Synthesis of bounded Petri Nets from Prime Event Structures with Cutting Context

Gabriel Juhás[1] and Robert Lorenz[2]

[1] SLOVAK UNIVERSITY OF TECHNOLOGY in Bratislava
Faculty of Electrical Engineering and Information Technology
Ilkovičova 3, 812 19 Bratislava, Slovak Republic
gabriel.juhas@stuba.sk
[2] UNIVERSITY OF AUGSBURG
Department of Applied Computer Science
robert.lorenz@informatik.uni-augsburg.de

Abstract. In this paper we present token flow based synthesis of bounded Petri nets from labelled prime event structures (LPES) associated with a cutting context. For this purpose we use unfolding semantics based on token flows.
Given an infinite LPES represented by some finite prefix equipped with a cutting context and cut-off events it is shown how to synthesize a bounded Petri net, such that the unfolding of the synthesized net preserves common prefixes and concurrency of runs of the LPES. The partial language of this unfolding is the minimal partial language of an unfolding of a Petri net, which includes the partial language of the LPES.
This result extends the class of non-sequential behaviour, for which Petri nets can be synthesized, because finite representations of infinite LPES by a finite prefix equipped with a cutting context and cut-off events are more expressive than finite representations of infinite partial languages by terms.

1 Introduction

Non-sequential Petri net semantics can be classified into unfolding semantics, process semantics, step semantics and algebraic semantics [11]. While the last three semantics do not provide semantics of a net as a whole, but specify only single, deterministic computations, unfolding models are a popular approach to describe the complete behavior of nets accounting for the fine interplay between concurrency and nondeterminism.

To study the behavior of Petri nets primarily two models for unfolding semantics were retained: labelled occurrence nets and event structures. The standard unfolding semantics for the general class of place/transition Petri nets or p/t-nets is based on the developments in [12, 3] (see [7] for an overview) in terms of so called branching processes, which are acyclic occurrence nets having events representing transition occurrences and conditions representing tokens in places. Branching processes allow events to be in conflict through branching conditions. Therefore, branching processes can represent alternative processes simultaneously (processes are finite branching processes without conflict). Branching processes for p/t-nets individualize tokens having the same "history", i.e. several (concurrent) tokens produced by some transition occurrence in the same place are distinguished through different conditions. One can define

a single maximal branching process, called the unfolding of the system, capturing the complete non-sequential branching behavior of the p/t-net.

A problem with unfoldings is that they are infinite whenever the original p/t-nets have infinitely many runs. It turns out that Petri net unfoldings can often be truncated in such a way that the resulting prefixes, though finite, contain full information w.r.t. some behavioral property. Such prefixes are complete for this property. In the case of bounded nets, according to a construction by McMillan [10] a complete finite prefix preserving full information on reachable markings can always be constructed. In the case of bounded nets, the construction of unfoldings and complete finite prefixes is well analyzed and several algorithmic improvements are proposed in literature [4, 8, 6]. The essential feature of the existing unfolding algorithms is the use of cut-off events, beyond which the unfolding starts to repeat itself and so can be truncated without loss of information. In [8] a generalized, parametric setup, called *cutting context* is proposed, in which completeness can be discussed in a uniform and algorithm-independent way.

By restricting the relations of causality and conflict of a branching process to events, one obtains a labelled prime event structure (LPES) [18] underlying the branching process, which represents the causality between events of the branching process. Events not being in conflict define consistency sets, that is, an LPES is a partially ordered set of events (transition occurrences) together with a set of (so called) *consistency sets* [18]. "History-closed" (left-closed) consistency sets correspond to processes and their underlying runs in the unfolding. Thus, event structures are in their nature a branching time model of computation, which enable to specify common history of runs.

In [1] we presented a new unfolding approach, so called token flow unfoldings based on LPES, which avoid the representation of isomorphic processes or even processes with isomorphic runs in many situations. The new unfolding semantics combines LPES with so called token flows, which were developed in [5] for a compact representation of processes. Token flows abstract from the individuality of conditions of a branching process and encode the flow relation of the branching process by natural numbers, which are assigned to the edges of the partially ordered run underlying a branching process for each place. Such a natural number assigned to an edge (e, e') represents the number of tokens produced by the transition occurrence e and consumed by the transition occurrence e' in the respective place. An LPES with assigned token flow information is called a *token flow unfolding* if each process is represented through a left-closed consistency set with assigned token flows corresponding to the process.

Besides their importance as the fundamental model of concurrency, event structures become also interesting to applications: Recently, Dumas in [2] advocates event structures, and in particular labelled prime event structures, as the unifying representation of process models and event logs in the context of process mining. In the Outlook of the paper [2], it is stated that "the use of event structures for process model synthesis would require new techniques to be developed or existing ones to be heavily adapted". The Outlook of the paper [2] also states that "A key challenge is handling repeated behavior." and later continues that "synthesizing a process model from the event structure derived from a log requires being able to detect and isolate repeated behavior." In this paper we propose such new techniques handling repeated behavior adapting our recent

development in the area of synthesis of Petri nets from partial languages [9], which is basically based on token flows.

In section 2 we introduce basic mathematical notions and recall the definitions of LPES and of Petri nets with token flow unfoldings, as they were described in [1]. Further, we introduce complete finite prefixes of token flow unfoldings defined in [1].

Given a labelled prime event structure, to handle repeated behaviour, we propose to use cutting contexts and to equip the labelled prime event structure with cut-off events. Then, given an infinite labelled prime event structure represented by some finite prefix equipped with a cutting context and cut-off events it is shown in section 3 how to synthesize a bounded Petri net, while preserving the shared history and concurrency of runs. This result extends the class of non-sequential behaviour, for which Petri nets can be synthesized, because finite representation of infinite labelled prime event structures by finite prefix equipped with cutting context and cut-off events is more expressive than finite representation of infinite partial languages by terms as given in [9].

2 Token Flow Unfolding Semantics of P/T-nets

In this section we recall the definitions of place/transition Petri nets, the unfolding semantics based on token flows as they were described in [1], we recall the theory of region based synthesis [9], and the theory of complete prefixes of unfoldings proposed in [8]. We begin with some basic mathematical notations.

2.1 Basic Notions

We use $\mathbb{N}$ to denote the *nonnegative integers*. A *multi-set* over a set A is a function $m : A \to \mathbb{N}$. For an element $a \in A$ the number $m(a)$ determines the number of occurrences of a in m. Given a binary relation $R \subseteq A \times A$ over A, the symbol R^+ denotes the *transitive closure* of R. A *directed graph* is a tuple $G = (V, \to)$, where V is its set of *nodes* and $\to \subseteq V \times V$ is its set of *arcs*. As usual, given a binary relation $\to$, we write $v \to w$ to denote $(v, w) \in \to$. In this case v is called *pre-node* of w amd w is called *post-node* of v. For $v \in V$ we denote by ${}^\bullet v = \{w \in V \mid w \to v\}$ the *preset* of v, and by $v^\bullet = \{w \in V \mid v \to w\}$ the *postset* of v.

A *partial order* is a directed graph $(V, <)$, where $< \subseteq V \times V$ is an irreflexive and transitive binary relation. In the context of this paper, a partial order is interpreted as an "earlier than"-relation between events. A node v is called *maximal* if $v^\bullet = \emptyset$, and *minimal* if ${}^\bullet v = \emptyset$. A subset $W \subseteq V$ is called *left-closed* if $\forall v, w \in V : (v \in W \wedge w < v) \implies w \in W$. For a left-closed subset $W \subseteq V$, the partial order $(W, < |_{W \times W})$ is called *prefix* of $(V, <)$, defined by W. The *left-closure* of a subset W is given by the set $W \cup \{v \in V \mid \exists w \in W : v < w\}$. 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 $<_1 \subseteq <_2$. By $<_s \subseteq <$ we denote the smallest subset $<_s$ of $<$ which fulfils $(<_s)^+ = <$, called the *skeleton of* $<$.

A *labelled partial order* (LPO) is a triple $(V, <, l)$, where $(V, <)$ is a partial order, and l is a *labelling function* on V. We use all notations defined for partial orders also for LPOs. LPOs are used to represent partially ordered runs of Petri nets. Such runs are distinguished only up to isomorphism [1].

2.2 Petri Nets

A *net* is a triple $N = (P, T, F)$, where P is a set of *places*, T is a set of *transitions*, satisfying $P \cap T = \emptyset$, and $F \subseteq (P \cup T) \times (T \cup P)$ is a *flow relation*. Places and transitions are called the *nodes* of N.

Definition 1 (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 with finite sets of places and transitions, and* $W : F \to \mathbb{N} \setminus \{0\}$ *is a* weight function*. A* marking *of a p/t-net* $N = (P, T, F, W)$ *is a function* $m : P \to \mathbb{N}$*. 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 extend the weight function W to pairs of net elements $(x, y) \in (P \times T) \cup (T \times P)$ satisfying $(x, y) \notin F$ by $W((x, y)) = 0$. A transition $t \in \mathbb{N}$ is *enabled to occur in a marking* m of N if $\forall p \in P : \ m(p) \geq W((p, t))$. If t is enabled to occur in a marking m, then its *occurrence* leads to the new marking m' defined by $m'(p) = m(p) - W((p, t)) + W((t, p))$ for all $p \in P$.

2.3 Prime Event Structures

A prime event structure (PES) consists of a set of events, a partial order representing an "earlier than"-relation between events and a set of so called consistency sets, where left-closed consistency sets represent single runs. Events which are never in the same consistency set are assumed to be in conflict and to belong to alternative runs. Labels of events represent action names.

Definition 2 (Prime event structure). *A* prime events structure (PES) *is a triple* $\mathrm{pes} = (E, Con, \prec)$ *consisting of a set* E *of events, a partial order* $\prec$ *on* E *and a set* Con *of finite subsets of* E *satisfying:*

- $\forall e \in E : \ \{e' \mid e' \prec e\}$ *is finite.*
- $\forall e \in E : \ \{e\} \in Con$*.*
- $Y \subseteq X \in Con \Longrightarrow Y \in Con$*.*
- $((X \in Con) \wedge (\exists e' \in X : \ e \prec e')) \Longrightarrow (X \cup \{e\} \in Con)$*.*

A consistent subset *of* E *is a subset* X *satisfying* $\forall Y \subseteq X$*,* Y *finite* $: \ Y \in Con$*. The conflict relation* $\#$ *between events of* pes *is defined by* $e \# e' \Leftrightarrow \{e, e'\} \notin Con$*.*

A tuple $(E, Con, \prec, l)$*, where* $(E, Con, \prec)$ *is a PES and* l *is a labelling function on* E*, is called* labelled prime event structure (LPES)*.*

Notice that the conflict relation expresses that the respective events are always in conflict. A PES definition using a binary conflict relation instead of consistency sets can also be found in the literature [12]. The definition according to [18] used in this paper is more expressive. Imagine a trivial example with two PES $\mathrm{pes} = (E, Con, \prec)$, $\mathrm{pes}' = (E, Con', \prec)$ differing just in consistency sets, with $E = \{a, b, c\}$, $\prec = \emptyset$, Con given by all subsets of E except $\{a, b, c\}$ and Con' given by all subsets of E including $\{a, b, c\}$. Obviously, the conflict relation for both pes and pes' coincide: It is empty. But intuitively, pes represents a system with three different runs, where in each run at most

two events from three will occur in parallel, but never three events can occur in parallel in a run. On the contrary, pes' represents a system where all three events can occur in parallel in one single run. This difference cannot be captured by a binary conflict relation.

As LPOs, LPES are distinguished only up to isomorphism [1]. An LPES, where its whole set of events E forms a consistent set, we interpret as an LPO, i.e. in this case we omit the set of consistency sets Con.

We denote the set of left-closed consistency sets of a LPES $\mathrm{lpes} = (E, Con, \prec, l)$ by $Con_{pre} \subseteq Con$. If $C \in Con_{pre}$ is a left-closed consistency set, then $\mathrm{lpo}_C = (C, \prec |_{C\times C}, l|_C)$ is an LPO which we interpret as a run given by lpes. We define *partial language corresponding to* lpes as the sequentialization closure of $\{\mathrm{lpo}_C \mid C \in Con_{pre}\}$. It is denoted by $\mathcal{L}(\mathrm{lpes})$. For every event $e \in E$, the finite left-closed consistency set $[e] = \{f \mid f \prec e\}$ is called a *local* consistency set.

For a set of events E' and $C \in Con_{pre}$ we denote by $C \oplus E'$ the fact that $C \cup E' \in Con_{pre}$ and $C \cap E' = \emptyset$. If $E' = \{e\}$, we also write $C \oplus e$ to denote $C \oplus \{e\}$. Such an E' is a *suffix* of C, and $C \oplus E'$ is an *extension* of C.

Finally, we introduce a new notion of history and concurrency preservation of LPES.

Definition 3 (History and concurrency preservation). *Let* $\mathrm{lpes} = (E, Con, \prec, l)$ *and* $\mathrm{lpes}' = (E', Con', \prec', l')$ *be two LPES. If there exists a function* $b : E \to E'$ *such that for each left-closed consistency set* C *of* lpes *there holds that* $b(C)$ *is a left-closed consistency set of* lpes' *and* $b|_C$ *defines an isomorphism between* lpo_C *and a sequentialization of* $lpo_{b(C)}$*, then we say that* lpes' preserves common prefixes and concurrency of runs of lpes.

In particular, if lpes' preserves common prefixes and concurrency of runs of lpes, then the partial language of lpes' includes the partial language of lpes. Basically, the existence of function b also means, that whenever some runs of $lpes$ share a common prefix with events in their intersection equal to X, then their b-images in lpes' share at least the prefix with events given by $b(X)$. The runs of lpes are sequentializations of their b-images, i.e. at least the same amount of concurrency is preserved in each run.

Later on we will show that, given an LPES lpes as specification, the token flow unfolding of the synthesized net (as defined in the following subsection) preserves common prefixes and concurrency of runs of lpes.

2.4 Token Flow Unfolding of Petri Nets

In this section we recall one of the unfolding semantics of p/t-nets based on token flows from [1]. Let $\mathrm{lpes} = (E, Con, \prec, l)$ be an LPES and $N = (P, T, W, m_0)$ be a marked p/t-net. We want to interpret lpes as a model of the behavior of N, where the events in E represent transition occurrences. A *token flow function* $x : \prec \to \mathbb{N}^P$ is a function assigning multisets of places of N to the arcs of lpes. For an arc (e, e') between transition occurrences e and e' the multiset $x(e, e')$ is intended to represent the token flow between these transition occurrences, that is to represent for each place the number of tokens which are produced by e and then consumed by e'.

For a token flow function x, a consistency set $C \in Con_{pre}$ and an event $e \in C$ we denote

– $IN^x(e) = \sum_{e' \prec e} x(e', e)$ the *x-intoken flow of e.*
– $OUT^x_C(e) = \sum_{e \prec e',\, e' \in C} x(e, e')$ the *x-outtoken flow of e w.r.t. C.*

A prime token flow event structure is an LPES together with a token flow function. Since equally labelled events represent different occurrences of the same transition, they are required to have equal intoken flow. Since not all tokens which are produced by an event are consumed by further events, there is no analogous requirement for the outtoken flow. It is assumed that there is a unique initial event producing the initial marking.

Definition 4 (Prime token flow event structure). *A* prime token flow event structure over T *is a pair* (lpes, x)*, where* $\mathrm{lpes} = (E, Con, \prec, l)$ *is an LPES with a unique minimal event* e_{init} *w.r.t.* $\prec$ *with* $l(e_{init}) \neq l(e)$ *for all* $e \neq e_{init}$ *and* $l(E \setminus \{e_{init}\}) \subseteq T$ *and* $x : \prec \rightarrow \mathbb{N}^P$ *is a* token flow function *satisfying* $\forall e, e' :\ l(e) = l(e') \Longrightarrow IN^x(e) = IN^x(e')$.

Two events are called strongly identical (w.r.t. a token flow function), if they are labelled by the same action name and depend on the same events with identical token flow. In [1] we showed that strong identical events that are in conflict always lead to isomorphic processes, i.e. omitting strong identical events lead to a more compact representation of behavior without loss of information.

Definition 5 (Strongly identical events). *Let* $((E, Con, \prec, l), x)$ *be a prime token flow event structure. Two events* $e, e' \in E$ *fulfilling* $(l(e) = l(e')) \wedge ({}^\bullet e = {}^\bullet e') \wedge (\forall f \in {}^\bullet e : x(f, e) = x(f, e'))$ *are called strongly identical.*

A token flow unfolding of a marked p/t-net is a prime token flow event structure, in which intoken and outtoken flows are consistent with the arc weights resp. the initial marking of the net within each left-closed consistency set. It is also required that the token flow on a skeleton arc may not be zero, that means only real causal dependencies are represented in an unfolding.

Definition 6 (Token flow unfolding). *Let* (N, m_0)*,* $N = (P, T, F, W)$*, be a marked p/t-net. A* token flow unfolding of (N, m_0) *is a prime token flow event structure* (lpes, x) *over* T*,* $\mathrm{lpes} = (E, Con, \prec, l)$*, satisfying:*

– *(U_{in}):* $\forall e \neq e_{init},\ \forall p \in P :\ IN^x(e)(p) = W(p, l(e))$.
– *(U_{out}):* $\forall C \in Con_{pre},\ \forall e \in C \setminus \{e_{init}\},\ \forall p \in P :\ OUT^x_C(e)(p) \leqslant W(l(e), p)$.
– *(U_{init}):* $\forall C \in Con_{pre},\ \forall p \in P :\ OUT^x_C(e_{init})(p) \leqslant m_0(p)$.
– *(U_{min}):* $\forall (e, e') \in \prec_s :\ (\exists p \in P :\ x(e, e')(p) \geqslant 1)$.
– *(U_{id}): There are no strongly identical events e, e' (w.r.t. x) satisfying* $\{e, e'\} \notin Con$.

As LPOs and LPES, token flow unfoldings are distinguished only up to isomorphism ([1]).

Example 1. Figure 1 shows a marked p/t-net (N, m_0) (right side) together a finite token flow unfolding (left side). As usual, places of a p/t-net are drawn as circles and transitions as big squares with transition names shown inside. Markings are represented by

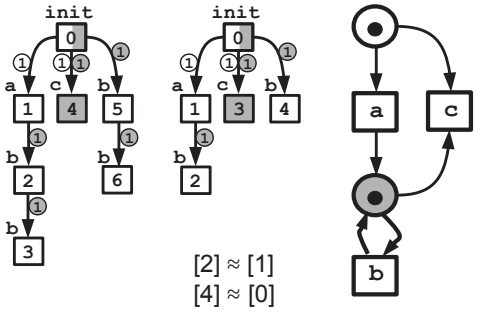

Fig. 1. A marked p/t-net (right side) together with a finite prefix of its maximal token flow unfolding (left side) and the canonical prefix of its maximal token flow unfolding w.r.t. $\approx = \approx_{mar}$ and $\lhd = \subset$ (in the middle).

dots inside places. Arc weights are represenetd by numbers assigned to arcs, where the arc weight 1 is not shown.

Events of an LPES are drawn as small squares with event names shown inside and event labels shown outside. Note that usually not all transitive arcs are shown, but only those establishing the partial order.

Different maximal consistency sets are distinguished by different colors. The token flow unfolding of figure 1 has the two maximal consistency sets $C = C_{grey} = \{0, 4\}$ (color grey) and $C' = C_{white} = \{0, 1, 2, 3, 5, 6\}$ (color white).

The token flow function is graphically represented by numbers in small circles of different colors assigned to arcs. The colors belong to the places of the net. In figure 1 there is a white place and a grey place. Token flows of value 0 are not shown.

It is easy to see, that in figure 1 (left side) the defining properties of token flow unfoldings are satisfied, for example: If $p = p_{grey}$ is the grey place then $IN^x(3)(p) = 1 = W((p, b))$, $OUT^x_{C'}(3)(p) = 0 < 1 = W((b, p))$ and $OUT^x_C(0)(p) = 1 = m_0(p)$.

Note that the maximal unfolding (defined below) of (N, m_0) is infinite and contains the shown unfolding as a finite prefix.

There exists a maximal (in general infinite) token flow unfolding $Unf_{max}(N, m_0)$ (w.r.t. given prefix relation), which is unique up to isomorphism. If (lpes, x) is the maximal token flow unfolding of a Petri net, then $\mathcal{L}(\mathrm{lpes})$ is called Petri net unfolding partial language. For each finite left-closed consistency set C of Unf_{max}, the LPO lpo_C is a run of N and for each run lpo of N there is a left-closed consistency set C of Unf_{max} with $\mathrm{lpo} = \mathrm{lpo}_C$.

For each finite left-closed consistency set C of Unf_{max} the multi-set of places $Mark(C) = m_0(\cdot) + \sum_{e \in C}(W(l(e), \cdot) - W(\cdot, l(e))$ is a reachable marking of (N, m_0), called the *final marking* of C. Every final marking in Unf_{max} is reachable in (N, m_0), and every reachable marking is a final marking in Unf_{max}.

2.5 Region based Synthesis

In this paper we follow the traditional region based synthesis approach (see for example [9]), which we briefly describe now.

Given a specification of the behavior of a system based on runs over a finite set of action names, the action names are used as the transitions of the searched Petri net. It remains to find a suitable finite set of places, each place with initial marking and connections via arcs and arc-weights to all transitions. Places are found in a three-step-approach.

First the set of feasible places is defined. A place is feasible, if it does not prohibit some of the specified behavior. The set of feasible places usually is infinite.

Second so called regions of the behavioral specification are defined as non-negative integral solutions of a linear homogenous equation system $\mathbf{A} \cdot \mathbf{x} = \mathbf{0}$, such that each region defines a feasible place and each feasible place is defined by a region.

Adding all feasible places to the net leads to the so called *saturated net*, which has the following properties:

- Its behavior includes the specified behavior.
- There is no net with less behavior having the first property.

Third, for practical application, it remains to define a finite representation, i.e. a finite set of regions such that the corresponding finite net has the same behavior as the saturated net. One possibility of a finite representation, used in this paper, is to use the fact that the set of solutions of a linear homogenous equation systems always has a finite set of integral basis solutions [15]. There is a well-established mathematical theory to compute these basis solutions. Since each solution can be represented as a non-negative linear combination of the basis solutions, the finite set of basis solution is a finite representation, calles *basis representation*.

Summarized, in order to get a concrete synthesis algorithm for a behavioral specification based on runs over action names, it is enough to

- define feasible places.
- define regions as solutions of a linear homogenous equation system, and
- show that each regions defines a feasible place and each feasible place is defined by a region.

2.6 Complete Prefixes of Unfoldings

Since the maximal unfolding is infinite whenever the original net has infinite behavior, there are approaches for constructing finite and complete prefixes. The essential feature of the existing unfolding algorithms computing finite, complete prefixes is the use of cutoff events, beyond which the unfolding can be truncated without loss of information. In [8] a parametric setup, called *cutting context*, is proposed in which completeness and cutoff events can be discussed in a uniform, general and algorithm-independent way.

In the following, we briefly recall the notions which are relevant in the context of this paper. Let Unf_{max} be the maximal token flow unfolding of a marked p/t-net (N, m_0). We denote by Con and Con_{pre} the sets of consistency sets and left-closed consistency sets of Unf_{max}. A *cutting context* is a triple $\Theta = (\approx, \lhd, \{C_e\}_{e \in E})$, where:

1. $\approx$ is an equivalence relation on Con_{pre}, capturing the information which is intended to be retained in a complete prefix. In the standard case $\approx = \approx_{mar}$, this is the set of reachable markings, i.e. $C \approx_{mar} C'$ if $Mark(C) = Mark(C')$.

2. $\lhd$ is a so called *adequate order* on Con_{pre} which refines $\subset$. All $\lhd$-minimal left-closed consistency sets in each equivalence class of $\approx$ will be preserved in a complete prefix (see [10, 4] for concrete choices of $\lhd$).
3. $\approx$ and $\lhd$ are preserved by finite extensions $C \oplus E$ of left-closed consistency sets C by suffixes E.
4. $\{C_e\}_{e \in E}$ is a family of subsets of Con_{pre} specifying the set of left-closed consistency sets used to decide whether an event can be designated as a cutoff event. In the standard case, C_e contains the local consistency sets of Unf_{max}.

Roughly spoken, a prefix of Unf_{max} is *complete*, if each equivalence class w.r.t. $\approx$ is represented once in it. Hence, for the relation $\approx_{mar}$, each reachable marking is represented by a left-closed consistency set of a complete prefix.

With these notions, cutoff events can be defined in a "static way" without referring to a specific algorithm building the unfolding. The set $CutOff$ of static cutoff events is defined together with the set $Feas$ of feasible events. *Feasible events* are precisely those events whose causal predecessors are not cutoff events, and as such must be included in the prefix determined by the static cutoff events. An event e is a *static cutoff event*, if it is feasible, and there is $C \in C_e$ such that $C \subseteq Feas \setminus CutOff$, $C \approx [e]$, and $C \lhd [e]$. The token flow unfolding Unf_Θ defined by the set of events $Feas$ is called the *canonical prefix of* Unf_{max}. In [1] it is shown that Unf_Θ is uniquely determined by the cutting context Θ, complete and finite if (N, m_0) is bounded, $\{C_e\}_{e \in E}$ contains all local left-closed consistency sets and $\approx = \approx_{mar}$.

Example 2. Figure 1 shows a marked p/t-net (right side) together with the canonical prefix of its maximal token flow unfolding w.r.t. $\approx = \approx_{mar}$ and $\lhd = \subset$ (in the middle). The equivalence relation $\approx$ is shown below the canonical prefix. The set of cutoff events is $\{2, 4\}$. The canonical prefix has the two maximal consistency sets $C = C_{grey} = \{0, 3\}$ (color grey) and $C' = C_{white} = \{0, 1, 2, 4\}$ (color white).

3 Synthesis of bounded p/t-nets

In this section we first define regions and feasible places and prove their one-to-one correspondence for a finite LPES lpes and then extend this result by the handling of repeated behavior.

From subsection 2.5 we know that, in order to get a concrete synthesis algorithm, it is enough to

- define regions as solutions of a linear homogenous equation system,
- define feasible places, and
- show that each region defines a feasible place and each feasible place is defined by a region.

We assume that a finite lpes has a unique minimal event e_{init} with empty label.

We define a *token flow region* of a finite lpes as a tuple $\mathbf{r} = (r_k)_{k \in \mathcal{K}}$ of non-negative integers with

$$\mathcal{K} = (\prec \cup (\bigcup_{e \in E} (Con^e_{pre} \times \{e\}))),$$

satisfying properties (R_{init}), (R_{in}) and (R_{out}) as defined below, where Con^e_{pre} is the set of all *maximal left-closed consistency sets of Con containing event e*. As shown on the left side in figure 2, the values of a token flow region are graphically illustrated through numbers asigned to arcs.

The intuition behind the choice of the range $\mathcal{K}$ is as follows: A region $\mathbf{r}$ defines a place $p_{\mathbf{r}}$. Each event may produce an amount of tokens in this place. Considering a concrete run, there are the following two possibilities: A produced token is consumed from this place by a subsequent event of the run, or it is not further consumed by any subsequent event. A token flow region represents the amounts of tokens for both possibilities as follows:

- $r_{(e,f)}$ represents the number of tokens produced by e in $p_{\mathbf{r}}$ and subsequently consumed by f from $p_{\mathbf{r}}$ in each run containing the edge $(e, f) \in \prec$.
- $r_{(C,e)}$ represents the number of tokens produced by e in $p_{\mathbf{r}}$ and not consumed from $p_{\mathbf{r}}$ by any subsequent event belonging to the run lpo_C.

For an event e and a consistency set $C \in Con^e_{pre}$, we denote

- $IN^{\mathbf{r}}(e) = \Sigma_{e' \prec e} r_{(e',e)}$, the *($\mathbf{r}$-)intoken flow* of e.
- $OUT^{\mathbf{r}}_C(e) = r_{C,e} + \Sigma_{e \prec e', e' \in C} r_{(e,e')}$, the *($\mathbf{r}$-)outtoken flow of e w.r.t. C*. The outtoken flow of e_{init} we call *initial token flow*.

Let Con^{max}_{pre} denote the set of all maximal left-closed consistency sets.

The defining properties of a token flow region $\mathbf{r}$ of lpes, giving a homogeneous linear inequation system with variables $(r_k)_{k \in \mathcal{K}}$, are as follows:

(R_{init}) $OUT^{\mathbf{r}}_C(e_{init}) = OUT^{\mathbf{r}}_{C'}(e_{init})$ for $C, C' \in Con^{max}_{pre}$ (different runs have the same initial token flow).

(R_{in}) $IN^{\mathbf{r}}(e) = IN^{\mathbf{r}}(e')$ for events e, e' with $l(e) = l(e')$ (equally labelled events have the same intoken flow).

(R_{out}) $OUT^{\mathbf{r}}_C(e) = OUT^{\mathbf{r}}_{C'}(e')$ for events e, e' with $l(e) = l(e')$, $C \in Con^e_{pre}$ and $C' \in Con^{e'}_{pre}$ (equally labelled events have the same outtoken flow).

Then $\mathbf{r}$ defines a p/t-net-place $p_{\mathbf{r}}$ in the following way:

- $m_0(p_{\mathbf{r}}) := OUT^{\mathbf{r}}_C(e_{init})$ for some $C \in Con^{max}_{pre}$.
- $W(p_{\mathbf{r}}, t) := IN^{\mathbf{r}}(e)$ for some $e \in E$ with $l(e) = t$.
- $W(t, p_{\mathbf{r}}) := OUT^{\mathbf{r}}_C(e)$ for some $e \in E$ with $l(e) = t$ and $C \in Con^e_{pre}$.

Observe that the properties (R_{init}), (R_{in}) and (R_{out}) ensure that the definition of $p_{\mathbf{r}}$ is well-defined.

Example 3. Figure 2 shows, among other things, an LPES lpes with assigned token flow region $\mathbf{r}$ (left side) and a marked p/t-net (N, m_0) (right side). Maximal consistency sets are distinguished using the colors grey and white: There are the maximal consistency sets $C = C_{white} = \{0, 1, 2, 4\}$ and $C' = C_{grey} = \{0, 3\}$. The token flow region defines place p of (N, m_0), where only non-zero token flows are shown. For example $IN^{\mathbf{r}}(2) = IN^{\mathbf{r}}(4) = 1 = W((p, b))$, $OUT^{\mathbf{r}}_C(1) = 1 = W((a, p))$ and $OUT^{\mathbf{r}}_C(0) = OUT^{\mathbf{r}}_{C'}(0) = 1 = m_0(p)$.

We call a place p with weight function $W_p(p,\cdot) \cup W_p(\cdot,p)$ (with corresponding flow relation F_p) and initial marking $m_p(p)$ *feasible w.r.t. a finite* lpes, if p does not prohibit any run of lpes, i.e. if there is a token flow function x on E such that (lpes, x) fulfills properties (U_{in}), (U_{out}) and (U_{init}) from Definition 6 for $(N_p, m_p) = (\{p\}, T, F_p, W_p, m_p)$.[3]

Given a finite $lpes$, there is a one-to-one correspondence between token flow reagions of lpes and feasible places w.r.t. lpes.

Theorem 1. *(1) If p is a feasible place w.r.t.* lpes*, then there is a token flow region* $\mathbf{r}$ *of* lpes *with* $p = p_{\mathbf{r}}$.
(2) If $\mathbf{r}$ *is a token flow region of* lpes*, then* $p_{\mathbf{r}}$ *is a feasible place w.r.t.* lpes.

Proof. (1): Let p be a feasible place w.r.t. lpes and (lpes, x) fulfill properties (U_{in}), (U_{out}) and (U_{init}). We define a token flow region $\mathbf{r}$ of lpes as follows:

(i) $r_{(e,f)} := x(e,f)$ for $e \prec f$ (the value is an integer since the net has exactly one place).
(ii) $r_{C,e_{init}} := m_p(p) - \sum_{e \in C} r_{(e_{init},e)}$ for $C \in Con^{max}_{pre}$ (the value is non-negative from the definition of token flow unfoldings).
(iii) $r_{C,e} := W(l(e),p) - \sum_{e<f, f \in C} r_{(e,f)}$ for $e \neq e_{init}$ and $C \in Con^{e}_{pre}$ (the value is non-negative from the definition of token flow unfoldings).

Then the defining properties of token flow regions can be seen for $\mathbf{r}$ as follows (where C, C' are given as in the definition of token flow regions):

– (R_{init}):
$OUT^{\mathbf{r}}_C(e_{init}) = r_{C,e_{init}} + \sum_{e \in C} r_{(e_{init},e)} = m_p(p) = r_{C'} + \sum_{e \in C'} r_{(e_{init},e)} = OUT^{\mathbf{r}}_{C'}(e_{init})$ (the second and second to last equation follow from (ii)).
– (R_{in}):
$IN^{\mathbf{r}}(e) = \sum_{f \prec e} r_{(f,e)} = W(p,l(e)) = W(p,l(e')) = \sum_{f \prec e'} r_{(f,e')} = IN^{\mathbf{r}}(e')$ for $l(e) = l(e')$ (the second and second to last equation follow from (i) and the definition of token flow unfoldings).
– (R_{out}):
$OUT^{\mathbf{r}}_C(e) = r_{C,e} + \sum_{e \prec f, f \in C} r_{(e,f)} = W(l(e),p) = W(l(e'),p) = r_{C',e'} + \sum_{e' \prec f, f \in C'} r_{(e',f)} = OUT^{\mathbf{r}}_{C'}(e')$ for $l(e) = l(e')$ (the second and second to last equation follow from (iii) and the definition of token flow unfoldings).

Moreover, we get by construction $p = p_{\mathbf{r}}$.

(2): Let $\mathbf{r}$ be a token flow region of lpes and $p = p_{\mathbf{r}}$. We define a token flow function x on lpes by $x(e,f)(p) := r_{(e,f)}$ for $e \prec f$. Then the defining properties of feasible places (U_{in}), (U_{out}) and (U_{init}) from Definition 6 can be seen for x and (N_p, m_p) as follows: If $e \in E$, $C \in Con_{pre}$ and $C \subseteq C^{max} \in Con^{e}_{pre}$, then

– $IN^x(e)(p) = \sum_{f \prec e} x(f,e)(p) = \sum_{f \prec e} r_{(f,e)} = IN^{\mathbf{r}}(e) = W(p,l(e))$.
– $OUT^x_C(e)(p) = \sum_{e \prec f, f \in C} x(e,f)(p) \leqslant \sum_{e \prec f, f \in C^{max}} x(e,f)(p) \leqslant r_{C^{max},e} + \sum_{e \prec f, f \in C^{max}} r_{(e,f)} = OUT^{\mathbf{r}}_{C^{max}}(e) = W(l(e),p)$ (the first inequality follows, since the numbers $x(e,f)(p)$ are non-negative and $C \subseteq C^{max}$).

[3] We will explain in the proof of Theorem 3 why feasible places do not necessarily fulfill properties U_{it} and U_{min}.

– $OUT^x_C(e_{init})(p) = \sum_{e_{init} \prec e \in C} x(e_{init}, e)(p) \leqslant \sum_{e_{init} \prec e \in C^{max}} x(e_{init}, e)(p) \leqslant \sum_{e_{init} \prec e, e \in C^{max}} r_{(e_{init},e)} + r_{C^{max},e_{init}} = OUT^{\mathbf{r}}_{C^{max}} = m_p(p)$ (the first inequality follows, since all $x(e_{init}, e)(p)$ are non-negative and $C \subseteq C^{max}$).

In each case, the last equation follows from the definition of token flow regions. □

We now extend this approach by considering repeated behavior. To this end, we consider a specification given by a finite LPES lpes and a list of cutoff-events w.r.t. some cutting context $\Theta = (\approx, \lhd, \{C_e\}_{e \in E})$ with $\approx = \approx_{mar}$ and $\lhd = \subset$.

It is easy to see that $\approx_{mar}$ can be represented by a linear equation in terms of a token flow region $\mathbf{r} = (r_k)_{k \in \mathcal{K}}$ of lpes. If $C \in Con_{pre}$, then the marking reached after the execution of lpo_C is given by the tokens produced by events in C, which are not consumed by other events in C. In other words, roughly spoken, the marking is given by the sum of token flows on edges leaving C. This can be formalized as in the case of partial languages [9]: If $C^{max} \in Con^{max}_{pre}$ is some maximal left-closed consistency set with $C \subseteq C^{max}$, then

$$Mark(C)(p_{\mathbf{r}}) = \sum_{e \in C} r_{C^{max},e} + \sum_{e \in C,\, f \in C^{max} \setminus C\,,\, e \prec f} r_{(e,f)}.$$

Note that the result of the above sum is (by definition of token flow regions) independent of the choice of C^{max}.

The basic idea for the representation of repeated behavior is to specify additionally, that after some maximal events a repeated marking is reached, which was already seen before. These maximal events are given by a subset $E_{cut} = \{e_1, \ldots, e_n\}$ of the set of maximal events of E. For each event e_i, the corresponding repeated marking is specified by a left-closed consistency set $C_i \in C_{e_i}$ with $C_i \subset [e_i]$. The aim is to synthesize a net fulfilling $Mark(C_i) = Mark([e_i])$. Altogether, the specification consists of a finite LPES $\mathrm{lpes} = (E, Con, \prec, l)$ together with a so-called *cutoff-list* $Cut = (C_i, e_i)_{e_i \in E_{cut}}$.

This specification represents an infinite LPES lpes^{max}, called *completion of* (lpes, Cut). The cutoff-list Cut specifies that exactly the extensions of C_i should also be possible extensions of $[e_i]$ in lpes^{max}.

Definition 7 (Completion). *A* completion of (lpes, Cut) *is a minimal LPES* lpes^{max} *containing* lpes *and satisfying:*

Let $\approx'$ *be the smallest equivalence relation on left-closed consistency sets of* lpes^{max} *satisfying* $[e_i] \approx' C_i$ *which is preserved by finite extensions. Then for* $C \approx' C'$, $C \subset C'$ *and each extension* $E = \{e\}$ *of* C *in* lpes^{max} *there is an extension* $E' = \{e'\}$ *of* C' *in* lpes^{max} *with* $l(e) = l(e')$.

Note that it is not defined, how extensions are appended to $[e_i]$. Thus, there are many possible completions of (lpes, Cut), or, in other words, (lpes, Cut) represents a family of inifinte completions. By definition, different completions have the same set of events and only differ in the set of arcs used to append extensions.

Note also, that not all maximal events of lpes need to be in E_{cut}. If e is maximal, but not in E_{cut}, then the final marking of $[e]$ is indended not to enable any further transition occurrence.

It turns out, that a net (N, m_0) synthesized from a specification (lpes, Cut) in general does not satisfy several plausible conjectures:

- In general, lpes is not a complete prefix of the maximal unfolding of (N, m_0), since it may be longer than the complete prefix.
- In general, lpes is not a prefix of the maximal unfolding of (N, m_0) at all, since it may contain strong identical events.
- In general, lpes^{max} is not a prefix of the maximal unfolding of (N, m_0), since it may contain strong identical events.

Nevertheless, the maximal unfolding of the synthesized net (N, m_0) has a very strong relation to a completion lpes^{max} of (lpes, Cut): it perserves common prefixes and concurrency of runs of $lpes^{max}$ as given by definition 3. The formal problem statement, which we consider, is:

- **Given**: A finite LPES $\mathrm{lpes} = (E, Con, \prec, l)$ over a finite alphabet of transition names T together with a cutoff-list Cut.
- **Searched**: A marked p/t-net (N, m_0) with set of transitions T such that there is a completion $lpes^{max}$ of (lpes, Cut), satisfying:
  - the maximal token flow unfolding of (N, m_0) preserves common prefixes and concurrency of runs of $lpes^{max}$;
  - the partial language of this unfolding is the minimal partial language of an unfolding of a Petri net, which includes the partial language given by $lpes^{max}$.

Definition 8 (Token Flow Region). *A tuple* $\mathbf{r}$ *is a token flow region of* (lpes, Cut) *, if it satisfies the properties* (R_{init})*,* (R_{in})*,* (R_{out}) *and additionally*

(R_{cut}) $Mark([e_i])(p_{\mathbf{r}}) = Mark(C_i)(p_{\mathbf{r}})$ *for all* $e_i \in E_{cut}$.

A token flow region $\mathbf{r}$ defines a place $p_{\mathbf{r}}$ in the same way as in the finite case.

Example 4. Figure 2 shows an LPES and the cutoff-list $Cut = \{([1], 2), ([0], 4)\}$ and with assigned token flow region $\mathbf{r}$ (left side) and a marked p/t-net (N, m_0) (right side). The token flow region defines place p of (N, m_0), where only non-zero token flows are shown. For example $Mark([2])(p) = 1 = Mark([1])(p)$. Note that all places of (N, m_0) are feasible (see following definitions).

We call a place $p_{\mathbf{r}}$ potentially feasible w.r.t. lpes and Cut, if adding this place to the net does not prohibit any run of some completion of lpes.

Definition 9 (Potentially Feasible Place). *Let* $\mathrm{lpes} = (E, Con, \prec, l)$ *and* Cut *be given as in the formal problem statement. A place* p *with weight function* $W_p(p, \cdot) \cup W_p(\cdot, p)$ *(with corresponding flow relation* F_p*) and initial marking* $m_p(p)$ *is* potentially feasible *w.r.t.* (lpes, Cut)*, if there is a completion* lpes^{max} *of* (lpes, Cut) *and a token flow function* x *such that* (lpes^{max}, x) *fulfills properties* (U_{in})*,* (U_{out}) *and* (U_{init}) *from Definition 6 for the marked p/t-net* $(N_p, m_p) = (\{p\}, T, F_p, W_p, m_p)$.

We are interested in bounded nets, so we will consider only such potentially feasible places to be truly feasible, which, together with other potentially feasible places, form a bounded net.

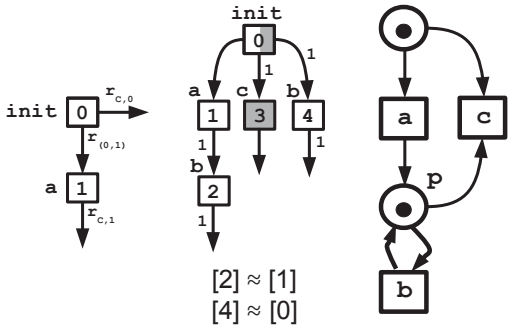

Fig. 2. Left: Graphical illustration of a token flow region for a maximal left-closed consistency set C. Middle: An LPES wit a cutoff-list and assigned token flow region. Right: A marked p/t-net with several feasible places. The place p corresponds to the token flow region.

Definition 10 (Feasible Place). *Let p is a potentially feasible place; then p is* feasible *iff there is a set of potentially feasible places P such that $p \in P$ and the marked p/t-net $(N_P, m_P) = (P, T, \cup_{p\in P} F_p, \cup_{p\in P} W_p, \cup_{p\in P} m_p)$ is bounded.*

Theorem 2. *(1) If p is a feasible place w.r.t. (lpes, Cut), then there is a token flow region $\mathbf{r}$ of (lpes, Cut) with $p = p_{\mathbf{r}}$.*
(2) If $\mathbf{r}$ is a token flow region of (lpes, Cut), then $p_{\mathbf{r}}$ is a feasible place of (lpes, Cut).

Proof. (1): Let p be a feasible place w.r.t. (lpes, Cut), lpes^{max} be a completion of (lpes, Cut) and (lpes^{max}, x) fulfill properties (U_{in}), (U_{out}) and (U_{init}). Since lpes is a part of lpes^{max}, we deduce from Theorem 1 that there is a token flow region $\mathbf{r}$ of lpes with $p = p_{\mathbf{r}}$. In particular, $\mathbf{r}$ satisfies the defining properties (R_{init}), (R_{in}) and (R_{out}).

Property (R_{cut}) follows from the fact that lpes^{max} contains the parts $[e_i] \setminus C_i$ infinitely often. Namely, by definition of lpes^{max} the non-empty part $[e_i] \setminus C_i$ is an extension of C_i which occurs iteratively in lpes^{max} after $[e_i]$. The existence of a token flow function of lpes^{max} shows that after the occurrence of lpo_{C_i}, the part $[e_i] \setminus C_i$ can occur arbitrarily often in an iterated way in (N_p, m_p). This means the occurrence of $[e_i] \setminus C_i$ does not decrease the number of tokens in p. Suppose that the occurrence of $[e_i] \setminus C_i$ increases the number of tokens in the place. This would imply that p is unbounded. Because each potentially feasible place will preserve that $[e_i] \setminus C_i$ can occur arbitrarily often, p will remain unbounded for any set of potentially feasible places which includes p. This contradicts with the fact that p is feasible. Thus, the occurrence of $[e_i] \setminus C_i$ does not change the number of tokens in p.

(2): Let $\mathbf{r}$ be a token flow region of (lpes, Cut) and $p = p_{\mathbf{r}}$. We deduce from Theorem 1 that there is a token flow function x on lpes, such that (lpes, x) fulfills properties (U_{in}), (U_{out}) and (U_{init}).

From property (R_{cut}) we deduce that there is a completion lpes^{max} of lpes such that x can be extended to a token flow function on lpes^{max} and (lpes^{max}, x) fulfills properties (U_{in}), (U_{out}) and (U_{init}). Namely, it can be seen by an easy inductive proof that for arbitrary $C \approx' C'$, $C \subset C'$ we have $Mark(C) = Mark(C')$ (for $C = C_i$ and $C' = [e_i]$ this corrsponds to (R_{cut})). This means that after the occurrence of C',

there are enough tokens in p for the occurrence of each transition, which can occur after the occurrence of C, i.e. (lpes, x) can be extended as required by the definition of completions, and therefore $p = p_{\mathbf{r}}$ is potentially feasible.

Let max be the maximum of the set $\{Mark(C) \mid C \in Con_{pre}\}$. Let p' be a complement place w.r.t. place p (i.e. $W(p,t) = W(t,p')$ and $W(t,p) = W(p',t)$ for each $t \in T$) with initial marking $m_{p'} = max - m_p$. One may observe that p' is potentially feasible and the net $(\{p,p'\}, T, F_p \cup F_{p'}, W_p \cup W_{p'}, m_p \cup m_{p'})$ is bounded, and therefore p is feasible. □

Theorem 3. *Let* $\mathrm{lpes} = (E, Con, \prec, l)$ *be a finite LPES together with a cutoff list* Cut *and let* (N, m_0) *be the finite p/t-net derived from the basis representation of the set of all token flow regions of* lpes*. Let* $Unf_{max}(N, m_0) =: (\mathrm{lpes}', x')$*,* $\mathrm{lpes}' = (E', Con', \prec', l')$*, be the maximal token flow unfolding of* (N, m_0)*. Then there is a completion* $lpes^{max}$ *of* (lpes, Cut) *satisfying:*

(1) lpes' *preserves common prefixes and concurrency of runs of* $lpes^{max}$*.*
(2) The partial language of lpes' *is a minimal Petri net unfolding partial language, which includes the partial language of* $lpes^{max}$*.*

Proof. Throughout the proof, we use the following notions:

- Let $\{r_1, \ldots, r_n\}$ be the set of basis regions of the set of all token flow regions of (lpes, Cut) and $p_i = p_{r_i}$ the corresponding places for $i = 1, \ldots, n$.
- Let (N, m_0), $N = (P, T, F, W)$, $P = \{p_1, \ldots, p_n$, be the finite p/t-net derived from the basis representation of the set of all token flow regions of lpes.
- Let (N_i, m_i), $N = (\{p_i\}, T_i, F_i, W_i)$, be the the net derived from (N, m_0) by restricting P to the place p_i.

The outline of the proof is the following:

(A) First we construct a completion $lpes^{max} = (E^{max}, Con^{max}, \prec^{max}, l^{max})$ of (lpes, Cut) and a token flow function x on $lpes^{max}$, such that $(lpes^{max}, x)$ is a prime token flow event structure and satisfies properties (U_{in}), (U_{out}) and (U_{init}) from Definition 6 w.r.t. the net (N, m_0).
(B) Second we construct a function $b : E^{max} \to E'$ such that for each left-closed consistency set C of lpes^{max} there holds that $b(C)$ is a left-closed consistency set of lpes' and $b|_C$ defines an isomorphism between lpo_C and a sequentialization of $lpo_{b(C)}$. This gives property (1).
(C) Third we show that the partial language of lpes' is a minimal Petri net unfolding partial language, which includes the partial language of $lpes^{max}$. This gives property (2).

ad (A): For each computed basis region r_i the corresponding place p_i is feasible w.r.t. (lpes, Cut) by Theorem 2. That means, in particular, that for each i there is a completion $\mathrm{lpes}_i^{max} = (E_i, Con_i, \prec_i, l_i)$ of (lpes, Cut) and a token flow function x_i on lpes_i^{max} such that $(\mathrm{lpes}_i^{max}, x_i)$ is a prime token flow event structure and satisfies properties (U_{in}), (U_{out}) and (U_{init}) w.r.t. the net (N_i, m_i). All these completions have the same set of events, the same set of consistency sets and the same labelling and can be combined into one completion $\mathrm{lpes}^{max} = (E^{max}, Con^{max}, \prec^{max}, l^{max})$ in the following way:

- $E^{max} = E_i$ for some i,
- $Con^{max} = Con_i$ for some i,
- $\prec^{max} = \bigcup_{i=1}^{n} \prec_i$,
- $l^{max} = l_i$ for some i.

Moreover, if we extend each x_i to the set $\prec^{max}$ by $x_i(e, e') = 0$ for $(e, e') \in \prec^{max} \setminus \prec_i$, we can combine the token flow functions x_i into a token flow function x on $\prec^{max}$ by

$$x = \Sigma_{i=1}^{n} x_i.$$

By construction, the properties of the completions $(\mathrm{lpes}_i^{max}, x_i)$ carry over to (lpes^{max}, x):

- (lpes^{max}, x) ist a prime token flow event structure: We have to show that $IN^x(e) = IN^x(e')$ for $l^{max}(e) = l^{max}(e')$. This follows by construction from $IN^{x_i}(e) = IN^{x_i}(e')$.
- (lpes^{max}, x) satisfies property (U_{in}) w.r.t. the net (N, m_0): We have to show that $IN^x(e)(p_i) = W(p_i, l(e))$ for each i. This follows by constructiuon from $IN^x(e)(p_i) = IN^{x_i}(e)(p_i)$ and $IN^{x_i}(e)(p_i) = W(p_i, l(e))$.
- (lpes^{max}, x) satisfies property (U_{out}) w.r.t. the net (N, m_0): Follows analoguously to (U_{in}).
- (lpes^{max}, x) satisfies property (U_{init}) w.r.t. the net (N, m_0): Follows analoguously to (U_{in}).

ad (B): We first show the desired property for the finite LPES lpes and then extend the result to infinite LPES lpes^{max}.

Note that the properties (U_{in}), (U_{out}) and (U_{init}) correspond to feasability of places w.r.t. the finite partial language corresponding to lpes as defined in [9]. That means, lpo_C is a run of (N, m_0) for each left-closed consistency set C of lpes. Moreover, there is a corresponding process of (N, m_0) such that lpo_C is a sequentialization of this process and $x(e, e')$ corresponds to the set of conditions connecting the events e and e' of the process for each edge (e, e'). Note that lpo_C is a proper sequentialization of the corresponding process if and only if $x(e, e') = 0$ for some edge (e, e'). In this case, (lpes^{max}, x) does not satisfy (U_{min}) and the synthesis problem does not have an exact solution - this is one of two cases where (lpes^{max}, x) is not a token flow unfolding.

We fix a total order on the set E of events of lpes respecting $\prec$ and try to build a finite prefix of lpes' by appending the events in this order starting with e_{init}. In fact, since on each run x is a token flow corresponding to some process (as seen above), it is always possible to append the next event e except in one case: If there is an event e_{id} already appended which is strong identical to e w.r.t x, then e is not appended according to the appending proceedure defined in [1], since it leads to isomorphic runs. In this case (lpes^{max}, x) does not satisfy (U_{id}) - and this is the second case where (lpes^{max}, x) is not a token flow unfolding.

We define $b : E \to E'$ inductively in the order of appending events as follows:

- (Induction basis) Let e_{init} be the initial event of lpes and e'_{init} be the initial event of lpes'. Then $b(e_{init}) = e'_{init}$.

– (Induction step) Let e be the next event of lpes in the appending procedure. Then we try to append the following event e' to the actual prefix of lpes':
  - $l'(e') = l(e)$,
  - ${}^\bullet e' = b(\{f \in {}^\bullet e \mid x(f,e) \neq 0\})$,
  - $x'(b(f), e') = x(f, e)$.

  If it is possible to append e', then conistency sets are updated w.r.t. e' (see below) and we define $b(e) = e'$. If it is not possible to append e' since there is a strong identical event e'_{id} of the actual prefix of lpes', then then conistency sets are updated w.r.t. e'_{id} (see below) we define $b(e) = e'_{id}$.
– (Updating consistency sets) Let e be the next event of lpes in the appending procedure and let C be a left-closed consistency set which is a subset of the events appended so far, such that $C \cup \{e\}$ is also a left-closed consistency set. We claim that $b(C) \cup \{b(e)\}$ is a left-closed consistency set and $b|_{C\cup\{e\}}$ defines an isomorphism between $lpo_{C\cup\{e\}}$ and a sequentialization of $lpo_{b(C)\cup\{b(e)\}}$. This can be seen inductively as follows:
  - (Induction hypothesis for C) $b(C)$ is a left-closed consistency set of lpes' and $b|_C$ defines an isomorphism between lpo_C and a sequentialization of $lpo_{b(C)}$. By construction, $x(e, e') = x'(b(e), b(e'))$ for each edge (e, e') with $x(e, e') \neq 0$, i.e. both runs have the same corresponding process.
  - ($b(C) \cup \{b(e)\}$ is left closed) Since $C \cup \{e\}$ is left-closed, we deduce ${}^\bullet e \subset C$. By construction, this implies ${}^\bullet b(e) \subset b(C)$, i.e. $b(C) \cup \{b(e)\}$ is left-closed.
  - (Induction step for $C \cup \{e\}$) Since $C \cup \{e\}$ is a consistency set, lpo_C is a run which enables e, and $\mathrm{lpo}_{C\cup\{e\}}$ is also a run. By induction hypothesis, lpo_C is (isomorphic to) a sequentialization of $\mathrm{lpo}_{b(C)}$. Moreover, lpo_C and $\mathrm{lpo}_{b(C)}$ correspond to the same process. Therefore, $\mathrm{lpo}_{b(C)}$ enables $b(e)$. By construction, $b(e)$ is appended to $b(C)$ using the same token flow as for the connection between e and C. This gives the induction hypothesis also for $C \cup \{e\}$.

In particular, two runs of (lpes, x) which are isomorphic due to strong identical events are mapped onto the same run of (lpes', x').

Finally, observe that the above argumentation for lpes holds for each finite prefix of lpes^{max}. Since the construction of the maximal unfolding is independend of the order in which events are appended [1], this gives the desired result for lpes^{max}.

ad (C): This classical result of region based synthesis follows by construction from Theorem 2, the considerations in subsection 2.5 and the fact, that the basis representation of the set of all token flow regions of (lpes, Cut) is used. □

For a specification which is itself a complete prefix of a maximal token flow unfolding of a p/t-net we get:

Corollary 1. *Let* lpes *be the LPES underlying the complete finite prefix of the maximal token flow unfolding of a bounded net* (N, m_0) *w.r.t.* $\approx{=}\approx_{mar}$ *and* $\lhd = \subset$. *Moreover, let* $E_{cut} = \{e_1, \ldots, e_n\}$ *be the set of static cutoff events of* lpes *and* C_i *be left-closed consistency sets with* $C_i \in C_{e_i}$, $C_i \approx [e_i]$ *and* $C_i \subset [e_i]$, *and let* $Cut = (C_i, e_i)_{e_i \in E_{cut}}$. *Let* (N', m_0') *be the finite p/t-net derived from the basis representation of the set of all token flow regions of* (lpes, Cut). *Let* $Unf_{max}(N', m_0') = (\mathrm{lpes}', x')$ *be the maximal token flow unfolding of* (N', m_0'). *Then* lpes' *and a completion of* lpes *have equal partial languages.*

Note that we used the choice $\lhd = \subset$ only in order to simplify the presentation. The results can be generalized to each choice of $\lhd$ and each choice of $\approx$ allowing its representation by a linear inequation in terms of token flows.

4 Related works

In [16] a one-to-one correspondence between finite 1-safe Petri nets and infinite regular trace event structures is established. On the one side in [16] more restrictive models as in this paper are considered (binary conflict relation, static concurrency relation). On the other side, the problem solved in [16] is stronger than the synthesis problem considered in this paper, since we neither give a characterization of LPES representing p/t-net behavior, nor answer the question, whether a given LPES corresponds to a p/t-net unfolding. Altogether, in [16] and in our paper different problems are considered which cannot be compared directly. There are several other works, which uses a Petri net as a final result and uses event structures as an intermediate representation [13, 14]. These works use folding methods and do consider a more restrictive setting (binary conflict relation, nets without arc weights).

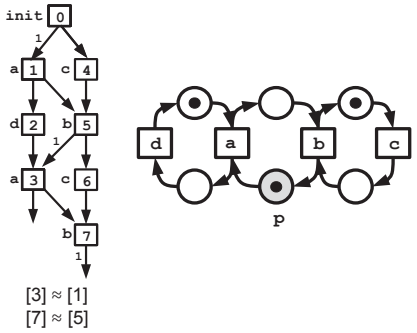

Fig. 3. Net with behavior which cannot be expressed by finite representations of infinite partial languages using terms from [9]

Up to now, several finite representations of infinite partial languages equipped with a corresponding notion of regions were developed. The most general one was given in [9] using terms with operations for the union, the parallel composition and the sequential composition of runs and partial languages. For each of these representations there are still examples of inifinite Petri net languages which cannot be represented, whereas we show in this paper, that each Petri net language can be specidied by means of a finite LPES together with a cut-off list.

Figure 3 shows on the left side an LPES with one maximal left-closed consistency set, together with a cutoff-list $\{e_3, e_7\}$ and corresponding consistency sets $C_{e_3} = [e_1]$ and $C_{e_7} = [e_5]$. It represents exactly the behavior of the Petri net on the right side and has assigned the token flow region $\mathbf{r}$ corresponding to place p. For example

$IN^{\mathbf{r}}(e_3) = IN^{\mathbf{r}}(e_1) = 1 = W(p,a)$, $OUT^{\mathbf{r}}(e_5) = OUT^{\mathbf{r}}(e_7) = 1 = W(b,p)$, $Mark([e_3])(p) = 0 = Mark([e_1])(p)$ and $Mark([e_7])(p) = 1 = Mark([e_5])(p)$. Note that it is not possible to represent the behavior of the shown net by other finite representations of infinite partial languages as for example terms [9].

5 Conclusion and future research

The main contribution of the paper is that it enables to handle arbitrary cyclic behaviour of bounded p/t nets. Moreover, it gives a basic result for both history (conflict) and concurrency preserving synthesis of Petri nets from a true concurrent and branching time behaviour specification - labelled prime event structures. The topic of branching time history preserving or conflict preserving synthesis remain full of non-trivial unsolved problems. One may for example ask for synthesis which preserves the absence of common prefixes. In general, analogously to a spectrum of behavioural equivalences [17] (and simulations), one can define a similar spectrum of history preserving synthesis methods w.r.t. given behavioural equivalences.

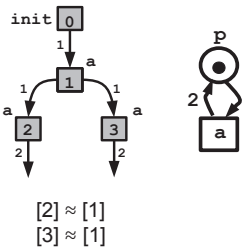

Fig. 4. Net with behavior which cannot be overapproximated by a bounded net with nonempty set of places [9].

With one modification, our mechanism may also serve for the synthesis of unbounded nets. As for bounded case, we represent an infinite partial language by an LPES lpes together with a cutoff-list $\{(e_1, C_1), \ldots, (e_n, C_n)\}$. In contrast to the case of bounded nets, we do not require, that the consistency sets C_i represent repeated markings, but increased markings, i.e. $Mark([e_i]) \geq Mark(C_i)$. Figure 4 shows on the right side an unbounded net whose behavior can be represented by the LPES shown on the left side together with the cutoff-list $\{(e_2, [e_1]), (e_3, [e_1])\}$. The LPES has assigned the token flow region $\mathbf{r}$ corresponding to place p of the net, for example: $IN^{\mathbf{r}}(e_1) = IN^{\mathbf{r}}(e_2) = 1 = W(p,a)$, $OUT^{\mathbf{r}}(e_1) = OUT^{\mathbf{r}}(e_3) = 2 = W(a,p)$ and $Mark([e_3])(p) = Mark([e_2])(p) = 3 \geq 2 = Mark([e_1])(p)$.

Note that the specification on the left side does not have a nonempty bounded solution, that means in general there are specifications such that there is no bounded net with nonempty set of places whose behavior overapproximates them.

References

1. R. Bergenthum, S. Mauser, R. Lorenz, and G. Juhás. Unfolding semantics of petri nets based on token flows. *Fundam. Inform.*, 94(3-4):331–360, 2009.
2. M. Dumas and L. García-Bañuelos. Process mining reloaded: Event structures as a unified representation of process models and event logs. In R. Devillers and A. Valmari, editors, *Application and Theory of Petri Nets and Concurrency*, volume 9115 of *Lecture Notes in Computer Science*, pages 33–48. Springer International Publishing, 2015.
3. J. Engelfriet. Branching processes of petri nets. *Acta Informatica*, 28(6):575–591, 1991.
4. J. Esparza, S. Römer, and W. Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
5. G. Juhás, R. Lorenz, and J. Desel. Can i execute my scenario in your net?. In G. Ciardo and P. Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 289–308. Springer, 2005.
6. V. Khomenko and M. Koutny. Towards an efficient algorithm for unfolding petri nets. In K. G. Larsen and M. Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 2001.
7. V. Khomenko and M. Koutny. Branching processes of high-level petri nets. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 458–472. Springer, 2003.
8. V. Khomenko, M. Koutny, and W. Vogler. Canonical prefixes of petri net unfoldings. *Acta Inf.*, 40(2):95–118, 2003.
9. R. Lorenz, J. Desel, and G. Juhás. Models from scenarios. In *Transactions on Petri Nets and Other Models of Concurrency VII*, pages 314–371. Springer Berlin Heidelberg, 2013.
10. K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In G. von Bochmann; D. K. Probst, editor, *CAV*, volume 663 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 1992.
11. J. Meseguer, U. Montanari, and V. Sassone. On the model of computation of place/transition petri nets. In R. Valette, editor, *Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 16–38. Springer, 1994.
12. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part i. *Theoretical Computer Science*, 13:85–108, 1981.
13. A. Polyvyanyy, L. García-Bañuelos, D. Fahland, and M. Weske. Maximal structuring of acyclic process models. *The Computer Journal*, 2012.
14. H. Ponce de León, C. Rodríguez, J. Carmona, K. Heljanko, and S. Haar. Unfolding-based process discovery. *CoRR*, abs/1507.02744, 2015.
15. A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.
16. P. S. Thiagarajan. Regular event structures and finite petri nets: A conjecture. In *Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg*, volume 2300 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.
17. G. U. van Glabbeek, R. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4-5):229–327, 2001.
18. G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.