequence* of lpo.

We will often consider LPOs only up to isomorphism. As usual, two LPOs $(V, <, l)$ and $(V', <', l')$ are called *isomorphic*, if there is a bijective mapping $\psi : V \rightarrow V'$ 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 [lpo] we will denote the set of all LPOs isomorphic to lpo. The LPO lpo is said to *represent* the isomorphism class [lpo].

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

2.3 Place/Transition-Nets

In this subsection we give the definitions of p/t-nets and their semantics based on processes and labelled partial orders (also known as partial words [14] or pomsets [27]).

The syntax of p/t-nets is based on the notion of nets. 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*.

Let $x \in P \cup T$. The *preset* ${}^\bullet x$ *of* x is the set $\{y \in P \cup T \mid (y,x) \in F\}$. The *postset* $x^\bullet$ *of* x is the set $\{y \in P \cup T \mid (x,y) \in F\}$. Given a set $X \subseteq P \cup T$, this notation is extended by ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. For technical reasons, we consider only nets in which every transition has a nonempty preset and postset.

Definition 4 (Place/transition net)
A place/transition-net *(shortly* 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*.

We omit the semantics of a p/t-net $N = (P,T,F,W)$ based on occurrence sequences and step sequences. The semantics of p/t-nets based on processes is defined using occurrence nets. An *occurrence net* is a net $O = (B,E,G)$ such that $|{}^\bullet b|, |b^\bullet| \leqslant 1$ for every $b \in B$ (i.e. places are *unbranched*), and O is *acyclic* (i.e. G^+ is a partial order). Places of an occurrence net are called *conditions* and transitions of an occurrence net are called *events*.

The set of conditions of an occurrence net $O = (B,E,G)$ which are minimal (maximal) according to G^+ is denoted by $\mathrm{Min}(O)$ ($\mathrm{Max}(O)$). Clearly, $\mathrm{Min}(O)$ and $\mathrm{Max}(O)$ are cuts w.r.t. G^+ (recall that events have nonempty pre- and postsets by assumption).

Definition 5 (Process)
Let (N, m_0) *be a marked p/t-net,* $N = (P,T,F,W)$*. A* process *of* (N, m_0) *is a pair* $K = (O, \rho)$*, where* $O = (B,E,G)$ *is an occurrence net and* $\rho : B \cup E \to P \cup T$ *is a labelling function, satisfying:*

(i) $\rho(B) \subseteq P$ *and* $\rho(E) \subseteq T$*,*
(ii) $\forall e \in E, \forall p \in P : |\{b \in {}^\bullet e \mid \rho(b) = p\}| = W((p, \rho(e)))$ *and* $\forall e \in E, \forall p \in P : |\{b \in e^\bullet \mid \rho(b) = p\}| = W((\rho(e), p))$*, and*
(iii) $\forall p \in P : |\{b \in Min(O) \mid \rho(b) = p\}| = m_0(p)$*.*

Definition 6 (Run, Execution)
Let $K = (O, \rho)$*,* $O = (B,E,G)$*, be a process of a marked p/t-net* (N, m_0)*. Then* $\mathrm{lpo}_K = (E, G^+|_{E \times E}, \rho|_E)$ *is called* run *of* (N, m_0) *representing* K*.*

A run lpo *of* (N, m_0) *is said to be* minimal*, if* lpo *is not a proper sequentialization of a run* lpo'*.*

A sequentialization lpo *of a run* lpo' *of* (N, m_0) *is called an* execution *of* (N, m_0)*.*

If lpo *is an execution of* (N, m_0)*, then the isomorphism class* [lpo] *is also called* execution *of* (N, m_0)*, . Denote* $\mathfrak{Lpo}(N, m_0) = \{[\mathrm{lpo}] \mid \mathrm{lpo}$ *execution of* $(N, m_0)\}$ *the* partial language of executions *of a marked p/t-net* (N, m_0)*.*

Another definition of the non-sequential semantics of p/t-nets is given by the notion of *enabled LPOs*:

Definition 7 (Enabledness). *Let (N, m_0) be a marked p/t-net, $N = (T, P, F, W)$. An LPO* $\text{lpo} = (V, <, l)$ *with $l : V \to T$ is called* enabled (to occur) w.r.t. (N, m_0) *if for every cut C of* lpo *and every $p \in P$:*

$$m_0(p) + \sum_{v \in V \wedge v < C} (W((l(v), p)) - W((p, l(v)))) \geq \sum_{v \in C} W((p, l(v))).$$

Its occurrence *leads to the marking m' given by* $m'(p) = m(p) + \sum_{v \in V}(W((l(v), p)) - W((p, l(v))))$ *for $p \in P$.*

An isomorphism class [lpo] is called *enabled* w.r.t. a marked p/t-net (N, m_0), if lpo is *enabled* w.r.t. (N, m_0). An important result relating the notions of *enabled LPOs* and *executions* was proven in [20, 29].

Theorem 1. *Let (N, m_0) be a marked p/t-net. An LPO* lpo *is enabled w.r.t. (N, m_0) if and only if* lpo *is an execution of (N, m_0).*

3 Synthesis of p/t-Nets

In this section, we first recall an alternative characterization of LPOs to be an execution called *token flow property*, we presented in [18]. We then discuss the formal problem setting of p/t-net synthesis from a given partial language. After that, we propose a notion of regions of partial languages and derive a characterization of those partial languages which allow a p/t-net synthesis. Finally, we establish the relationship between our notion of regions and the definition of regions of a trace language as defined in [17].

3.1 Token Flow Property

There is another characterization of LPOs to be executions of a marked p/t-net called *token flow property*. For motivation purposes we shortly recall the basic definitions and results.

Fix a marked p/t-net (N, m_0) and a place p of $N = (P, T, F, W)$. Given an LPO $\text{lpo} = (V, \prec, l)$ with $l(V) = T$ we assign non-negative integers to its edges through a so called *token flow function*. The aim is to find a token flow function x assigning values $x((v, v'))$ to edges (v, v') in such a way that there is a process with exactly $x((v, v'))$ post-conditions of v labelled by p which are also pre-conditions of v'. Thus, such a token flow function of lpo abstracts from the individuality of conditions of a process and encodes the flow relation of this process by natural numbers. Clearly, finding such a token flow function for every place means that lpo is a sequentialization of the run representing this process. In order to simplify the formal definition of the token flow property, let us define an extension of $\text{lpo} = (V, \prec, l)$ by adding an initial node which is smaller than all nodes from V and is labelled by a new label. It represents a transition producing the initial marking and helps to avoid several case differentiations in the following definitions.

Definition 8 (0-extension, of an LPO). *Let* $\text{lpo} = (V, \prec, l)$ *be a LPO. Then an LPO* $\text{lpo}^0 = (V^0, \prec^0, l^0)$*, where $V^0 = (V \cup \{v_0\})$ with $v_0 \notin V$, $\prec^0 = \prec \cup (\{v_0\} \times V)$, and $l^0(v_0) \notin l(V)$ and $l^0|_V = l$, is called* 0-extension of lpo.

We denote for an LPO $\text{lpo} = (V, <, l)$, a function $x :<\to \mathbb{N}$ and $v \in V$:

- $In(v, x) = \sum_{v' < v} x((v', v))$.
- $Out(v, x) = \sum_{v < v'} x((v, v'))$.

Definition 9 (Token flow function, of an LPO). *Let* $\text{lpo} = (V, \prec, l)$ *be an LPO and* $\text{lpo}^0 = (V^0, \prec^0, l^0)$ *be a* 0*-extension of* lpo. *A function* $x :\prec^0\to \mathbb{N}$ *is called* token flow function *of* lpo, *if it satisfies the following property:*

(Tff) x *is* consistent *with the labelling* l *in the following sense:*

$$\forall v, v' \in V^0 : \; l(v) = l(v') \implies In(v, x) = In(v', x).$$

We denote $In(v, x)$ *the* intoken flow of v w.r.t. x *and* $Out(v, x)$ *the* outtoken flow of v w.r.t. x *for* $v \in V$.

Example 1. Figure 1 shows a p/t-net, a process of this p/t-net and a 0-extension lpo^0 of the run representing this process. The LPO lpo^0 has annotated a token flow function. Its intoken flow (outtoken flow) equals 0 (2) for the a- and b-labelled node, resp. 2 (0) for the c-labelled nodes.

This definition differs from that in [18]. While in [18] token flow functions were defined as general as possible, we here additionally require property $(\mathbf{Tff})$. This is more intuitive and does not restrict the setting or change the argumentations, since $(\mathbf{Tff})$ is implicitly contained in the token flow property defined below. Each process of a marked p/t-net defines a token flow function of the run representing this process in the following way:

Let $K = (O, \rho)$ be a process of a marked p/t-net (N, m_0) with $O = (B, V, G)$ and let $\text{lpo} = (V, <, l)$ be the run representing K. Let $\text{lpo}^0 = (V^0, <^0, l^0)$ be a 0-extension of lpo. Denote $v_0^\bullet = \text{Min}(O)$. We define for every place $p \in P$ the *canonical token flow function* $x_p :<^0\to \mathbb{N}$ *of* lpo *w.r.t.* p as follows:

$$x_p((v, v')) = |\{b \in B \mid \rho(b) = p \wedge b \in v^\bullet \cap {}^\bullet v'\}|.$$

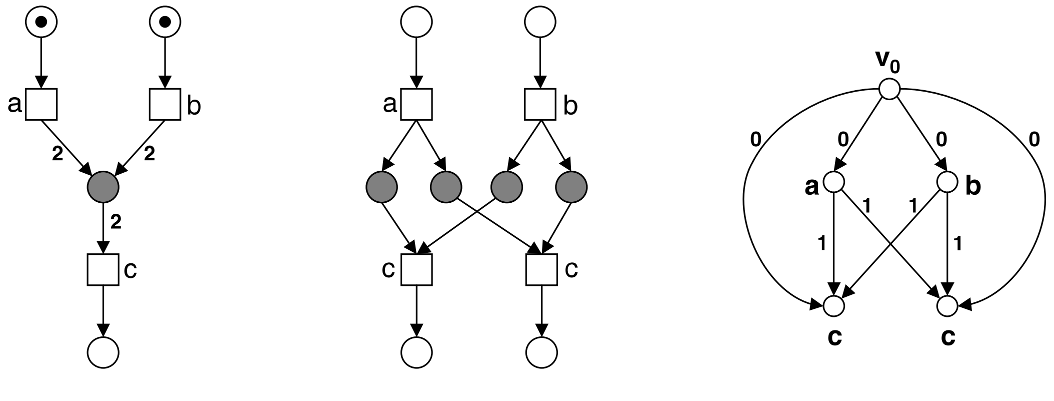

Fig. 1. A p/t-net (left picture), a process of this p/t-net (middle picture) and a 0-extension of the run representing this process together with annotated canonical token flow function w.r.t. the grey place (right picture)

Example 2. In Figure 1 there is shown the canonical token flow function of the shown run w.r.t. the grey place.

Observe, that by definition 5 (ii) the canonical token flow function of a run lpo w.r.t. a place p fulfills $(\mathbf{Tff})$ (what justifies the above definition). By definition, each canonical token flow function respects the weight function and the initial marking of (N, m_0) in the sense that the following two properties are fulfilled:

(IN) The intoken flow of an event v equals the number of tokens consumed from place p by the occurrence of transition $l(v)$.
(OUT) The outtoken flow of an event v is less or equal to the number of tokens which are produced by the occurrence of transition $l(v)$ in place p. In particular, the outtoken flow of the source event v_0 is less or equal to the number of tokens in place p of the initial marking m_0.

In general, we say that an arbitrary LPO, whose labels are transitions of (N, m_0), fulfills the token flow property w.r.t. (N, m_0), if for every place there exists a token flow function which fulfills the properties $(\mathbf{IN})$ and $(\mathbf{OUT})$.

Definition 10 (Token flow property). *Let* $\mathrm{lpo} = (V, \prec, l)$ *be an LPO with* $l(V) = T$, $\mathrm{lpo}^0 = (V^0, \prec^0, l^0)$ *be a* 0*-extension of* lpo *and let* (N, m_0) *be a marked p/t-net,* $N = (P, T, F, W)$. *Denote* $W((l(v_0), p)) = m_0(p)$ *for each place* $p \in P$. *We say that* lpo *fulfills the token flow property w.r.t.* (N, m_0) *if the following statement holds: For all* $p \in P$ *there is a token flow function* $x_p : \prec^0 \to \mathbb{N}$ *with*

(IN) $In(v', x_p) = W((p, l(v')))$*for every* $v' \in V$,
(OUT) $Out(v', x_p) \leqslant W((l(v'), p))$ *for every* $v' \in V^0$.

In [18] we showed:

Theorem 2. *An LPO is an execution of a marked p/t-net if and only if it fulfills the token flow property w.r.t. this marked p/t-net.*

3.2 Problem Setting

As mentioned in the introduction, in this paper we consider the problem of finding a marked p/t-net whose non-sequential behavior is represented by a given partial language $\mathcal{L}$. The *p/t-net synthesis problem* reads formally:

Given: A partial language $\mathcal{L}$.
Searched: A marked p/t-net (N, m_0) satisfying $\mathcal{L} = \mathfrak{Lpo}(N, m_0)$.

Obviously, a marked p/t-net may have an infinite number of (finite) runs, that means the partial language of executions of a marked p/t-net may be countably infinite. When considering the problem in practice, one has to restrict the setting to finite sets of finite LPOs or to use some adequate finite representation of infinite sets of finite LPOs. A detailed discussion of this topic is out of scope of this paper, but we will present some thoughts in the conclusion.

A partial language $\mathcal{L}$ of executions of a marked p/t-net satisfies the following immediate properties (induced by the definition of executions):

($\mathcal{L}_1$) The set of labels of $\mathcal{L}$ is finite.
($\mathcal{L}_2$) $\forall[\mathrm{lpo}] \in \mathcal{L}$: If lpo' is a prefix of lpo, then $[\mathrm{lpo}'] \in \mathcal{L}$.
($\mathcal{L}_3$) $\forall[\mathrm{lpo}] \in \mathcal{L}$: If lpo' is a sequentialization of lpo, then $[\mathrm{lpo}'] \in \mathcal{L}$.

Therefore, in the following, we will only consider partial languages which satisfy these conditions and denote them as *LPO-specifications*.

Definition 11 (LPO-specification). *A partial language $\mathcal{L}$ is called* LPO-specification*, if it satisfies the properties ($\mathcal{L}_1$) - ($\mathcal{L}_3$).*

For the following notions we will need a set L of concrete LPOs representing a given LPO-specification $\mathcal{L}$, i.e. satisfying $[\mathrm{lpo}] \in \mathcal{L} \iff \exists \mathrm{lpo}' \in L : [\mathrm{lpo}] = [\mathrm{lpo}']$. We denote $\mathcal{L}(L) = \{[\mathrm{lpo}] \mid \mathrm{lpo} \in L\}$. For technical reasons we will require such a representation L of $\mathcal{L}$ to fulfill the following properties.

(L1) The set of labels of L is finite.
(L2) $\forall \mathrm{lpo} \in L$: If lpo' is prefix of lpo, then $\mathrm{lpo}' \in L$.
(L3) $\forall \mathrm{lpo} \in L$: If lpo' is sequentialization of lpo, then $\mathrm{lpo}' \in L$.
(L4) $\forall (V, <, l), (V', <', l') \in L$: $v \in V \cap V' \Rightarrow l(v) = l'(v)$.
(L5) For each two LPOs $\mathrm{lpo} = (V, <, l), \mathrm{lpo}' = (V', <', l') \in L$ with $V \cap V' \neq \emptyset$ there is $\mathrm{lpo}'' \in L$ such that lpo and lpo' both are sequentializations of prefixes of lpo''.

That means, L should be prefix and sequentialization closed and should represent alternative executions by LPOs with disjoint sets of nodes (condition (L5) separates the node sets of executions which are in conflict). It is straightforward to observe that an LPO-specification always allows such a representation.

Remark 1. In particular, (L5) allows to decompose L into subsets X of LPOs with pairwise disjoint node sets, each representing one (infinite) run. Namely, each such set X is defined as a maximal prefix and sequentialization closed subset of L satisfying:

(Seq) Each two LPOs $\mathrm{lpo}, \mathrm{lpo}' \in X$ are sequentializations of prefixes of another LPO $\mathrm{lpo}'' \in X$.

For a set of BPOs L we denote $W_L = \bigcup_{(V,<,l)\in L} V$, $E_L = \bigcup_{(V,<,l)\in L} <$ and $l_L = \bigcup_{(V,<,l)\in L} l$ and write for an LPO $\mathrm{lpo} = (V, <, l) \in L$, a function $x : E_L \to \mathbb{N}$ and $v \in V$:

- $In_{\mathrm{lpo}}(v, x) = \sum_{v' < v} x((v', v))$.
- $Out_{\mathrm{lpo}}(v, x) = \sum_{v < v'} x((v, v'))$.

In examples we will always give such L by a set of minimal LPOs, such that each LPO in L is a sequentialization of some prefix of one of these minimal LPOs.

Example 3. Figure 2 shows the two different LPO-specifications $L_1 = \{\mathrm{lpo}_1, \mathrm{lpo}_2\}$ and $L_2 = \{\mathrm{lpo}_{3,n} \mid n \in \mathbb{N}\}$.

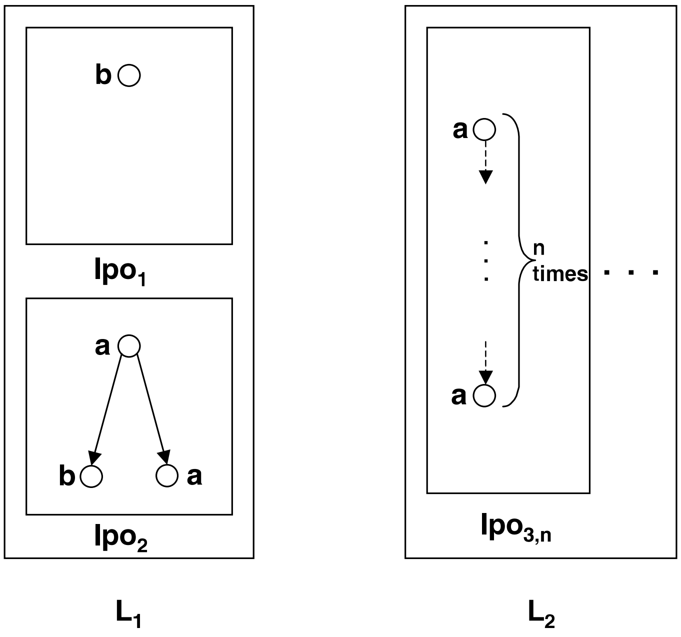

Fig. 2. Two LPO-specifications

3.3 Regions

For the rest of the paper we consider each LPO-specification given by a set of LPOs L satisfying (L1)-(L5). In subsection 3.1 we established that a token flow function of a partially ordered multi-set of transitions which fulfills the properties $(\mathbf{IN})$ and $(\mathbf{OUT})$ for some place defines a distribution of the tokens produced by a transition in this place onto the following transition in the given order, such that all transitions in the LPO are enabled w.r.t. this place. This gives rise to the idea to consider token flow functions of LPOs as regions, that define places of a p/t-net. For this, we define, as for LPOs, the notions of 0-*extension* and *token flow function* of L.

Definition 12 **(0-extension).** *Let* $v_0 \notin W_L$ *and* $\text{lpo}^0 = (V^0, <^0, l^0)$ *be a* 0*-extension of each* $\text{lpo} \in L$ *such that* $V^0 = V \cup \{v_0\}$*, and* $l_1^0(v_0) = l_2^0(v_0)$ *for each two* $(V_1, <_1, l_1), (V_2, <_2, l_2) \in L$*. Then the set* $L^0 = \{\text{lpo}^0 \mid \text{lpo} \in L\}$ *is called* 0-extension of L*. We denote* $W_L^0 = W_{L^0}$*,* $E_L^0 = E_{L^0}$ *and* $l_L^0 = l_{L^0}$*.*

Example 4. Figure 3 shows 0-extensions of the LPO-specifications L_1 and L_2 shown in Figure 2.

Definition 13 (Token flow function). *Let* L^0 *be a* 0*-extension of* L*. A function* $x : E_L^0 \to \mathbb{N}$ *is called* token flow function of L*, if it satisfies the following property:*

($\mathfrak{Tff}$) x *is* consistent *over all LPOs:*

$$\forall \text{lpo} = (V, <, l), \text{lpo}' = (V', <', l') \in L^0,$$
$$\forall v \in V^0, \forall v' \in (V')^0 :$$
$$l(v) = l'(v') \implies In_{\text{lpo}^0}(v, x) = In_{(\text{lpo}')^0}(v', x).$$

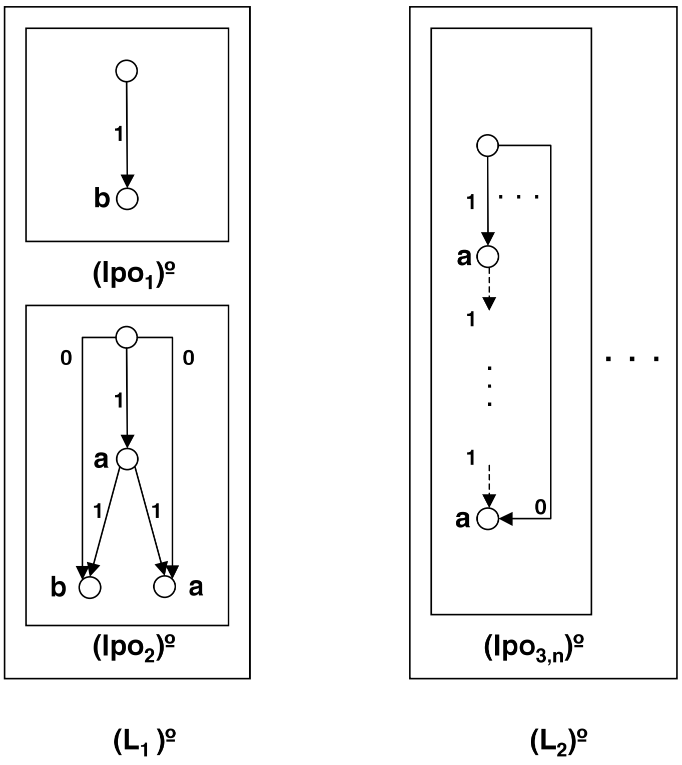

Fig. 3. 0-extensions of L_1 and L_2 with annotated possible token flow functions x_1 of L_1 and x_2 of L_2

Example 5. Figure 3 shows possible token flow functions x_1 of L_1 and x_2 of L_2 given by arc annotations.

Observe that according to $(\mathfrak{Tff})$ for $\mathrm{lpo} = (V, <, l) \in L$ and $v \in V$ the value of $In_{\mathrm{lpo}^0}(v, x)$ is independent of the choice of lpo. It is interpreted as the number of tokens consumed by transition $l(v)$. On the other hand, the value $Out_{\mathrm{lpo}^0}(v, x)$ is dependent on lpo and can even differ for equally labelled nodes. It is interpreted as the number of tokens produced by transition $l(v)$ which are consumed by other transitions. Not consumed tokens remain in the final marking of the considered lpo. Of course, a token flow function can represent a place of a p/t-net only if the number of produced tokens represented by $Out_{\mathrm{lpo}^0}(v, x)$ is bounded over all LPOs $\mathrm{lpo} \in L$:

Definition 14 (Region). *Let L^0 be a 0-extension of L. A token flow function $r : E_L^0 \to \mathbb{N}$ of L is called a* region of L *if there is $M \in \mathbb{N}$ such that*

$$(\mathbf{R}) \quad |\{Out_{\mathrm{lpo}^0}(v, r) \mid \mathrm{lpo} = (V, <, l) \in L,\ v \in V^0\}| \leqslant M.$$

We denote

- $\mathrm{Pre}_r(v) = In_{\mathrm{lpo}^0}(v, r)$ *for* $\mathrm{lpo} = (V, <, l) \in L$ *with* $v \in V$.
- $\mathrm{Post}_r(v) = \max\{Out_{\mathrm{lpo}^0}(v', r) \mid \mathrm{lpo} = (V, <, l) \in L,\ v' \in V^0,\ l_L^0(v') = l_L^0(v)\}$ *for* $v \in W_L^0$.
- $m_0(r) = \mathrm{Post}_r(v_0)$.

A region which does not equal the 0*-function is called* non-trivial.

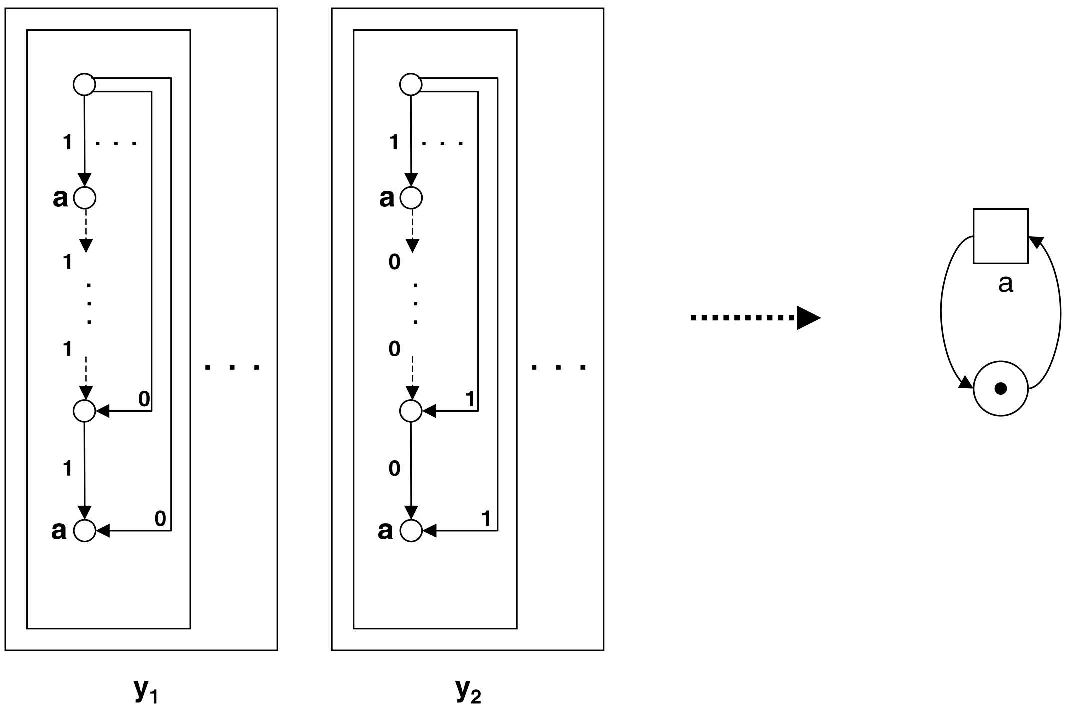

Fig. 4. Two token flow function y_1 and y_2 of L_2 and a part of the p/t-net associated to L_2

For a label a, a node v with $l_L(v) = a$ and a region r of L we also denote $\mathrm{Pre}_r(a) = \mathrm{Pre}_r(v)$ *and* $\mathrm{Post}_r(a) = \mathrm{Post}_r(v)$.

Example 6. The token flow functions x_1 and x_2 shown in Figure 3 are both regions. It holds: $\mathrm{Pre}_{x_1}(a) = \mathrm{Pre}_{x_1}(b) = 1$, $\mathrm{Post}_{x_1}(a) = 2$, $\mathrm{Post}_{x_1}(b) = 0$ and $m_0(x_1) = 1$.

Consider the token flow functions y_1 and y_2 of L_2 shown in Figure 4. While y_1 defines a region with $\mathrm{Pre}_{y_1}(a) = \mathrm{Post}_{y_1}(a) = m_0(y_1) = 1$, y_2 does not because $Out_{\mathrm{lpo}^0_{3,n}}(v_0, y_2) = n$ for $n \in \mathbb{N}$.

By the definition of regions, we can associate a p/t-net to L by considering regions as places. By construction, each LPO of L is an execution of the associated p/t-net, but in general the associated p/t-net has more executions as specified.

Definition 15 (associated p/t-net). *We denote $\mathcal{R}_L$ the set of non-trivial regions of L. Denote $P = \{p_r \mid r \in \mathcal{R}_L\}$, T the set of labels of L, $W((p_r, l_L(v))) = \mathrm{Pre}_r(v)$ and $W((l_L(v), p_r)) = \mathrm{Post}_r(v)$ for $l_L(v) \in T$ and $p_r \in P$, $F = \{(x,y) \mid W((x,y)) > 0\}$, and $m_0(p_r) = m_0(r)$ for $p_r \in P$. Then the p/t-net (N_L, m_L), $N_L = (P, T, F, W)$, we call the* p/t-net associated to L.

Example 7. The left part in Figure 5 shows some (of the infinite many) regions of L_2. The right part shows (a part of) the p/t-net associated to L_2 with the places $p_i = p_{z_i}$, $i = 1, 2, 3$.

Of course, the associated p/t-net (N_L, m_L) is a candidate for a p/t-net satisfying $\mathcal{L}(L) = \mathfrak{Lpo}(N_L, m_L)$. Observe that $P = \emptyset$ (i.e. $\mathcal{R}_L = \emptyset$) is possible. In this case the associated p/t-net would have an unrestricted behavior and trivially $\mathcal{L}$ is part of this behavior.

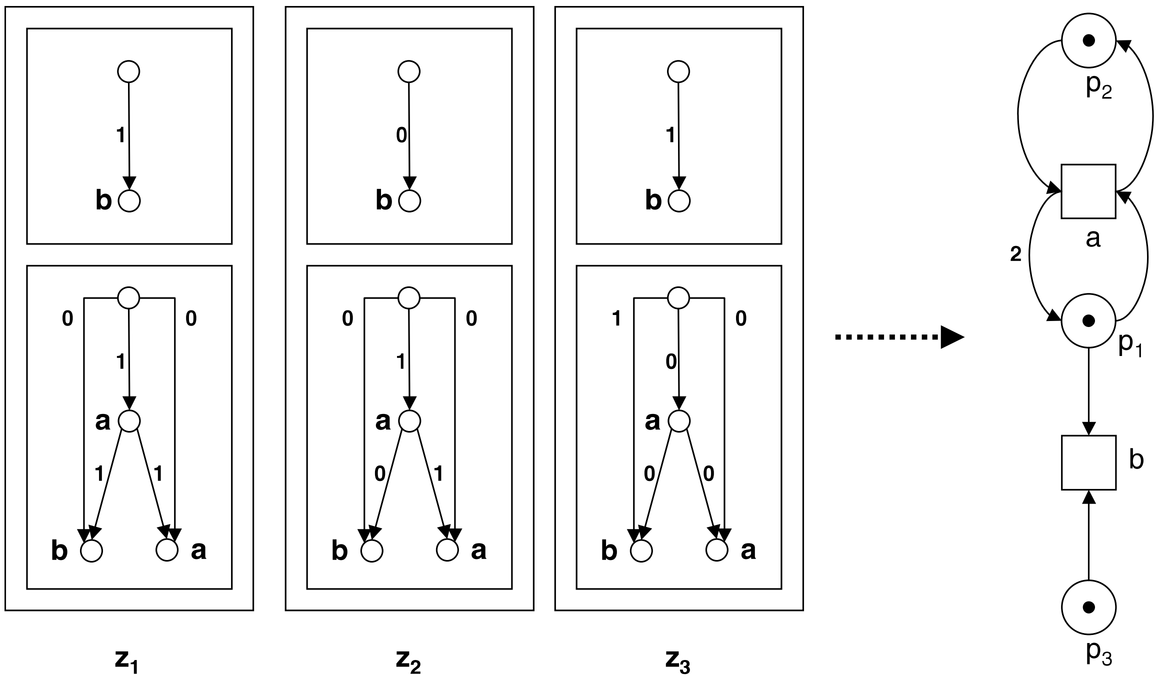

Fig. 5. Some regions of L_2 with corresponding places of the associated p/t-net

In particular, if ${}^{\bullet}t = \emptyset$ for a transition t of (N_L, m_L), (N_L, m_L) does not satisfy $\mathcal{L}(L) \supseteq \mathfrak{Lpo}(N_L, m_L)$. The following Lemma tells us that (N_L, m_L) always satisfies $\mathcal{L}(L) \subseteq \mathfrak{Lpo}(N_L, m_L)$.

Lemma 1. *Let* lpo $\in L$ *and* (N_L, m_L)*,* $N_L = (P, T, F, W)$*, be the p/t-net associated to L. Then* [lpo] *is an execution of* (N_L, m_L)*.*

Proof. We can assume without loss of generality that $P \neq \emptyset$. According to Theorem 2, it is enough to show that lpo $= (V, \prec, l)$ fulfills the token flow property w.r.t (N_L, m_L). In fact, for each place $p_r \in P$ the token flow function $r|_{\prec^\circ}$ fulfills the properties $(\mathbf{IN})$ and $(\mathbf{OUT})$ according to $(\mathfrak{Tff})$ and $(\mathbf{R})$. $\square$

The main theorem reads:

Theorem 3. *There is a marked p/t-net* (N, m_0) *with* $\mathcal{L}(L) = \mathfrak{Lpo}(N, m_0)$ *if and only if* $\mathcal{L}(L) = \mathfrak{Lpo}(N_L, m_L)$ *for the marked p/t-net* (N_L, m_L) *associated to L.*

We prove the theorem in the next subsection.

3.4 Proof of the Main Result

The "if"-part is obvious. To prove the "only if"-part, let (N, m_0), $N = (P, T, F, W)$, be a p/t-net satisfying $\mathcal{L}(L) = \mathfrak{Lpo}(N, m_0)$. It is enough to show that for each $p \in P$ there is a region $r = r(p)$ of L such that for each transition t

(i) $\text{Pre}_r(t) = W((p, t))$,
(ii) $\text{Post}_r(t) \leqslant W((t, p))$,
(iii) $m_0(r) \leqslant m_0(p)$.

Namely, in this case each token flow function of some LPO, which satisfies ($\mathbf{IN}$) and ($\mathbf{OUT}$) w.r.t. p_r, also satisfies ($\mathbf{IN}$) and ($\mathbf{OUT}$) w.r.t. p. In other words, then each LPO which is an execution of (N_L, m_L) is also an execution of (N, m_0), i.e. $\mathcal{L}(L) \subseteq \mathfrak{Lpo}(N_L, m_L) \subseteq \mathfrak{Lpo}(N, m_0) = \mathcal{L}(L)$. The construction of such regions $r(p)$ is shown through the following Lemmata.

Fix a place $p \in P$. For the construction of $r(p)$ we use that for each $\mathrm{lpo} \in L$ there is a token flow function x_p^{lpo} of lpo fulfilling ($\mathbf{IN}$) and ($\mathbf{OUT}$) w.r.t. p (since each $\mathrm{lpo} \in L$ is enabled w.r.t. (N, m_0) according to $\mathcal{L}(L) = \mathfrak{Lpo}(N, m_0)$). According to remark 1, L can be decomposed (through an appropriate equivalence relation) into sets X of LPOs , which have pairwise disjoint node sets and satisfy property

(Seq) Each two LPOs $\mathrm{lpo}, \mathrm{lpo}' \in X$ are sequentializations of prefixes of another LPO $\mathrm{lpo}'' \in X$.

We first construct $r(p)$ for each such maximal subset X of L. Each such set X obviously satisfies additionally the properties (L1)-(L5). The further argumentation is based on the "representation" of X by an appropriate sequence of minimal LPOs which increases w.r.t. the prefix ordering:

Lemma 2. *Let X be a set of LPOs satisfying (L1) - (L5) and (Seq). Then there is a sequence of LPOs* $(\mathrm{lpo}_n)_{n\in\mathbb{N}} \subseteq X$*, such that*

(i) lpo_n *is prefix of* lpo_{n+1},
(ii) lpo_n *is minimal in X, and*
(iii) each LPO in X is a sequentialization of a prefix of some lpo_n.

Proof. Observe that each two LPOs $\mathrm{lpo}, \mathrm{lpo}' \in X$ are sequentializations of prefixes of another LPO $\mathrm{lpo}'' \in X$. Order the set of all nodes of LPOs in X inductively as follows:

- As node v_1 choose an arbitrary node which is minimal for one arbitrary LPO in X.
- Suppose the nodes $v_1, \ldots, v_n$ are chosen. Remove the nodes $v_1, \ldots, v_n$ from all LPOs in X and denote X_n the set of all those modified LPOs. As node v_{n+1} choose an arbitrary node which is minimal for one arbitrary LPO in X_n.

Note that for each $n \in \mathbb{N}$ there are LPOs in X with node set $\{v_1, \ldots, v_n\}$ (according to the chosen ordering and (L5)), and that there is in fact a unique minimal one (since two different LPOs with the same node set are sequentializations of prefixes of some other LPO). Let lpo_n be this minimal LPO with node set $\{v_1, \ldots, v_n\}$. Obviously, lpo_n is then a prefix of lpo_{n+1}, since lpo_{n+1} restricted to the node set $\{v_1, \ldots, v_n\}$ is also in X. Clearly, every LPO in X is a sequentialization of a prefix of some lpo_n. □

If the LPOs of all such $X \subseteq L$ have "consistent" token flow functions, we can construct a region of L:

Lemma 3. *Let* $\mathrm{Pre}, \mathrm{Post} : T \to \mathbb{N}$ *and* $(\mathrm{lpo}_n)_{n\in\mathbb{N}}$ *be a sequence of LPOs such that*

(i) The set of labels of each lpo_n *is a subset of T.*
(ii) lpo_n *is prefix of* lpo_{n+1},
(iii) for each n there is a token flow function x of $\mathrm{lpo}_n = (V_n, <_n, l_n)$ *with*

(IN) $In(v', x) = \mathrm{Pre}(a)$ *for every* $v' \in V_n^0$ *with* $l_n(v') = a$,
(OUT) $Out(v', x) \leqslant \mathrm{Post}(a)$ *for every* $v' \in V_n^0$ *with* $l_n(v') = a$.

Then for each $n \in \mathbb{N}$ *there is a token flow function* x_n *of* lpo_n *satisfying* $(\mathbf{IN})$ *and* $(\mathbf{OUT})$*, such that* $x = \lim_{n\to\infty} x_n$ *is a region of the set* X *of all sequentializations of prefixes of some* lpo_n.

Proof. Choose the token flow functions x_n inductively as follows:

- Let x_1 be an arbitrary token flow function of lpo_1 satisfying $(\mathbf{IN})$ and $(\mathbf{OUT})$.
- Choose a token flow function x_{n+1} of lpo_{n+1} satisfying $(\mathbf{IN})$ and $(\mathbf{OUT})$ such that m is maximal with $x_{n+1}|_{<_m} = x_n|_{<_m}$.

We first show that the sequence of functions $(x_n)_{n\in\mathbb{N}}$ converges pointwise, i.e. on every edge. For this, we fix $k \in \mathbb{N}$ and show that there is $n(k) \in \mathbb{N}$ such that $\forall n, m \geqslant n(k) : x_n|_{<_k} = x_m|_{<_k}$. That means that the sequence $(x_n)_{n\in\mathbb{N}}$ becomes stationary on every lpo_k. This can be seen as follows: The set of all token flow functions of lpo_k fulfilling $(\mathbf{IN})$ and $(\mathbf{OUT})$ is finite, since there are only finitely many arcs and the value of such a token flow function on an arc is bounded through $(\mathbf{IN})$ and $(\mathbf{OUT})$ by $\max(\max\{\mathrm{Pre}(a) \mid a \in T\}, \max\{\mathrm{Post}(a) \mid a \in T\})$. The functions $y_n = x_n|_{<_k}$ are token flow function of lpo_k fulfilling $(\mathbf{IN})$ and $(\mathbf{OUT})$ for each $n \geqslant k$. Assume that $y_i \neq y_{i+1}$ and $y_i = y_j$ for $k \leqslant i < j$. Then $z_{i+1} = x_j|_{<_{i+1}}$ is a token flow function of lpo_{i+1} fulfilling $(\mathbf{IN})$ and $(\mathbf{OUT})$ with $z_{i+1}|_{<_k} = y_j = y_i = x_i|_{<_k}$. This contradicts the choice of x_{i+1} since $x_{i+1}|_{<_k} = y_{i+1} \neq y_i = x_i|_{<_k}$. Thus, y_n can change only finitely often.

Thus, x is well-defined. Assuming $x_n = 0$ on every edge not in $<_n$, we get that x is a token flow function of X. To see this, let $\mathrm{lpo} = (V, <, l), \mathrm{lpo}' = (V', <', l') \in X$ and $v \in V^0$ and $v' \in (V')^0$ with $l(v) = l'(v')$, lpo be sequentialization of a prefix of lpo_k and lpo' be sequentialization a prefix of lpo_m. Then, since all x_n fulfill $(\mathbf{IN})$:

$$\begin{aligned} In_{\mathrm{lpo}^0}(v, x) &= \lim_{n\to\infty} In_{\mathrm{lpo}^0}(v, x_n) \\ &= \lim_{n\to\infty} In_{\mathrm{lpo}_k^0}(v, x_n) \overset{(\mathbf{IN})}{=} \mathrm{Pre}(l(v)) \\ &\overset{(\mathbf{IN})}{=} \lim_{n\to\infty} In_{\mathrm{lpo}_m^0}(v', x_n) = In_{(\mathrm{lpo}')^0}(v', x). \end{aligned}$$

Observe here that $x_n|_{<_k}$ is a token flow function of lpo_k for $n \geqslant k$. Then x fulfills $(\mathfrak{Tff})$, i.e. is a token flow function. Finally we get that x is even a region: Let $\mathrm{lpo} = (V, <, l) \in X$ be a prefix of a sequentialization of lpo_k. It follows for $v \in V$, since all x_n fulfill $(\mathbf{OUT})$:

$$\begin{aligned} Out_{\mathrm{lpo}^0}(v, x) &= \lim_{n\to\infty} Out_{\mathrm{lpo}^0}(v, x_n) \\ &\leqslant \lim_{n\to\infty} Out_{\mathrm{lpo}_k^0}(v, x_n) \overset{(\mathbf{OUT})}{\leqslant} \mathrm{Post}(l(v)). \end{aligned}$$

That means x fulfills $(\mathbf{R})$, i.e. is a region of X. □

For each maximal subset $X \subsetneq L$ there is a sequence $(\mathrm{lpo}_n)_{n\in\mathbb{N}} \subsetneq X$ satisfying the properties stated in Lemma 2 and fulfilling the preconditions of Lemma 3 with $\mathrm{Pre}(l(v)) = W((p, l(v)))$ and $\mathrm{Post}(a) = W((l(v), p))$ by construction. By Lemma 3 there is an appropriate region for each such X, Moreover, BPOs in different such sets X have disjoint node sets. Therefore we can construct a searched region $r(p)$ as the union of all regions of such sets X.

3.5 Relationship to Other Definitions of Regions

In this section we establish the relationship between regions of LPO-specifications and other definitions of regions. For this one has to distinguish between regions of languages and regions of graphs. Whereas in the case of languages it is required that the synthesized p/t-net produces the given language, in the case of graphs it is required that state-graph of the synthesized p/t-net and the given graph are isomorphic, what is a substantial stronger requirement. The regions presented in this paper are language based, since we distinguish different executions by LPOs with disjoint node sets (no matter whether they have a common prefix or not). Another definition of regions which is based on languages and allows the specification of concurrent events is that for trace languages. Therefore, we exemplarily compare our definition of regions to that one for trace languages as presented in [17].

We recall the definition of trace semantics of a p/t-net as given in [17]. A *(generalized) concurrency alphabet* is a pair (A, I), where A is a finite alphabet, and $I \subseteq (\mathbb{N}^A)^+ \times \mathbb{N}^A$ is an *independence relation*. For a concurrency alphabet (A, I) we denote $\rho \sim_I \rho'$ if there are $\rho_1, \rho_2 \in (\mathbb{N}^A)^+$ and $u, v, u', v' \in \mathbb{N}^A$, such that $\rho = \rho_1 uv \rho_2$ and $\rho' = \rho_1 u'v' \rho_2$, $u + v = u' + v'$ and $(\rho_1, u + v) \in I$, $\equiv_I = (\sim_I)^*$, and $[\rho]_I = \{\rho' \mid \rho \equiv_I \rho'\}$. $L \subset (\mathbb{N}^A)^+$ is called *consistent* (w.r.t. (A, I)), if $\forall \rho \in L : [\rho]_I \subseteq L$. A *trace language* (w.r.t. (A, I)) is a triple (L, A, I), where L is consistent (w.r.t. (A, I)).

For a trace language $Tr = (L, A, I)$, a *region* of Tr is a function $r : L \cup A \to \mathbb{N} \cup (\mathbb{N} \times \mathbb{N})$ satisfying the following conditions for $\rho \in L$, $a \in A$ and $u \in \mathbb{N}^A$:

(TR1) $r(\rho) \in \mathbb{N}$ and $r(a) = (\mathrm{Pre}_r(a), \mathrm{Post}_r(a)) \in \mathbb{N} \times \mathbb{N}$
(TR2) If $\rho u \in L$ then $r(\rho) \geqslant \sum_{a\in A} u(a)\mathrm{Pre}_r(a)$, and
(TR3) If $\rho u \equiv \rho'$ then $r(\rho') = r(\rho) + \sum_{a\in A} u(a)(\mathrm{Post}_r(a) - \mathrm{Pre}_r(a))$.

The set of all regions is denoted by R_{Tr}. We can now associate a p/t-net (N, m_0) to Tr with $N = (R_{Tr}, A, F, W)$ via $W(r, a) = \mathrm{Pre}_r(a)$, $W(a, r) = \mathrm{Post}_r(a)$ and $m_0(r) = r(\underline{0})$, where $\underline{0}$ denotes the empty step sequence.

A trace language $Tr = (L, A, I)$ is called *PN trace language*, if the following axioms are satisfied: (PN1) $L \neq \emptyset$, (PN2) $\rho u \in L \Rightarrow \rho \in L$, (PN3) $\rho u \in L \Rightarrow (\rho, u) \in I$ and (PN4) $\rho \in L \Rightarrow (\forall r \in R_{Tr} : r(\rho) \geqslant \sum_{a\in A} u(a)\mathrm{Pre}_r(a) \Rightarrow \rho u \in L)$. If a trace language $Tr = (L, A, I)$ satisfies (PN1) - (PN4), then the associated p/t-net can be shown to have L as its set of step sequences. On the other hand, the trace language of a given p/t-net always fulfills (PN1) - (PN4) (see [17]).

To each LPO-specification it can associated a trace language in a natural way:

Definition 16 (Trace language, of an LPO-specification). *Let L' represent an LPO-specification. Define the trace language* (L, A, I) associated to L' *as follows: L is the* *set of all step sequences of step-linearizations of LPOs in L', $A = l_{L'}(W_{L'})$ and $(\rho, u) \in I \iff \rho u \in L$.*

Obviously such L is consistent.

Lemma 4. *Let L' represent an LPO-specification and let r be a region of L'. Let further (L, A, I) be the trace language associated to L'. Define $r_L : L \cup A \to \mathbb{N} \cup (\mathbb{N} \times \mathbb{N})$ by $r_L(\underline{0}) = \mathrm{Post}_r(v_0)$, $r_L(a) = (\mathrm{Pre}_r(a), \mathrm{Post}_r(a))$ and $r_L(u_1 \dots u_n) = r_L(0) + \sum_{i=1}^n \sum_{a \in A} u_i(a)(\mathrm{Post}_r(a) - \mathrm{Pre}_r(a))$. Then r_L is a region of (L, A, I).*

Proof. We first show (TR2), i.e. that for $\rho \in L$ and $u \in \mathbb{N}^A$:

$$(*) \quad \rho u \in L \Rightarrow r_L(\rho) \geqslant \sum_{a \in A} u(a)\mathrm{Pre}_r(a).$$

Let lpo $\in L'$ such that ρu is the step sequence of a step-linearization of lpo. Denote $\rho = u_1 \dots u_n$. Since lpo is enabled w.r.t. the p/t-net $(N_{L'}, m_{L'})$, $N_{L'} = (P, T, F, W)$, associated to L', it holds for the place p_r corresponding to the region r and each cut C of lpo:

$$(**) \quad m_{L'}(p_r) + \sum_{v \in V \wedge v < C} (W((l(v), p_r)) - W((p_r, l(v)))) \geq \sum_{v \in C} W((p_r, l(v))).$$

Choose $S \subseteq V$ with $|C| = u$ and $|\{v \mid v < C\}| = u_1 + \ldots + u_n$ and remember that $W((p_r, l(v))) = \mathrm{Pre}_r(v)$ and $W((l(v), p_r)) = \mathrm{Post}_r(v)$. Then $(**)$ translates to $(*)$.

This implies moreover $r_L(\rho) \in \mathbb{N}$ for $\rho \in L$. Since obviously $r_L(\underline{0}) \in \mathbb{N}$ and $r(a) \in \mathbb{N} \times \mathbb{N}$ for $a \in A$, we deduce (TR1).

Finally, we show (TR3). For $\rho u = \rho'$ it holds by definition that $r_L(\rho') = r_L(\rho) + \sum_{a \in A} u(a)(\mathrm{Post}_r(a) - \mathrm{Pre}_r(a))$. To show this equation for the general case, it is enough to show $\rho \equiv_I \rho' \Rightarrow r_L(\rho) = r_L(\rho')$. It is even enough to consider the case $\rho \sim_I \rho'$, i.e. $\rho = \rho_1 v w \rho_2$ and $\rho' = \rho_1 v' w' \rho_2$ with $v + w = v' + w'$ and $(\rho_1, v + w) \in I$. The statement follows then from $\sum_{a \in A}(v+w)(a)(\mathrm{Post}_r(a) - \mathrm{Pre}_r(a)) = \sum_{a \in A}(v' + w')(a)(\mathrm{Post}_r(a) - \mathrm{Pre}_r(a))$. □

Lemma 5. *Let L' represent an LPO-specification and (L, A, I) be the trace language associated to L'. Let further $r : L \cup A \to \mathbb{N} \cup (\mathbb{N} \times \mathbb{N})$ be a region of (L, A, I). Then there is a region r' of L' satisfying $\mathrm{Post}_{r'}(v_0) = r(\underline{0})$, $\mathrm{Pre}_{r'}(a) = \mathrm{Pre}_r(a)$ and $\mathrm{Post}_{r'}(a)) \leqslant \mathrm{Post}_r(a)$.*

Proof. It is enough to show that for each lpo $= (V, \prec, l) \in L'$ there is a token flow function x fulfilling $In(v', x) = \mathrm{Pre}_r(a)$ and $Out(v', x) \leqslant \mathrm{Post}_r(a)$ for every $v' \in V^0$. Then analogously as in the proof of Theorem 3 the statement follows.

Assume there is lpo $= (V, \prec, l) \in L'$ such that there is no token flow function x satisfying $(\mathbf{IN})$ and $(\mathbf{OUT})$. Then, by an argumentation in our paper [18] (Lemma 7), there is a cut C of lpo for which $r(\underline{0}) + \sum_{v<C} \mathrm{Post}_r(v) - \sum_{v<C} \mathrm{Pre}_r(v) - \sum_{v \in C} \mathrm{Pre}(v) < 0$. This contradicts (TR2) for the step sequence of an adequate step linearization of lpo. □

4 Conclusion

As already mentioned in the introduction, we consider the presented results as a step towards solving the p/t-net synthesis problem from non-sequential specifications. To this end, the problem and the results must be transformed in two ways. First, it is necessary to represent the possibly infinite set of all regions of an LPO-specification by a finite subset. By this, a finite p/t-net could be assigned to an LPO-specification L with the same behavior as the (possibly infinite) p/t-net associated to L. Second, one needs a finite representation of LPO-specifications. When these problems are solved, finally an effective test of $L = \mathfrak{Lpo}(N', m_0')$ for the marked p/t-net (N', m_0') associated to L has to be developed. We are currently working on these topics and have already found partial solutions. These are not presented in detail due to lack of space. Instead, we shortly give some hints in this subsection.

Concerning the finite representation of LPO-specifications let us first consider the case, where simply the LPO-specification L is finite. Then, of course, the associated p/t-net cannot have an infinite behavior, i.e. it's marking graph cannot contain cycles (see the LPO-specifications from Figure 2 as an example). That means, in a finite representation of an infinite LPO-specification, the LPOs of a representation are only considered as prefixes of runs of the searched p/t-net, and one has additionally to specify desired cycles. This could be done for example by specifying cut-off events of LPOs. If no desired cycles are specified, it is nevertheless possible to detect such regions (places of the associated p/t-net) which prohibit cycles (which are, loosely spoken, those with initial marking different to the final marking of some sub-LPO).

Concerning finite representations of p/t-nets corresponding to LPO-specifications L, it is well known (see e.g. [24]) that there are p/t-nets with an infinite set of places which cannot be represented by p/t-nets with a finite set of places (see also the example below). That means we need a further characterization of LPO-specifications equaling the partial language of executions of p/t-nets with finite set of places. However, if L is finite then also the set of states (represented by L) is finite. In this case one can construct a p/t-net with finite set of places having L as the partial language of its executions. Observe moreover that for the synthesis of elementary nets from LPO-specifications there are only finitely many possible regions.

Example 8. Consider the marked p/t-net $N = (P, T, F, W, m_0)$ defined by $T = \{a, b\}$, $P = \{p_n \mid n \in \mathbb{N}\}$, $W((a, p_n)) = n+1$ and $W((p_n, b)) = n$, and $m_0(p_n) = n-1$ for $n \in \mathbb{N}$. After one occurrence of a, the concurrent step $2b$ is enabled. Then *each* further occurrence of a enables transition b *exactly once*, since for increasing n the places p_n more and more restrict the behavior. This is the reason, why P cannot be replaced by a finite subset, say $P_k = \{p_1, \ldots, p_k\}$ for some $k \in \mathbb{N}$. Namely, in this case after each k occurrences of a the transition b would be enabled twice (not only once). In fact, the places p_n "converge" to the place p with $W((a, p)) = 1$, $W((p, b)) = 1$ and $m_0(p) = 1$. This place can be added to P without further restricting the behavior. On the other side, since b is not enabled in the initial marking, P can also not be equivalently replaced by the place set $P' = \{p\}$ only containing p, since w.r.t. P' the transition b is enabled in the initial marking. The problem is the increasing number of initial tokens in p_n for increasing n. If this number would be bounded (that means $m_0(p) \leqslant k$ for a fixed $k \in \mathbb{N}$), *each* occurrence of a (from the beginning) would enable transition b *exactly once*. The same behavior could be achieved by the finite place set $P'' = \{q\}$ with $W((a,q)) = 1$, $W((q,b)) = 1$ and $m_0(q) = 0$, i.e. in this case there would be a finite representation.

When considering finite LPO-specifications L with some representation $\mathfrak{Lpo}(L)$, there is moreover the following effective test of $L = \mathfrak{Lpo}(N', m_0')$ for the marked p/t-net (N', m_0') associated to L. In fact it must be shown that LPOs not in L are no executions of (N', m_0'). For this, first compute a finite set of regions representing the set of all regions. Then test the following LPOs to satisfy the token flow property w.r.t. all these regions: All LPOs constructed from minimal LPOs of maximal length in $\mathfrak{Lpo}(L)$ by deleting some edge, and all LPOs not in $\mathfrak{Lpo}(L)$ constructed from minimal LPOs in $\mathfrak{Lpo}(L)$ by adding a unique maximal node with some label. Each LPO not in L is also no execution of (N', m_0') if and only if none of these LPOs fulfills the token flow property w.r.t. all of the finite many regions.

References

1. W.M.P. van der Aalst, A.K.A. de Medeiros and A.J.M.M. Weijters. *Genetic Process Mining.* LNCS 3536, pages 48-69, 2005.
2. 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, pages 1128-1142, 2004.
3. E. Badouel and P. Darondeau. *Theory of Regions.* LNCS 1491, pages 529-586, 1998.
4. J. Cortadella, M. Kishinevsky, L. Lavagno and A. Yakovlev. *Deriving Petri Nets for Finite Transition Systems.* IEEE Trans. Computers 47/8, pages 859-882, 1998.
5. P. Darondeau. *Unbounded Petri Net Synthesis.* LNCS 3098, pages 413-438, 2004.
6. J. Desel and W. Reisig. *The Synthesis Problem of Petri Nets.* Acta Inf. 33/4, pages 297-315, 1996.
7. A. Ehrenfeucht and G. Rozenberg. *Partial (Set) 2-Structures. Part I: Basic Notions and the Representation Problem.* Acta Inf. 27/4, pages 315-342, 1989.
8. A. Ehrenfeucht and G. Rozenberg. *Partial (Set) 2-Structures. Part II: State Spaces of Concurrent Systems.* Acta Inf. 27/4, pages 343-368, 1989.
9. J. Esparza and K. Heljanko *Implementing LTL Model Checking with Net Unfoldings.* LNCS 2057, pages 37-56, 2001.
10. L.R. Ford, Jr. and D.R. Fulkerson. Maximal Flow Through a Network. *Canadian Journal of Mathematics* 8, pp. 399–404, 1955.
11. A. Goldberg and S. Rao. Beyond the Flow Decomposition Barrier. *Journal of the ACM* 45/5, pp. 783–797, 1998.
12. U. Goltz and W. Reisig. The Non-Sequential Behaviour of Petri Nets. *Information and Control*, 57(2-3), pp. 125-147, 1983.
13. U. Goltz and W. Reisig. Processes of Place/Transition Nets. LNCS 154, pp. 264-277, 1983.
14. J. Grabowski. On Partial Languages. *Fundamenta Informaticae* IV.2, pp. 428–498, 1981.
15. D. Harel, H. Kugler and A. Pnueli. *Synthesis Revisited: Generating Statechart Models from Scenario-Based Requirements.* LNCS 3393, pages 309-324, 2005.
16. D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine* Springer, 2003.
17. P.W. Hoogers and H.C.M. Kleijn and P.S. Thiagarajan. A trace semantics for Petri nets. *Information and Computation* 117/1, pp. 98–114, 1995.

18. G. Juhás and R. Lorenz and J. Desel. Can I execute my scenario in your net? LNCS 3536, pp. 289–308, 2005.
19. A.V. Karzanov. Determining the Maximal Flow in a Network by the Method of Preflows. *Soviet Math. Doc.* 15, pp. 434–437, 1974.
20. A. Kiehn. On the Interrelationship between Synchronized and Non-Synchronized Behavior of Petri Nets. *Journal Inf. Process. Cybern. EIK* 24, pp. 3 – 18, 1988.
21. J. Klose and H. Wittke. *An Automata Based Interpretation of Live Sequence Charts.* LNCS 2031, pages 512-527, 2001.
22. M. Lettrari and J. Klose. *Scenario-Based Monitoring and Testing of Real-Time UML Models.* LNCS 2185, pages 317-328, 2001.
23. N. Mansurov. *Automatic synthesis of SDL from MSC and its applications in forward and reverse engineering.* Comput. Lang. 27/1, pages 115-136, 2001.
24. M. Mukund. *Petri Nets and Step Transition Systems.* Int. J. Found. Comput. Sci. 3/4, pages 443-478, 1992.
25. M. Nielsen, G. Rozenberg and P.S. Thiagarajan. *Elementary Transition Systems.* Theor. Comput. Sci. 96/1, pages 3-33, 1992.
26. M. Nielsen, G. Rozenberg and P.S. Thiagarajan. *Transition systems, event structures and unfoldings.* Information and Computation 118/2, pages 191-207, 1995.
27. V. Pratt. Modelling Concurrency with Partial Orders. *Int. Journal of Parallel Programming* 15, pp. 33–71, 1986.
28. A. Roychoudhury, P.S. Thiagarajan, T. Tran and V.A. Zvereva. *Automatic Generation of Protocol Converters from Scenario-Based Specifications.* Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS 2004), pages 447-458, 2004.
29. W. Vogler. Modular Construction and Partial Order Semantics of Petri Nets. *LNCS* 625, 1992.