Theory of Regions for the Synthesis of Inhibitor Nets from Scenarios

Robert Lorenz, Sebastian Mauser, and Robin Bergenthum

Department of Applied Computer Science,
Catholic University of Eichstätt-Ingolstadt
firstname.lastname@ku-eichstaett.de

Abstract. In this paper we develop a theory for the region-based synthesis of system models given as place/transition-nets with weighted inhibitor arcs (pti-nets) from sets of scenarios describing the non-sequential behaviour. Scenarios are modelled through labelled stratified order structures (LSOs) considering "earlier than" and "not later than" relations between events [6,8] in such a way that concurrency is truly represented.

The presented approach generalizes the theory of regions we developed in [10] for the synthesis of place/transition-nets from sets of labelled partial orders (LPOs) (which only model an "earlier than" relation between events). Thereupon concrete synthesis algorithms can be developed.

1 Introduction

Synthesis of Petri nets from behavioural descriptions has been a successful line of research since the 1990ies. There is a rich body of nontrivial theoretical results and there are important applications in industry, in particular in hardware design [3], in control of manufacturing systems [15] and recently also in workflow design [13,14].

The synthesis problem is the problem to construct, for a given behavioural specification, a Petri net of a considered Petri net class such that the behaviour of this net coincides with the specified behaviour (if such a net exists). There exist theories for the synthesis of place/transition-nets (p/t-nets) from behavioural models describing sequential semantics [1], step semantics [1] and partial order semantics [10]. There are also sequential, respectively step semantics, based approaches for the synthesis of elementary nets [4,5] and extensions to elementary nets with inhibitor arcs [2,11,12].

In this paper we generalize the synthesis theory for partial order semantics from [10] to p/t-nets with weighted inhibitor arcs (pti-nets). In [10] the behavioural specification is given by a set of labelled partial orders (LPOs) – a so called partial language – interpreted as a scenario-based description of the non-sequential behaviour of p/t-nets. The aim in [10] is the characterization and synthesis of a p/t-net whose behaviour coincides with a given partial language. That means, the LPOs of the partial language should exactly be the partially ordered executions of the searched p/t-net. Note hereby that partial languages regard the most general concurrency relationships between events (in contrast to sequential semantics considering no concurrency relations and step semantics considering only restricted transitive concurrency relations).

The synthesis of the p/t-net is based on the notion of regions: The p/t-net synthesized from a partial language inherits its transitions from the event labels of the LPOs which in turn describe the respective occurring actions. Through places causal dependencies between transitions are added restricting the set of executions. The idea is to add all places which do not restrict the set of executions too much in the sense that they do not prohibit the executability of any LPO specified in the partial language. These places are called feasible (w.r.t. the given partial language). Adding all feasible places yields a p/t-net – the so called saturated feasible p/t-net – which has a minimal set of partially ordered executions including the specified partial language (among all p/t-nets). Consequently the saturated feasible p/t-net solves the synthesis problem or there exits no solution of the problem. The general approach of a theory of regions is to determine feasible places by so called regions of the behavioural model.[1] As the main result in [10] we proposed a notion of regions for partial languages and showed that the set of regions exactly defines the set of feasible places. In this paper we lift this approach to the level of pti-nets. That means we generalize the notion of regions to a scenario-based behavioural model of pti-nets and show that these regions exactly define feasible places.

In the following we introduce the scenario-based behavioural model of pti-nets considered in this paper. We will examine the so called a-priori semantics of pti-nets [8] in which synchronicity of events is explicitly regarded.[2] Thus, as the model of non-sequential behaviour we consider a generalization of LPOs – so called labelled stratified order structures (labelled so-structures or LSOs) [6,8].[3] That means, given a pti-net, scenarios are specified by LSOs with transition names as event labels, and a specified scenario may be or may not be an execution of the net.

In an LPO ordered events are interpreted as causally dependent in the sense of an "earlier than" relation. Unordered events are considered as causally independent respectively concurrent. That means two events are concurrent, if they can occur in arbitrary order as well as synchronously. Thus, synchronicity cannot be distinguished from concurrency in the case of LPOs. A situation (1.) in which two events a and b can only occur synchronously or (2.) can occur synchronously and in the order $a \rightarrow b$, but not in the order $b \rightarrow a$, cannot be modelled with LPOs (obviously in both situations (1.) and (2.) the events are not concurrent, but synchronous occurrence is possible). For these situations LSOs include a "not later than" relation between events: a "not later than" b exactly describes (2.) and a symmetric "not later than" relation between events (a "not later than" b and b "not later than" a) models (1.). Thus, an LSO is based on an LPO (the "earlier than" relation is depicted with solid arcs in illustrations), to which a "not later than" relation (dashed arcs) between events is consistently added.

In [6] it was explained in detail that the "earlier than" relation of LPOs is not enough to describe executions of some Petri net classes such as inhibitor nets under the a-priori semantics and that LSOs form the adequate behavioural model for these net classes. In Figure 1 this phenomenon is illustrated: A pti-net and four LSOs describing executions

[1] For sequential or step semantics this theory lead to polynomial synthesis algorithms [1].

[2] There are also alternative semantics of inhibitor nets. The a-posteriori semantics (which is less general than the a-priori semantics from a causal point of view) is discussed in the conclusion.

[3] Note that just like LPOs in the case of p/t-nets, LSOs can model arbitrary dependency relations between transition occurrences of pti-nets, i.e. concurrency can be truly represented.

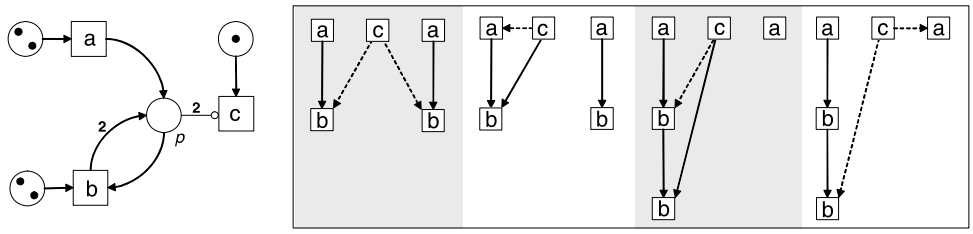

Fig. 1. A pti-net together with some executions

of the net are depicted. The pti-net has the only inhibitor arc (p, c) with inhibitor weight two. This arc restricts the behaviour of the net in such a way that the transition c is only enabled if additionally to the usual enabledness conditions of p/t-nets the place p contains at most two tokens. That means, through weighted inhibitor arcs it is tested if the number of tokens in a place does not exceed the inhibitor weight (as an enabledness condition). In the a-priori semantics the respective testing precedes the actual occurrence of the transition. That means the first LSO (from left) can be interpreted as an execution of the pti-net in the following sense: In the initial marking c and two instances of a are concurrently enabled (accordingly there exist no arcs modelling a causal dependency between the respective nodes), because the double occurrence of a produces (at most) two tokens in p. Therefore the occurrence of c is not prohibited (because the inhibitor arc (p, c) has the weight two). Moreover, after any occurrence of a the transition b is once enabled leading to the two solid "earlier than" arcs between each a and b. The two events labelled by b are concurrent. It is now important that after the double occurrence of a and one occurrence of b the place p contains three tokens. Thereby c is disabled by the inhibitor arc (p, c), i.e. b and c cannot occur in the order $b \rightarrow c$ (and therefore b and c are also not concurrent). However, the two transitions can occur synchronously, because in this situation the testing procedure (through the inhibitor arc (p, c)) precedes the occurrence procedure according to the a-priori rule. Thus, it precedes the enhancement of the number of tokens in p from two to three tokens through b. Furthermore, the occurrence in order $c \rightarrow b$ is obviously possible. Altogether, this behaviour of the b-labelled events and c can be described as follows: c cannot occur later than b or abbreviated c "not later than" b leading to dashed arcs between c and b in each case. Thus, an execution of a pti-net is an LSO, whose events are labelled with transition names, such that all transitions can occur in the given ordering and concurrency relations.

Technically executions will be defined as enabled LSOs. We propose a definition of enabledness for LSOs generalizing consistently the notion of enabled LPOs. Then every pti-net has assigned a set of executions (enabled LSOs). These describe the complete non-sequential behaviour of the pti-net, i.e. all possible causality and concurrency relationships between transition occurrences. Analogously to the notion of a partial language as a set of (non-isomorphic) LPOs we denote a set of (non-isomorphic) LSOs as a stratified language. Therefore, the non-sequential behaviour of a pti-net represented through the set of all executions of the net is a stratified language. The respective (scenario-based) synthesis problem can be formulated as follows:

Given: A stratified language $\mathcal{L}$ over a finite set of labels.
Searched: A pti-net whose set of executions coincides with the given language $\mathcal{L}$, if such a net exists.

As mentioned, for the less general problem with a partial language as the given behavioural model and a p/t-net as the searched system model the problem was solved in [10] applying the so called theory of regions. A region of a partial language defines a place by determining the initial marking of that place and the weights on each flow arc leading to and coming from a transition. A region of a stratified language additionally has to determine the weights of each inhibitor arc leading to a transition. It turns out that the notion of regions of stratified languages can be based on the notion of regions of partial languages. More precisely, omitting the "not later than" relation of all LSOs of a stratified language yields a set of LPOs forming the partial language underlying the given stratified language. To define regions of stratified languages we start with regions of the underlying partial language ignoring inhibitor arcs and complement these by "possible inhibitor arcs" as they are called in [2]. In this aspect the approach is similar as in [2,11,12] (where the authors started with classical regions of (step) transition systems and complemented these by "possible inhibitor arcs"). Roughly speaking, we add a "possible inhibitor arc" if in each possible intermediate marking state when executing a specified LSO subsequent events are not prohibited by this inhibitor arc. The identification of such inhibitor arcs is more complicated than for elementary nets and (step) transition systems (considered in [2,11,12]). On the one hand we have to regard weighted inhibitor arcs. On the other hand the marking states critical for the inhibitor tests are not directly modelled in LSOs (in contrast to transition systems). Having solved this problem, as the main theorem of this paper we show that the regions of a stratified language exactly define all feasible pti-net places (w.r.t. this stratified language). Thus, the regions of a stratified language define the saturated feasible pti-net. This net has a minimal set of executions including the given stratified language (among all pti-nets) and therefore solves the synthesis problem or is the best approximation if no solution exists. This solves the synthesis problem satisfactory from the theoretical point of view (for the considered setting). Practical algorithmic considerations are a topic of further research (see also the conclusion for a brief discussion).

The paper is structured as follows: First the basic notions of pti-nets and enabled LSOs are introduced (section 2). Then in section 3 the general fundamentals of the region based synthesis are developed and in section 4 the theory of regions is concretely evolved for the formulated synthesis problem.

2 Pti-nets

In this section we recall the basic definitions of *pti-nets* and introduce *enabled stratified order structures* as *executions* of pti-nets (leading to a formal model of scenario-based non-sequential semantics of pti-nets).

By $\mathbb{N}$ we denote the non-negative integers and by $\mathbb{N}^+$ the non-negative integers excluding 0. We additionally denote ω an infinite integer, i.e. $n < \omega$ for $n \in \mathbb{N}$. Given a finite set A, the identity relation on A is denoted by id_A and the set of all multi-sets over A is denoted by $\mathbb{N}^A$ (for $m \in \mathbb{N}^A$ we write $a \in m$ if $m(a) > 0$).

A *net* is a triple (P, T, F), where P is a set of *places*, T is a finite set of *transitions*, satisfying $P \cap T = \emptyset$, and $F \subseteq (P \cup T) \times (T \cup P)$ is a *flow relation*. Let (P, T, F) be a net and $x \in P \cup T$ be an element. The *preset* $\bullet x$ is the set $\{y \in P \cup T \mid (y, x) \in F\}$,

and the *post-set* $x\bullet$ 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 $X\bullet = \bigcup_{x \in X} x\bullet$.

A *place/transition net* (shortly *p/t-net*) 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 $W(x,y) = 0$.

Definition 1 (Pti-net). *A* pti-net N *is a five-tuple* (P,T,F,W,I)*, where* (P,T,F,W) *is a p/t-net and* $I : P \times T \to \mathbb{N} \cup \{\omega\}$ *is the* weighted inhibitor relation*. If* $I(p,t) \neq \omega$*, then* $(p,t) \in P \times T$ *is called* (weighted) inhibitor arc *and* p *is an* inhibitor place of t*.*

A *marking* of a pti-net $N = (P,T,F,W,I)$ is a function $m : P \to \mathbb{N}$ (a multi-set over P) assigning a number of tokens to each place. A transition t can only be executed if (in addition to the well-known p/t-net occurrence rule) each $p \in P$ contains at most $I(p,t)$ tokens. In particular, if $I(p,t) = 0$ then p must be empty. $I(p,t) = \omega$ means that t can never be prevented from occurring by the presence of tokens in p. In diagrams, inhibitor arcs have small circles as arrowheads. Just as normal arcs, inhibitor arcs are annotated with their weights. Now however, the weight 0 is not shown. A *marked pti-net* is a pair (N, m_0), where N is a pti-net and m_0 is a marking of N called *initial marking*. Figure 1 shows a marked pti-net.

According to the a-priori semantics of pti-nets, the inhibitor test for enabledness of a transition precedes the consumption and production of tokens in places. A multi-set (a step) of transitions is (synchronously) enabled in a marking if in this marking each transition in the step obeys the inhibitor constraints before the step is executed.

Definition 2 (Occurrence rule, a-priori semantics). *Let* $N = (P,T,F,W,I)$ *be a* pti-net*. A multi-set of transitions* τ *(a* step*) is* (synchronously) enabled to occur in a marking m (w.r.t. the a-priori semantics) *if* $m(p) \geq \sum_{t \in \tau} \tau(t) W(p,t)$ *and* $m(p) \leq I(p,t)$ *for each transition* $t \in \tau$ *(for every place* $p \in P$*).*

The *occurrence* of a step (of transitions) τ leads to the new marking m' defined by $m'(p) = m(p) - \sum_{t \in \tau} \tau(t)(W(p,t) - W(t,p))$ (for every $p \in P$). We write $m \xrightarrow{\tau} m'$ to denote that τ is enabled to occur in m and that its occurrence leads to m'. A finite sequence of steps $\sigma = \tau_1 \ldots \tau_n,\ n \in \mathbb{N}$ is called a *step occurrence sequence enabled in a marking* m *and leading to* m_n, denoted by $m \xrightarrow{\sigma} m_n$, if there exists a sequence of markings $m_1, \ldots, m_n$ such that $m \xrightarrow{\tau_1} m_1 \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} m_n$. A step occurrence sequence can be understood as a possible single *observation* of the behaviour of a pti-net, where the occurrences of transitions in one step are observed *at the same time* or *synchronously*. We use the notions for (marked) pti-nets also for (marked) p/t-nets (a p/t-net can be understood as a pti-net with an inhibitor relation which equals ω).

We now introduce *stratified order structures* (*so-structures*) to model executions of pti-nets as sketched in the introduction. We start with some basic notions preparative to the definition of so-structures. 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 $(a,b) \in \rightarrow$. Two nodes $a, b \in V$ are called *independent* w.r.t. the binary relation $\rightarrow$ if $a \not\rightarrow b$ and $b \not\rightarrow a$. We denote the set of all pairs of nodes independent w.r.t. $\rightarrow$ by $\mathrm{co}_{\rightarrow} \subseteq V \times V$. A *partial order* is a directed

graph $\mathrm{po} = (V, <)$, where $<$ is an irreflexive and transitive binary relation on V. If $\mathrm{co}_< = id_V$ then $(V, <)$ is called *total*. Given two partial orders $\mathrm{po}_1 = (V, <_1)$ and $\mathrm{po}_2 = (V, <_2)$, we say that po_2 is a *sequentialization* (or *extension*) of po_1 if $<_1 \subseteq <_2$.

So-structures are, loosely speaking, combinations of two binary relations on a set of nodes (interpreted as *events*), where one is a partial order representing an "earlier than" relation and the other represents a "not later than" relation. Thus so-structures describe finer causalities than partial orders. Formally, so-structures are *relational-structures* (*rel-structures*) satisfying certain properties. A rel-structure is a triple $\mathcal{S} = (V, \prec, \sqsubset)$, where V is a finite set (of *events*), and $\prec \subseteq V \times V$ and $\sqsubset \subseteq V \times V$ are binary relations on V. A rel-structure $\mathcal{S}' = (V, \prec', \sqsubset')$ is said to be an *extension* (or *sequentialization*) of another rel-structure $\mathcal{S} = (V, \prec, \sqsubset)$, written $\mathcal{S} \subseteq \mathcal{S}'$, if $\prec \subseteq \prec'$ and $\sqsubset \subseteq \sqsubset'$.

Definition 3 (Stratified order structure [6]). *A rel-structure $\mathcal{S} = (V, \prec, \sqsubset)$ is called* stratified order structure *(so-structure) if the following conditions are satisfied for all $u, v, w \in V$:*

$(C1)\ u \not\sqsubset u.$ $\qquad$ $(C3)\ u \sqsubset v \sqsubset w \wedge u \neq w \Longrightarrow u \sqsubset w.$

$(C2)\ u \prec v \Longrightarrow u \sqsubset v.$ $\qquad$ $(C4)\ u \sqsubset v \prec w \vee u \prec v \sqsubset w \Longrightarrow u \prec w.$

In figures $\prec$ is graphically expressed by solid arcs and $\sqsubset$ by dashed arcs. According to (C2) a dashed arc is omitted if there is already a solid arc. Moreover, we omit arcs which can be deduced by (C3) and (C4). It is shown in [6] that $(V, \prec)$ is a partial order. Therefore so-structures are a generalization of partial orders which turned out to be adequate to model the causal relations between events of pti-nets under the a-priori semantics. In this context $\prec$ represents the ordinary "earlier than" relation (as for p/t-nets) while $\sqsubset$ models a "not later than" relation (see Figure 1 for an example).

For our purposes we have to consider *labelled so-structures* (*LSOs*) where the nodes of an so-structure represent transition occurrences of a pti-net (nodes are labelled by transition names as in Figure 1). Formally these are so-structures $\mathcal{S} = (V, \prec, \sqsubset)$ together with a *set of labels* T and a *labelling function* $l : V \to T$. The labelling function l is lifted to a subset Y of V in the following way: $l(Y)$ is the multi-set over T given by $l(Y)(t) = |l^{-1}(t) \cap Y|$ for every $t \in T$. We will use the notations for so-structures also for LSOs as well as for LPOs (since an LPO can be understood as an LSO with $\prec = \sqsubset$). We will consider LSOs only up to isomorphism. Two LSOs $(V, \prec, \sqsubset, l)$ and $(V', \prec', \sqsubset', l')$ are called *isomorphic*, if there is a bijective mapping $\psi : V \to V'$ such that $l(v) = l'(\psi(v))$ for $v \in V$, $v \prec w \Leftrightarrow \psi(v) \prec' \psi(w)$ and $v \sqsubset w \Leftrightarrow \psi(v) \sqsubset' \psi(w)$ for $v, w \in V$. By $[\mathcal{S}]$ we will denote the set of all LSOs isomorphic to $\mathcal{S}$. The LSO $\mathcal{S}$ is said to *represent* the isomorphism class $[\mathcal{S}]$.

As explained, for the modelling of system behaviour the two relations of an LSO are interpreted as "earlier than" resp. "not later than" relation between transition occurrences. If two transition occurrences are in "not later than" relation, that means they can be observed (are allowed to be executed) synchronously or sequentially in one specific order. If two transitions are neither in "earlier than" relation nor in "not later than" relation, they are concurrent and can be observed (are allowed to be executed) synchronously or sequentially in any order. In this sense one LSO "allows" many observations (step sequences). If all these observations are enabled step occurrence sequences,

this LSO is called *enabled*. Formally the observations "allowed" by an LSO are defined through so called total linear extensions of the LSO:

Definition 4 (Total linear so-structures). *Let* $\mathcal{S} = (V, \prec, \sqsubset)$ *be an so-structure, then* $\mathcal{S}$ *is called* total linear *if* $\mathrm{co}_{\prec} = (\sqsubset \setminus \prec) \cup id_V$. *The set of all* total linear extensions *(or* linearizations*) of an so-structure* $\mathcal{S}$ *is denoted by* $lin(\mathcal{S})$.

Total linear so-structures are maximally sequentialized in the sense that no further $\prec$- or $\sqsubset$- relations can be added maintaining the requirements of so-structures according to Definition 3 (adding a $\prec$- or $\sqsubset$- relation leads to causal relations of the form $u \sqsubset v \prec u$). Therefore the linearizations $lin(\mathcal{S})$ of an so-structure $\mathcal{S}$ are its maximal extensions.

With this definition the set of step sequences (observations) "allowed" by an LSO is defined as the set of step sequences extending the LSO (that means emerging from adding causality to the LSO). A step sequence can be easily interpreted as a total linear LSO: Each step corresponds to a set of events labelled by transitions (transition occurrences) which are in "not later than" relation with each other representing synchronous transition occurrences. Transition occurrences in different steps are ordered in appropriate "earlier than" relation. Formally, for a sequence of transition steps $\sigma = \tau_1 \dots \tau_n$ define the total linear LSO $\mathcal{S}_\sigma = (V, \prec, \sqsubset, l)$ *underlying* σ by: $V = \bigcup_{i=1}^{n} V_i$ and $l : V \to T$ with $l(V_i)(t) = \tau_i(t)$, $\prec = \bigcup_{i<j} V_i \times V_j$ and $\sqsubset = ((\bigcup_i V_i \times V_i) \cup \prec) \setminus id_V$. ($\mathcal{S}_\sigma$ is total linear because $\mathrm{co}_{\prec} = \bigcup_{i=1}^{n} V_i \times V_i$). Altogether a step sequence σ is "allowed" by an LSO $\mathcal{S}$ if $\mathcal{S}_\sigma \in lin(\mathcal{S})$. For example the step sequences respectively observations "allowed" by the third LSO in Figure 1 can be characterized as follows: To each of the step sequences $cabb$, $(c + a)bb$, $acbb$ and $a(b + c)b$ an a has to be added either to one of the steps or representing a one-element step ordered in any position of the sequence. Any such possibility has to be regarded leading to 29 different "allowed" step sequences, e.g. including $cabab$, $(c + 2a)bb$, $2acbb$ or $a(b + c)(a + b)$.

Note that for each total linear LSO $\mathcal{S} = (V, \prec, \sqsubset, l)$ there is a step sequence σ such that $\mathcal{S}$ and $\mathcal{S}_\sigma$ are isomorphic. That means total linear LSOs can be interpreted as step sequences and the "allowed" observations of an LSO $\mathcal{S}$ in this sense are exactly the step sequences given by $lin(\mathcal{S})$.

Now we define enabled LSOs w.r.t. a marked pti-net as LSOs whose "allowed" observations are also "allowed" in the marked pti-net. More technically this means that any step sequence extending the LSO is enabled in the marked pti-net. Such an enabled LSO is called an execution of the marked pti-net.

Definition 5 (Enabled LSO). *Let* (N, m_0), $N = (P, T, F, W, I)$, *be a marked pti-net. An LSO* $\mathcal{S} = (V, \prec, \sqsubset, l)$ *with* $l : V \to T$ *is called* enabled (to occur) w.r.t. (N, m_0) (in the a-priori semantics) *if the following statement holds: Each finite step sequence* $\sigma = \tau_1 \dots \tau_n$ *with* $\mathcal{S}_\sigma \in lin(\mathcal{S})$ *is an enabled step occurrence sequence of* (N, m_0).

In other words an LSO is enabled if and only if it is consistent with the step semantics. This reflects the general idea for the modelling of non-sequential system behaviour that scenarios which are consistent with the non-sequential occurrence rule represent executions.[4] The presented definition is a proper generalization of the notion of enabled

[4] Another possibility for the definition of enabled LSOs is to consider sequences of concurrent steps of synchronous steps instead of sequences of synchronous steps. But both notions are equivalent, as discussed in [7].

LPOs: An LPO $\mathrm{lpo} = (V, \prec, l)$ with $l : V \to T$ is *enabled to occur in a marking* m of a marked p/t-net (P, T, F, W, m_0) if each step sequence which extends (sequentializes) lpo is a step occurrence sequence enabled in m_0. Since in LPOs concurrent and synchronous transition occurrences are not distinguished, here a step is considered as a set of events labelled by transitions (transition occurrences) which are concurrent.

Now it is possible to formally check that the LSOs from Figure 1 are indeed enabled LSOs w.r.t. the shown pti-net. For example in the case of the third LSO one would have to verify that the 29 step sequences "allowed" by this LSO (these are characterized above) are enabled step sequences of the marked pti-net.

Having defined single executions of marked pti-nets the behavioural model in our setting is defined as follows:

Definition 6 (Stratified language). *Let T be a finite set. A subset $\mathcal{L} \subseteq \{[\mathcal{S}] \mid \mathcal{S}$ is an LSO with set of labels $T\}$ is called* stratified language over T *(in the special case of LPOs it is called* partial language*). The* stratified language of executions $L(N, m_0)$ of a marked pti-net (N, m_0) *is defined as the stratified language consisting of all (isomorphism classes of) executions of (N, m_0).*

In the following we only consider stratified languages over sets T such that every $t \in T$ occurs as a label of some node of the stratified language (without explicitly mentioning this). Moreover, since we regard LSOs only up to isomorphism, we assume for the rest of the paper that a stratified language $\mathcal{L}$ over a finite set of labels is given by a set L of LSOs representing $\mathcal{L}$ in the sense that $[\mathcal{S}] \in \mathcal{L} \iff \exists \mathcal{S}' \in L : \ [\mathcal{S}] = [\mathcal{S}']$. Note that the stratified language of executions of a marked pti-net (N, m_0) is *sequentialization closed*. That means given an execution $\mathcal{S} \in L(N, m_0)$ of (N, m_0), any sequentialization of $\mathcal{S}$ is also an execution of (N, m_0). This is a simple observation using Definition 5, since sequentializations have a smaller set of linearizations. Moreover, as in the LPO-case, the stratified language of executions of (N, m_0) is *prefix closed*, where prefixes of so-structures are defined as subsets of nodes which are downward closed w.r.t. the $\sqsubset$-relation:

Definition 7 (Prefix). *Let $\mathcal{S} = (V, \prec, \sqsubset)$ be an so-structure and let $V' \subseteq V$ be such that $u' \in V'$, $u \sqsubset u' \Longrightarrow u \in V'$. Then $\mathcal{S}' = (V', \prec|_{V' \times V'}, \sqsubset|_{V' \times V'})$ is called* prefix *of $\mathcal{S}$. We say that the prefix $\mathcal{S}'$ is* defined by *V'. If additionally $(u \prec v \Longrightarrow u \in V')$ for some $v \in V \setminus V'$, then $\mathcal{S}'$ is called* prefix of v (w.r.t. $\mathcal{S}$).

3 The Synthesis Problem

The behaviour of a pti-net is described by its stratified language of executions. Therefore, for a stratified language L the question whether it represents the non-sequential behaviour of a marked pti-net can be formulated. The answer to this question together with a concrete characterization of such a net in the positive case are the central issues of this paper. Technically this synthesis problem can be fixed as follows:

Given: A stratified language L over a finite set of labels.
Searched: A marked pti-net (N, m_0) with $L(N, m_0) = L$ if such (N, m_0) exists.

In the following we outline the synthesis principles of the so called theory of regions. The concrete regions-based synthesis approach for the synthesis problem of pti-nets from stratified languages is developed in the next section.

The transition set T of the searched marked pti-net (N, m_0) is obviously given through the finite set of labels of the stratified language L (equally labelled nodes of LSOs in L represent occurrences of the same transition). Considering the pti-net $N = (\emptyset, T, \emptyset, \emptyset, \emptyset)$ with this transition set and an empty set of places, obviously any LSO in L is an execution of $(N, \emptyset)$. This is clear because in N there are no causal dependencies between the transitions. Therefore, every LSO with labels in T is enabled. On the other hand, there are also a lot of executions of $(N, \emptyset)$ not specified in L, i.e. $L(N, \emptyset) \supsetneq L$. Since we are interested in $L(N, m_0) = L$, we have to restrict the behaviour of (N, m_0) by introducing causal dependencies between transition occurrences. Such dependencies between transitions can (only) be realized by adding places to (N, m_0). Any place (with an initial marking) prohibits a certain set of LSOs from being enabled. The central idea is to add all places to (N, m_0) that do not prohibit LSOs specified in L from being enabled. These places are called *feasible places* and lead to the so called *saturated feasible pti-net* (N, m_0). For this net of course $L(N, m_0)$ still includes L, i.e. the specified LSOs in L are enabled w.r.t. (N, m_0) constructed in this way, while it is still not clear if $L(N, m_0) = L$. But now the marked pti-net (N, m_0) has minimal (w.r.t. set inclusion) non-sequential behaviour $L(N, m_0)$ including L, since all places not prohibiting L are regarded. That means that (N, m_0) is the appropriate candidate for the solution of the synthesis problem. If (N, m_0) does not solve the problem there exists no net solving the problem. This is ensured by construction because any other net solving the synthesis problem in this case would contradict the minimality property of (N, m_0) (since it would have a smaller set of executions including L).

The construction of the saturated feasible pti-net involves the introduction of places. Any place consists of an initial marking, a flow and an inhibitor relation to each transition and a flow relation from each transition. Consequently any place p can be defined by the value of its initial marking $m_0(p)$ together with the flow and inhibitor weights $W(p, t), W(t, p)$ and $I(p, t)$ for any transition $t \in T$ as depicted on the left of Figure 2 (a flow weight of 0 respectively an inhibitor weight of ω means that no such arc exists, compare section 2). Any place p restricts the behaviour of a marked pti-net by prohibiting a certain set of LSOs from being enabled. This set of LSOs prohibited by p does only depend on this place p. That means it does not matter if we consider the one-place net having p as its only place or a marked pti-net with a lot of places including p. More precisely, an LSO is enabled w.r.t. a marked pti-net (N, m_0), $N = (P, T, F, W, I)$, if and only if it is enabled w.r.t. every respective one-place net (for every $p \in P$). Regarding a given stratified language L the behavioural restriction of such a place p can be *feasible* or *non-feasible*, i.e. too restrictive, in the following sense (F, W, I and m_0 are determined by the definition of p – an example of a feasible and a non-feasible place is illustrated in Figure 2):

- *Non-feasible places p w.r.t. L*: There exists an LSO $\mathcal{S} \in L$, which is not enabled w.r.t. the one-place pti-net (N, m_0), $N = (\{p\}, T, F, W, I)$, i.e. $L \not\subseteq L(N, m_0)$.
- *Feasible places p w.r.t. L*: Every LSO $\mathcal{S} \in L$ is enabled w.r.t. the one-place pti-net (N, m_0), $N = (\{p\}, T, F, W, I)$, i.e. $L \subseteq L(N, m_0)$.

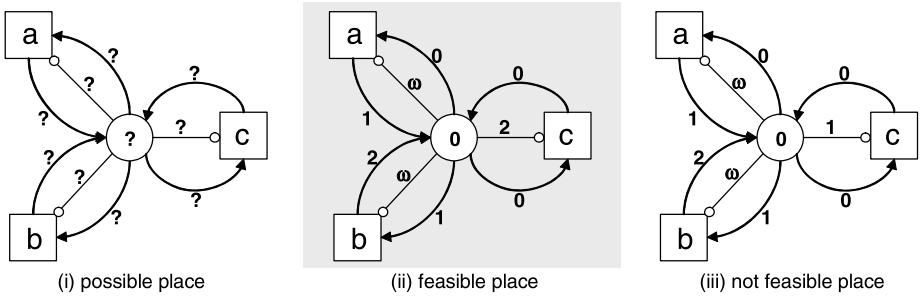

Fig. 2. (i) The general structure of a place. **(ii)** A feasible place w.r.t. the stratified language from Figure 1 (it coincides with the place p in Figure 1). **(iii)** A non-feasible place w.r.t. the stratified language from Figure 1. The inhibitor arc to the transition c (in contrast to (ii) with inhibitor weight 1 instead of 2) is causally too restrictive. To verify this recall the considerations in the context of Figure 1 in the Introduction.

Every net solving (positively) the synthesis problem necessarily does not contain a non-feasible place. Therefore the crucial idea is to consider the marked pti-net (N, m_0), $N = (P, T, F, W, I)$, containing exactly all feasible places w.r.t. L. Considering the above explanations this so called *saturated feasible pti-net* (N, m_0) guarantees that any LSO $\mathcal{S} \in L$ is enabled w.r.t. (N, m_0) (called property (A) of the saturated feasible pti-net in the following). Moreover, the saturated feasible pti-net (N, m_0) can have more executions than specified by L, but there is no marked pti-net with a smaller set of executions including L (called property (B) of the saturated feasible pti-net in the following). This is true because any other net (N', m'_0) whose set of executions $L(N', m'_0)$ includes L mandatory has less places than (N, m_0) since it may only contain feasible places (it holds $L(N', m'_0) \supseteq L(N, m_0)$ if (N', m'_0) has less places than (N, m_0)).

Definition 8 (Saturated feasible pti-net). *Let L be a stratified language over the set of labels T, then the marked pti-net (N, m_0), $N = (P, T, F, W, I)$, such that P is the set of all places feasible w.r.t. L is called* saturated feasible pti-net (w.r.t. L).

The saturated feasible pti-net (N, m_0) w.r.t. L in general has infinitely many (feasible) places. It fulfills (A) $L \subseteq L(N, m_0)$ and (B) $L(N, m_0) \subseteq L(N', m'_0)$ for each marked pti-net (N', m'_0), $N' = (P', T, F', W', I')$, fulfilling $L \subseteq L(N', m'_0)$ (thus fulfilling (A)). For the solution of the synthesis problem it is enough to consider only the saturated feasible pti-net, because either this net solves the synthesis problem or there is no solution for the problem:

Theorem 1. *Let L be a stratified language and (N, m_0), $N = (P, T, F, W, I)$, be the saturated feasible pti-net w.r.t. L, then $L(N, m_0) \neq L$ implies $L(N', m'_0) \neq L$ for every marked pti-net (N', m'_0), $N' = (P', T, F', W', I')$.*

Property (B) even tells us more than this theorem: In the case $L(N, m_0) \neq L$, $L(N, m_0)$ is the best upper approximation to L. That means the saturated feasible pti-net is the best approximation to a system model with non-sequential behaviour given by L among all marked pti-nets allowing the behaviour specified by L.

Altogether, in order to solve the synthesis problem in our setting, we want to calculate the saturated feasible pti-net. Therefore we are interested in a characterization

of feasible places based on L that leads to an effective calculation method for feasible places. In the p/t-net case such a characterization was developed for behavioural models w.r.t. sequential semantics and step semantics [1] with the notion of *regions* of the behavioural model. These approaches were generalized in [10] to partial languages. In the latter case it was shown that every region of a partial language L defines a place such that

(1) Each place defined by a region of L is feasible w.r.t. L.
(2) Each place feasible w.r.t. L can be defined by a region of L.

In [10] we used a slightly different terminology as in this paper. In particular, we did not use the notion of feasible places there but their characterization by the so called *token flow property*. To prove the mentioned results we assumed that the set L of LPOs representing the given partial language satisfies certain technical requirements. More precisely, L was assumed to be prefix and sequentialization closed, since such partial languages are the only candidates as models of the non-sequential behaviour of a marked p/t-net. Moreover, we required that LPOs which are in conflict (describe alternative executions) have disjoint node sets (for the exact formal definitions we refer to [10]). We showed that such representations always exist. Since our approach is based on the results in [10], we require analogous technical properties for the representation L of the specified stratified language. As in the p/t-net case it is no restriction for the synthesis problem to consider only such representations of prefix and sequentialization closed stratified languages.

In examples we will always give such L by a *set of minimal LSOs* of L (minimal LSOs of L are not an extension of some other LSO in L), such that each LSO in L is an extension of some prefix of one of these minimal LSOs. Thus every set of LSOs which are not extensions of each other can be interpreted as a representation of a stratified language by minimal LSOs. For example the four LSOs in Figure 1 represent the stratified language that exactly coincides with the stratified language of executions given by the non-sequential behaviour of the pti-net on the left of Figure 1.

The main aim of this paper is the generalization of the region definition to our setting such that (1) and (2) hold for stratified languages L w.r.t. pti-nets. With such a notion of regions based on stratified languages, the saturated feasible pti-net w.r.t. a stratified language L is directly defined by the set of all regions: Every region of L defines a place of the saturated feasible pti-net. This is the basis for effective solution algorithms for the synthesis problem considered in this paper: In the case of [1] as well as [10] (for the approach of [10] we developed a respective algorithm for finite partial languages in the recent paper [9]) algorithms for the calculation of finite representations of the set of regions were deduced. In the conclusion we argue why this is also possible in our setting. A detailed elaboration of this topic will be the issue of further publications.

4 Regions of Stratified Languages (w.r.t. Pti-nets)

In this section we extend the notion of regions known for partial languages and p/t-nets to the setting of pti-nets. In [10] it is shown that the regions of a partial language in the context of p/t-nets exactly correspond to the feasible places w.r.t. the partial language. Our aim is to show the same for stratified languages and pti-nets.

Fix a marked pti-net (N, m_0), $N = (P, T, F, W, I)$, and an LSO $\mathcal{S} = (V, \prec, \sqsubset, l)$ with $l : V \to T$. Assume that $\mathcal{S}$ is enabled to occur w.r.t. (N, m_0). Since the inhibitor relation I of (N, m_0) restricts the behaviour of the underlying p/t-net (P, T, F, W, m_0), $\mathcal{S}$ is then also enabled w.r.t. the p/t-net $(N', m_0) = (P, T, F, W, m_0)$ underlying N. In a p/t-net, transitions which can be executed synchronously can also be executed concurrently. Therefore, also the LPO $\mathrm{lpo}_{\mathcal{S}} = (V, \prec, l)$ (omitting the "not later than" relation) underlying $\mathcal{S}$ is enabled w.r.t. the p/t-net (N', m_0). Altogether, for a set of enabled LSOs w.r.t. (N, m_0), the LPOs underlying these LSOs are enabled w.r.t. the underlying p/t-net (N', m_0). Considering a one place-net (N, m_0) as in the definition of feasible places, it becomes clear that we have the following necessary condition for a feasible place p w.r.t. a stratified language L: The place p' underlying p defined by omitting the inhibitor relation from p is feasible w.r.t. the underlying partial language consisting of the LPOs underlying the LSOs from L.

Lemma 1. *Let L be a stratified language with transition labels T and let $L' = \{(V, \prec, l) \mid (V, \prec, \sqsubset, l) \in L\}$ be* the partial language underlying L. *Then for any place p feasible w.r.t. L (in the pti-net context) the place p'* underlying p, *defined by $W(p', t) = W(p, t), W(t, p') = W(t, p), I(p, t) = \omega$ for every $t \in T$ and $m_0(p') = m_0(p)$, is feasible w.r.t. L' (in the pti-net as well as the p/t-net context).*

That means, any place p feasible w.r.t. L can be constructed from a place p' which is feasible w.r.t. the underlying partial language L' and has inhibitor weights $I(p', t) = \omega$ (for every transition $t \in T$) by adding appropriate (respectively feasible) inhibitor weights $I(p, t)$. In particular, every place p feasible w.r.t. L fulfilling $I(p, t) = \omega$ for every transition $t \in T$ is feasible w.r.t. L'. On the other hand also the reverse holds: Every place p' feasible w.r.t. L' is feasible w.r.t. L because the enabledness of the underlying LPOs from L' w.r.t. the one place net defined by p' implies the enabledness of the original LSOs from L w.r.t. this net (since they have more causal ordering). Consequently, the sets of feasible places p with $I(p, t) = \omega$ for every $t \in T$ coincide for L and L'. Since L' is a partial language and the restriction $I(p, t) = \omega$ corresponds to p/t-net places, we can characterize these places using the theory of regions for partial languages and p/t-nets from [10]: The p/t-net places feasible w.r.t. the partial language L' are exactly the places defined by regions of L'. Thus, we can characterize the set of all feasible places p w.r.t. L fulfilling $I(p, t) = \omega$ for every $t \in T$ with the regions theory of [10]. Moreover, from Lemma 1 we know that any further place feasible w.r.t. L having inhibitor weights not equal to ω coincides with one of these feasible places p (fulfilling $I(p, t) = \omega$ for every $t \in T$) except of the inhibitor weights.

As a consequence, the regions definition in our setting is based on the regions definition for partial languages and p/t-nets. More precisely, we start with p/t-net regions of the underlying partial language L'. This leads to the set of feasible places p fulfilling $I(p, t) = \omega$ for every $t \in T$ as described above. Then we examine for each such p which other inhibitor weight combinations $I(p, t)$ (preserving the flow relation and the initial marking) also lead to feasible places. For this we use that incrementing an inhibitor weight alleviates the behavioural restriction of the respective inhibitor arc. In particular the set of enabled step sequences and the set of executions increases. Consequently incrementing the inhibitor weight of a feasible place obviously leads again to a

feasible place (since the resulting places are causally less restrictive). That means, considering a feasible place p as above with $I(p,t) = \omega$ for every $t \in T$, there is a minimal value $I_{min}(p,t) \in \mathbb{N} \cup \{\omega\}$ for the inhibitor weight to every single transition t such that the following holds: p is still feasible if we change $I(p,t)$ so that $I(p,t) \geq I_{min}(p,t)$ and no more feasible if we change $I(p,t)$ so that $I(p,t) < I_{min}(p,t)$ (preserving $I(p,t') = \omega$ for every $t' \in T \setminus \{t\}$). Now it is important that we can combine these different minimal values $I_{min}(p,t)$ (for different $t \in T$) to one global lower bound in the following sense: Preserving the flow relations and the initial marking, p is feasible if $I(p,t) \geq I_{min}(p,t)$ for every $t \in T$ and p is non-feasible if $I(p,t) < I_{min}(p,t)$ for one $t \in T$. This combination to one global bound is possible because, given a fixed flow relation, the inhibitor arcs have no causal interrelation between each other. That means it is possible to check the enabledness of an LSO by testing the enabledness w.r.t. the inhibitor arcs one by one. Altogether, the set of feasible places w.r.t. a stratified language L can be defined by the set of p/t-net places (places p with $I(p,t) = \omega$ for every $t \in T$) feasible w.r.t. L together with a global lower bound for the inhibitor weights of each such p/t-net place. Since the feasible p/t-net places p can be characterized by the regions definition for partial languages and p/t-nets, we first recall the regions definition of [10]. Based on this regions definition we then identify the lower inhibitor weight bounds $I_{min}(p,t)$ for the respective places p which then leads to the set of all feasible places w.r.t. L. This generalizes the definition of regions from [10].

The idea of defining regions for partial languages in [10] is based on the notion of *token flow functions*: If two events v and v' are ordered in an LPO $\mathrm{lpo} = (V, <, l)$ – that means $v < v'$ – this specifies that the corresponding transitions $l(v)$ and $l(v')$ are causally dependent in the sense of an "earlier than" relation. In a p/t-net such a causal dependency arises exactly if the occurrence of the transition $l(v)$ produces tokens in a place, which are consumed by the occurrence of the other transition $l(v')$. Such a place will be defined by a token flow function x: Assign to every edge (v, v') of lpo a natural number $x(v, v')$ representing *the number of tokens which are produced by the occurrence of* $l(v)$ *and consumed by the occurrence of* $l(v')$ *in the place to be defined*. Thus, a token flow function x describes the flow weights of a respective place. Additionally the initial and final marking of the place have to be regarded. Therefore, we extend an LPO lpo by an *initial and final event*, representing transitions producing the initial marking of the place to be defined and consuming the final marking of the place to be defined (after the occurrence of lpo). This leads to the $\star$*-extension* $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ of lpo defined by $V^\star = (V \cup \{v_0, v_{\max}\})$, $v_0, v_{\max} \notin V$, $\prec^\star = \prec \cup (\{v_0\} \times V) \cup (V \times \{v_{\max}\}) \cup \{(v_0, v_{\max})\}$, $l^\star(v_0), l^\star(v_{\max}) \notin l(V)$, $l^\star(v_0) \neq l^\star(v_{\max})$ and $l^\star|_V = l$ (v_0 is the *initial event of* lpo and $v_{\max}$ the *final event of* lpo). By defining the token flow function on the edges of $\mathrm{lpo}^\star$ (instead of lpo) also the initial and final marking can be specified.

The natural numbers assigned to the arcs of $\mathrm{lpo}^\star$ by x represent the consumed and produced tokens of the involved transitions in the respective place (whereas the tokens produced by the initial event are interpreted as the initial marking and the tokens consumed by the final event as the final marking). Since the consumed and produced tokens of a transition in a fixed place is given by the flow weights W, we can define the flow weights of the place by x. Clearly, a necessary condition for the definition of W is

that equally (with the same transition) labelled events should produce and consume the same overall number of tokens w.r.t. x. The number of tokens produced by an event v of an LPO $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ is called the *outtoken flow of v (w.r.t. lpo and x)* defined by $\mathrm{Out}_{\mathrm{lpo}}(v, x) = \sum_{v <^\star v'} x(v, v')$. The outtoken flow $\mathrm{Out}_{\mathrm{lpo}}(v_0, x)$, which by construction represents the initial marking of the place to be defined by x, is called the *initial token flow of lpo (w.r.t. x)*. The number of tokens consumed by an event v of an LPO $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ is called the *intoken flow of v (w.r.t. lpo and x)* defined by $\mathrm{In}_{\mathrm{lpo}}(v, x) = \sum_{v' <^\star v} x(v', v)$.

For the definition of the token flow function we not only have to regard one LPO, but a partial language L' over T. Thus we have to consider token flow functions on a set of LPOs. The central property that equally labelled events should produce and consume the same number of tokens has to be extended spanning all LPOs of the given partial language in this situation. Furthermore, since the initial marking has to be unique, the number of tokens produced by the initial event has to coincide for all regarded LPOs.

Formally we consider a $\star$-extension $\mathrm{lpo}^\star = (V^\star, <^\star, l^\star)$ of each $\mathrm{lpo} \in L'$ such that (i) for each two LPOs $(V, <, l), (V', <', l) \in L'$ $l^\star(v_0) = (l')^\star(v_0)$ and (ii) $l^\star(v_{\max}) \neq (l')^\star(v_{\max})$ ($\notin T$) for each two distinct $(V, <, l), (V', <', l') \in L'$. Then the set $(L')^\star = \{\mathrm{lpo}^\star \mid \mathrm{lpo} \in L'\}$ is called *$\star$-extension of L'*. We denote $E_{(L')^\star} = \bigcup_{(V^\star, <^\star, l^\star) \in (L')^\star} <^\star$ as the set of edges of all $\star$-extensions of LPOs in L'. A token flow function x of L' is a function assigning natural numbers to every edge in $E_{(L')^\star}$, such that the tokens produced and consumed by equally labelled events coincide.

Definition 9 (Token flow function of a partial language). *Let L' be a partial language, then a function $x : E_{(L')^\star} \to \mathbb{N}$ is called* token flow function *of L', if for all $\mathrm{lpo} = (V, <, l), \mathrm{lpo}' = (V', <', l') \in (L')^\star$ and for all $v \in V^\star, v' \in V'^\star$ there holds: $l(v) = l'(v') \Longrightarrow (\mathrm{In}_{\mathrm{lpo}}(v, x) = \mathrm{In}_{\mathrm{lpo}'}(v', x) \wedge \mathrm{Out}_{\mathrm{lpo}}(v, x) = \mathrm{Out}_{\mathrm{lpo}'}(v', x))$.*

Since we required that the initial events of all LPOs in $(L')^\star$ have the same label, Definition 9 especially implies that the initial token flows of all LPOs in L' are equal. As explained, the coincidence of the intoken and outtoken flow (respectively the consumed and produced tokens) w.r.t. x of equally labelled events allows to define the *corresponding place p_x to x* (in the net with transitions given by the node labels T of L') by $W(l(v), p_x) = \mathrm{Out}_{\mathrm{lpo}}(v, x)$, $W(p_x, l(v)) = \mathrm{In}_{\mathrm{lpo}}(v, x)$ and $m_0(p_x) = \mathrm{Out}_{\mathrm{lpo}}(v_0, x)$ for every $\mathrm{lpo} \in L'$ and every node v of lpo. That means the flow weights of p_x are given by the intoken and outtoken flow of the LPO-events and the initial marking by the initial token flow of the LPOs. In [10] the regions of a partial language L' are exactly the token flow functions of L' as defined here. The respective feasible places are the corresponding places.

We are now interested in token flow functions of the partial language L' underlying the given stratified language L. Thereto we formally define a *token flow function of a stratified language* as a token flow function of its underlying partial language:

Definition 10 (Token flow function of stratified languages). *Let L be a stratified language. Then a* token flow function *of L is a token flow function of the partial language $L' = \{(V, \prec, l) \mid (V, \prec, \sqsubset, l) \in L\}$ underlying L.*

In illustrations we annotate each $\prec$-arc of an LSO in L with the value assigned to the respective arc in L' by a token flow function x (the value 0 is not shown). The non-

zero values of x assigned to edges starting from v_0 respectively ending in v_{max} are depicted with small arrows without an initial node respectively without a final node. We only consider minimal LSOs of L because the values of a token flow function on the edges of an LSO already constitute the values on edges of prefixes and extensions (as in the LPO-case). Figure 3 sketches an example token flow function of the stratified language from Figure 1 and the respective corresponding place p (with $I(p,t) = \omega$ for all $t \in T$). The intoken and outtoken flow of equally labelled nodes coincide (e.g. all b-labelled nodes have intoken flow 1 and outtoken flow 2 and the initial token flow of all underlying LPOs is 0).

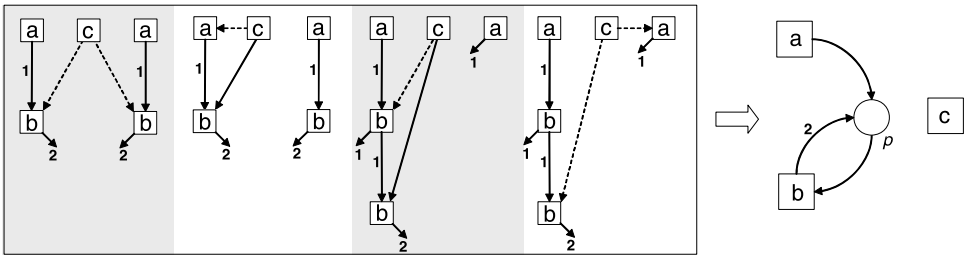

Fig. 3. A token flow function of the stratified language from Figure 1 and the corresponding (feasible) place (with inhibitor weights ω)

According to the above explanations, the places p corresponding to token flow functions x of a stratified language L now exactly define all feasible places w.r.t. L with inhibitor weights ω. In particular, the place p in Figure 3 is feasible w.r.t. the given stratified language. Now it remains to identify the lower bounds $I_{min}(p,t)$ ($t \in T$) for each of these feasible places p (such that $I(p,t) \geq I_{min}(p,t)$ for every $t \in T$ still leads to a feasible place p but $I(p,t) < I_{min}(p,t)$ for some $t \in T$ leads to a non-feasible place p). These minimal possible inhibitor weights $I_{min}(p,t)$ have to be detected with the token flow function x of L. The strategy is as follows: Considering a node v of an LSO $\mathcal{S} = (V, \prec, \sqsubset, l) \in L$ we calculate the minimal inhibitor weight $\mathrm{Inh}(x,v)$ from p to $l(v)$ (where p corresponds to x), such that the occurrence of the transition $l(v)$ according to the causal dependencies given for v in $\mathcal{S}$ is possible. That means, the event v in the context of the scenario given by $\mathcal{S}$ must not be prohibited by an inhibitor arc from p to $l(v)$ in the net if $I(p,l(v)) \geq \mathrm{Inh}(x,v)$, but it is prohibited by such an arc if $I(p,l(v)) < \mathrm{Inh}(x,v)$. Choosing the inhibitor weight $I(p,l(v))$ too small leads to an intermediate marking state of the scenario $\mathcal{S}$ in which a too large number of tokens in p prohibits the occurrence of v. Consequently, in order to determine the minimal inhibitor weight $\mathrm{Inh}(x,v)$ not prohibiting v – called *inhibitor value* of v (w.r.t. x) in the following – it is necessary to calculate the numbers of tokens in p for all intermediate states in which v can occur according to $\mathcal{S}$. Such states are exactly defined by prefixes of v. The maximum of all these possible numbers of tokens in p in such a prefix-state then defines the inhibitor value $\mathrm{Inh}(x,v)$ of v, because according to the scenario $\mathcal{S}$ the transition $l(v)$ should be enabled in each of these token allocations of p. The number of tokens in p in one such prefix-state can be calculated by the token flow function x. The respective number of tokens is given by the number of tokens in p after the execution of

the prefix in the corresponding one-place net, called the *final marking of the prefix w.r.t.* x. By construction, the values of x on $\prec^\star$-edges between events of the prefix correspond to tokens which are produced and consumed in p by events in this prefix. On the other hand, the values of x on $\prec^\star$-edges from events of the prefix to events subsequent to the prefix correspond to tokens which are produced by events in the prefix and remain in p after the execution of the prefix. Consequently, the final marking of a prefix can be determined by adding the values of x on $\prec^\star$-edges leaving the prefix.

Definition 11 (Final marking of prefixes). *Let L be a stratified language and x be a token flow function of L. Let $\mathcal{S}' = (V', \prec', \sqsubset', l')$ be a prefix of $\mathcal{S} = (V, \prec, \sqsubset, l) \in L$ and v_0 be the initial event of $\mathrm{lpo}^\star_{\mathcal{S}} = (V^\star, \prec^\star, l^\star)$. The* final marking of $\mathcal{S}'$ (w.r.t. x) *is denoted and defined by $m_{\mathcal{S}'}(x) = \sum_{u \in V',\, v \notin V',\, u \prec v} x(u, v) + \sum_{v \notin V'} x(v_0, v)$.*

The final marking of a prefix w.r.t. x can equivalently be calculated by firing the transitions corresponding to the prefix in the one-place net with the place p defined by x (i.e. it is independent from the concrete token flow distribution x and only dependent on p): $m_{\mathcal{S}'}(x) = \sum_{u \in V',\, v \notin V',\, u \prec v} x(u, v) + \sum_{v \notin V'} x(v_0, v) = \sum_{v \in V' \cup \{v_0\}} (\sum_{v \prec^\star w} x(v, w) - \sum_{w \prec^\star v} x(w, v)) = \mathrm{Out}(v_0, x) - \sum_{v \in V'} (\mathrm{In}(v, x) - \mathrm{Out}(v, x)) = m_0(p) - \sum_{v \in V'} (W(p, l(v)) - W(l(v), p))$ (the first equation follows since the values on edges within V' cancel each other out).

Summarizing, the calculation of $\mathrm{Inh}(x, v)$ is achieved by identifying all prefixes of v and calculating the final marking w.r.t. x for each such prefix. The maximum over all these numbers gives $\mathrm{Inh}(x, v)$; the inhibitor value $\mathrm{Inh}(x, v)$ specifies how small the inhibitor weight $I(p, l(v))$ may minimally be without prohibiting the event v.

Definition 12 (Inhibitor value). *Let L be a stratified language, x be a token flow function of L and v be an event of an LSO $\mathcal{S} \in L$. The* inhibitor value $\mathrm{Inh}(x, v)$ *of v w.r.t. x is defined by $Inh(x, v) = \max\{m_{\mathcal{S}'}(x) \mid \mathcal{S}' \text{ is prefix of } v \text{ w.r.t. } \mathcal{S}\}$.*

Figure 4 shows the token flow function from Figure 3 supplemented with the inhibitor values of all nodes (depicted in circles attached to the nodes). For example, consider the c-labelled node of the first LSO (from left). This node has four prefixes: the empty prefix with final marking 0, two prefixes consisting of one a-labelled node each with final marking 1 and a prefix with both a-labelled nodes and final marking 2.

Having determined $\mathrm{Inh}(x, v)$ for all nodes v of all LSOs in L one can specify the minimal inhibitor weight $I(p, t)$ from p to some transition t such that no t-labelled event

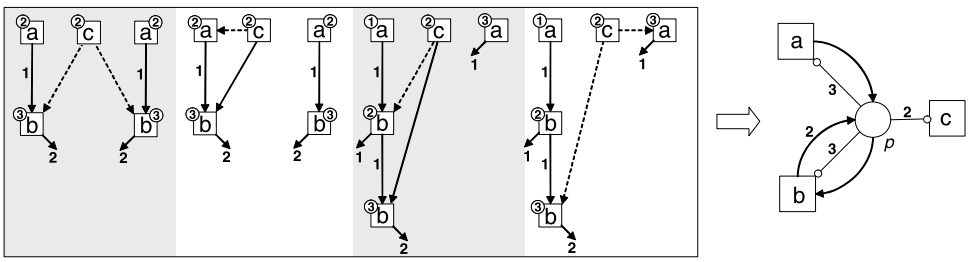

Fig. 4. The token flow function from Figure 3 supplemented with the inhibitor values of all LSO nodes and the feasible place corresponding to the respective region with minimal inhibitor weights

is prohibited by the supremum of all $\mathrm{Inh}(x,v)$ for events v labelled by t. This leads to $I_{min}(p,t)$ because the fact that no such t-labelled event is prohibited by the inhibitor weight $I(p,t)$ exactly describes that the place p is still feasible with this inhibitor weight $I(p,t)$ (instead of ω): $I_{min}(p,t) = sup(\{\mathrm{Inh}(x,v) \mid v \in V_L, l(v) = t\} \cup \{0\})$, where $V_L = \bigcup_{(V,\prec,\sqsubset,l)\in L} V$ is the set of all nodes of L. That means we calculate the inhibitor values of all nodes (over all LSOs of L) w.r.t. a given token flow function x using the method described above. The suprema of all inhibitor values of equally labelled nodes lead to the minimal inhibitor weights defining a feasible place w.r.t. L which corresponds to x. These minimal inhibitor weights $I(p,t) = I_{min}(p,t)$ represent the strongest behavioural restriction through inhibitor arcs for the place p defined by x guaranteeing the feasible-property. Thus *regions of stratified languages* w.r.t. pti-nets are defined by token flow functions x (defining p/t-net places) attached with inhibitor weight mappings $\mathbf{I} : T \to \mathbb{N} \cup \{\omega\}$ determining an inhibitor weight to every transition $t \in T$ which exceeds $I_{min}(p,t)$:

Definition 13 (Region). *A* region *of a stratified language L with labels T w.r.t. pti-nets is a tuple $r = (x, \mathbf{I})$ where x is a token flow function of L and $\mathbf{I} : T \to \mathbb{N} \cup \{\omega\}$ is a mapping assigning inhibitor weights to all transitions satisfying $\mathbf{I}(t) \geq sup(\{\mathrm{Inh}(x,v) \mid v \in V_L, l(v) = t\} \cup \{0\})$.*

The place p_r (in a net with transition set T) corresponding to a region $r = (x, \mathbf{I})$ of L *is defined by the flow weights and the initial marking of the place p_x corresponding to the token flow function x (i.e. $W(l(v), p_r) = \mathrm{Out}_{\mathrm{lpo}}(v,x)$, $W(p_r, l(v)) = \mathrm{In}_{\mathrm{lpo}}(v,x)$ and $m_0(p_r) = \mathrm{Out}_{\mathrm{lpo}}(v_0,x)$ for LPOs lpo underlying LSOs in L) and the inhibitor weights $I(p_r,t) = \mathbf{I}(t)$ for $t \in T$.*

The token flow function x in Figure 4 together with the mapping $\mathbf{I}$ given by $\mathbf{I}(a) = 3, \mathbf{I}(b) = 3, \mathbf{I}(c) = 2$ defines a region $r = (x, \mathbf{I})$. In fact this is the respective region with minimal inhibitor weights, i.e. $r' = (x, \mathbf{I}')$ is also a region if $\mathbf{I}' \geq \mathbf{I}$ but no region if $\mathbf{I}' \not\geq \mathbf{I}$. On the right the feasible place p corresponding to r is depicted.

The main theorem of this paper showing the consistency of the above regions definition now states (1) and (2) (compare Section 3) in this setting. Its proof essentially uses the definition of the enabledness of an LSO via the enabledness of its linearizations. According to the following lemma the enabledness of an event after some prefix of an LSO can be examined on the set of its linearizations.

Lemma 2. *Let $\mathcal{S} = (V, \prec, \sqsubset)$ be an so-structure, $V' \subseteq V$ and $v \in V$. Then V' defines a prefix of v w.r.t. $\mathcal{S}$ if and only if there is a linearization $\mathcal{S}' \in lin(\mathcal{S})$ such that V' defines a prefix of v w.r.t. $\mathcal{S}'$.*

Proof. The **if**-statement clearly follows from $\mathcal{S}' \supseteq \mathcal{S}$.

For the **only if**-statement we construct a sequence of event-sets $V_1 \ldots V_n$ with $V = V_1 \cup \ldots \cup V_n$ defining $\mathcal{S}'$ through $\prec_{\mathcal{S}'} = \bigcup_{i<j} V_i \times V_j$ and $\sqsubset_{\mathcal{S}'} = ((\bigcup_i V_i \times V_i) \cup \prec_{\mathcal{S}'}) \setminus id_V$ as follows: $V_1 = \{v \in V' \mid \forall v' \in V' : v' \not\prec v\}$, $V_2 = \{v \in V' \setminus V_1 \mid \forall v' \in V' \setminus V_1 : v' \not\prec v\}$ and so on, i.e. we define $V_i \subseteq V'$ as the set of nodes $\{v \in V' \setminus (\bigcup_{j=1}^{i-1} V_j) \mid \forall v' \in V' \setminus (\bigcup_{j=1}^{i-1} V_j) : v' \not\prec v\}$ which are minimal w.r.t. the restriction of $\prec$ onto the node set $V' \setminus (\bigcup_{j=1}^{i-1} V_j)$, as long as $V' \setminus (\bigcup_{j=1}^{i-1} V_j) \neq \emptyset$. Then continue with the same procedure on $V \setminus V' = V \setminus (\bigcup_{j=1}^{i} V_j)$, i.e. $V_{i+1} = \{v \in$

$V \setminus (\bigcup_{j=1}^{i} V_j) \mid \forall v' \in V \setminus (\bigcup_{j=1}^{i} V_j) : v' \not\prec v\}$ and so on. By construction V' is a prefix (of v) w.r.t. $\mathcal{S}'$. A straightforward computation also yields $\mathcal{S}' \in lin(\mathcal{S})$.

Theorem 2. *Given a stratified language L with set of labels T:* **(1)** *Every place corresponding to a region of L is feasible w.r.t. L and* **(2)** *every feasible place w.r.t. L is corresponding to a region of L.*

Proof. **(1):** Let p be corresponding to a region $r = (x, \mathbf{I})$ of L. We have to show that $\mathcal{S} \in L$ is enabled w.r.t. the one-place net (N, m_0) having p as its only place. Since x is a token flow function (called region in [10]) of the partial language L' underlying L the main result of [10] tells us that the LPO $\mathrm{lpo}_{\mathcal{S}} \in L'$ underlying $\mathcal{S}$ is enabled w.r.t. the place p_x corresponding to x. Consequently also $\mathcal{S}$ (since $lin(\mathcal{S}) \subseteq lin(\mathrm{lpo}_{\mathcal{S}})$)) is enabled w.r.t. p_x. In order to show that $\mathcal{S}$ is enabled w.r.t. p (differing from p_x only in the inhibitor weights), we consider a sequence of transition steps $\sigma = \tau_1 \ldots \tau_n$, whose underlying LSO $\mathcal{S}_\sigma$ is a linearization of $\mathcal{S}$. We have to show that σ is an enabled step occurrence sequence of (N, m_0). For this, we show inductively that if $\sigma_k = \tau_1 \ldots \tau_k$ is an enabled step occurrence sequence, then τ_{k+1} is a transition step enabled in the marking m reached after the execution of σ_k for $0 \leqslant k \leqslant n-1$. The above considerations ($\mathcal{S}$ enabled w.r.t. p_x) already imply the first condition of Definition 2 that $m(p) \geq \sum_{t \in \tau_{k+1}} \tau_{k+1}(t) W(p,t)$. It remains to verify the condition of Definition 2 that $m(p) \leq I(p,t)$ for each transition $t \in \tau_{k+1}$. If $\mathcal{S}_{\sigma_k} = (V_k, \prec_k, \sqsubset_k, l_k)$ is the LSO underlying σ_k and $\mathcal{S}_\sigma \supseteq \mathcal{S}$ is the LSO underlying σ, then $\mathcal{S}_{\sigma_k}$ is a prefix of an event $v \in V$ with $l(v) = t$ w.r.t. $\mathcal{S}_\sigma$. By Lemma 2, V_k also defines a prefix $\mathcal{S}_k$ of v w.r.t. $\mathcal{S}$. It is enough to show that $m(p) = m_{\mathcal{S}_k}(x)$, since $m_{\mathcal{S}_k}(x) \leq \mathrm{Inh}(x,v) \leq \mathbf{I}(l(v)) = I(p,t)$ (Definitions 12 and 13): $m(p) = m_0(p) - \sum_{i=1}^{k} \sum_{t \in \tau_i} \tau(t)(W(p,t) - W(t,p)) = m_0(p) - \sum_{v \in V_k} (W(p, l(v)) - W(l(v), p)) = m_{\mathcal{S}_k}(x)$ (compare the remarks to Definition 11).

(2): Let p be feasible w.r.t. L. Then, by Lemma 1 the place p' underlying p is feasible w.r.t. the partial language L' underlying L. The main result of [10] now states that there is a token flow function x of L' (called region in [10]) generating p'. We show now that $r = (x, I(p,\cdot))$ is a region of L (according to Definition 13). The first part that x is a token flow function of L is clear since x is a token flow function of L'. It remains to show $I(p,t) \geq sup(\{\mathrm{Inh}(x,v) \mid v \in V_L, l(v) = t\} \cup \{0\})$. For this let $v \in V$ for $\mathcal{S} = (V, \prec, \sqsubset, l) \in L$ with $l(v) = t$ and $\mathcal{S}'$ be a prefix of v defined by V'. We have to show that $m_{\mathcal{S}'}(x) \leq I(p,t)$ (compare Definition 12). By Lemma 2 there is a linearization $\mathcal{S}_{lin}$ of $\mathcal{S}$ such that V' also defines a prefix $\mathcal{S}'_{lin}$ of v w.r.t. $\mathcal{S}_{lin}$. Since $\mathcal{S}$ is enabled w.r.t. the one-place net (N, m_0) having p as its only place, there is an enabled step occurrence sequence $\sigma = \tau_1 \ldots \tau_n$ of (N, m_0) whose underlying LSO $\mathcal{S}_\sigma$ equals $\mathcal{S}_{lin}$. Since prefixes are downward $\sqsubset$-closed, a prefix $\sigma' = \tau_1 \ldots \tau_m$ $(m < n)$ of σ with $l(v) = t \in \tau_{m+1}$ must exist which corresponds to $\mathcal{S}'_{lin}$. In other words, the LSO $\mathcal{S}_{\sigma'}$ underlying σ' equals $\mathcal{S}'_{lin}$. It is enough to show now that $m(p) = m_{\mathcal{S}'}(x)$ for the marking m reached after the execution of σ' in (N, m_0), since $m(p) \leq I(p,t)$ for each transition $t \in \tau_{m+1}$. The necessary computation is as in (1).

Thus the set of all feasible places and therefore a solution for the synthesis problem can be derived from the set of regions.

5 Conclusion

In this paper we introduced the notion of regions for a (possibly infinite) set of LSOs – called stratified language – describing the behaviour of a pti-net. Given a stratified language L, using such regions allows to define the saturated feasible pti-net (N, m_0) w.r.t. L. The set of executions $L(N, m_0)$ of (N, m_0) includes L and is as small as possible with this property.[5] Thus, the contribution of this paper is to solve the synthesis problem satisfactory from the theoretical point of view (for the considered setting). Practical algorithmic considerations are a topic of further research (see also below).

The presented approach carries over to the a-posteriori semantics of pti-nets, whose non-sequential scenario-based behaviour is given by LPOs, i.e. by partial languages. To define regions for partial languages w.r.t. pti-nets, one can analogously start with regions of the partial language from [10] not specifying inhibitor arcs and then assign inhibitor values to each node. Now, these inhibitor values are determined as maxima over all final markings of classical prefixes of nodes of an LPO, where one has to use a slightly different definition of final markings. It is moreover possible to adapt the presented definition of regions to other less general inhibitor net classes, such as p/t-nets with unweighted inhibitor arcs and elementary nets with inhibitor arcs. Thereby in the case of elementary nets one additionally has to regard that a place defined by a region must not carry more than one token in each intermediate state of an LSO. This can be ensured by only allowing final markings of prefixes $\leqslant 1$ (that means by an analogous mechanism as used for the definition of inhibitor arcs). For step transition systems and stratified languages which produce the same language of step sequences, it would be interesting to compare our (adapted) definition of regions for elementary nets with inhibitor arcs and the definition of regions from [11,12]. The relation is not obvious since several different step transition systems may define the same language of step sequences. In general the ideas presented in this paper should also be useful for the consideration of the synthesis problem of other so-structure based net classes (such as nets with read arcs, priorities, reset arcs, etc.) as well as net classes conceptually similar to inhibitor nets (e.g. elementary nets and nets with capacities).

One of course is interested in practical algorithmic solutions of the synthesis problem. Basically the regions approach has the problem that there is an infinite number of feasible places respectively regions of a stratified language. Our recent publication [9] tackles this problem for finite partial languages and p/t-nets, i.e. a special case of the setting in [10]. Thereto the definition of token flow function is translated into a finite integer system of homogenous inequations $\mathbf{A} \cdot \mathbf{x} \geq 0$: The finite vector $\mathbf{x}$ represents the token flow function and the inequations reflect the conditions of Definition 9 and ensure positive token flows ($\mathbf{x} \geq 0$). It is shown that one can calculate a finite set of basis solutions of this system which defines a set of places spanning all feasible places.[6] That means the net consisting only of these finite, algorithmically determinable set of places

[5] Note that such a region based approach is not appropriate to find a pti-net (N, m_0) such that $L(N, m_0) \subseteq L$ and $L(N, m_0)$ is as large as possible.

[6] An alternative approach is to compute finite many regions which "separate" specified behaviour from not specified behaviour. It is possible to deduce appropriate separation properties from the mentioned algorithm. Such an approach leads to a different finite representation of the saturated feasible net.

has the same set of executions as the saturated feasible net. Furthermore an algorithm testing if this net has the behaviour specified by the finite partial language is shown. In the setting of this paper a similar approach for the effective synthesis of pti-nets from finite stratified languages is possible, i.e. it is possible to calculate finitely many basis regions spanning the set of all regions (using an adequate inequation system). The formal evolution and proofs for this approach including complexity issues are one of our recent research projects in this topic.

But this approach still leaves the problem that it does not work for infinite stratified languages. For algorithmic purposes an infinite stratified language first has to be finitely represented. This problem is strongly connected to the similar problem in the case of p/t-nets and partial languages which is one of our central current research fields.

References

1. Badouel, E., Darondeau, P.: On the synthesis of general petri nets. Technical Report 3025, Inria (1996)
2. Busi, N., Pinna, G.M.: Synthesis of nets with inhibitor arcs. In: Mazurkiewicz, A.W., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 151–165. Springer, Heidelberg (1997)
3. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Hardware and petri nets: Application to asynchronous circuit design. In: Nielsen, M., Simpson, D. (eds.) ICATPN 2000. LNCS, vol. 1825, pp. 1–15. Springer, Heidelberg (2000)
4. Ehrenfeucht, A., Rozenberg, G.: Partial (set) 2-structures. part i: Basic notions and the representation problem. Acta Inf. 27(4), 315–342 (1989)
5. Ehrenfeucht, A., Rozenberg, G.: Partial (set) 2-structures. part ii: State spaces of concurrent systems. Acta Inf. 27(4), 343–368 (1989)
6. Janicki, R., Koutny, M.: Semantics of inhibitor nets. Inf. Comput. 123(1), 1–16 (1995)
7. Juhás, G., Lorenz, R., Mauser, S.: Complete process semantics for inhibitor nets. In: Proceedings of ICATPN 2007 (2007)
8. Kleijn, H.C.M., Koutny, M.: Process semantics of general inhibitor nets. Inf. Comput. 190(1), 18–69 (2004)
9. Lorenz, R., Bergenthum, R., Mauser, S., Desel, J.: Synthesis of petri nets from finite partial languages. In: Proceedings of ACSD 2007 (2007)
10. Lorenz, R., Juhás, G.: Towards synthesis of petri nets from scenarios. In: Donatelli, S., Thiagarajan, P.S. (eds.) ICATPN 2006. LNCS, vol. 4024, pp. 302–321. Springer, Heidelberg (2006)
11. Pietkiewicz-Koutny, M.: The synthesis problem for elementary net systems with inhibitor arcs. Fundam. Inform. 40(2-3), 251–283 (1999)
12. Pietkiewicz-Koutny, M.: Synthesising elementary net systems with inhibitor arcs from step transition systems. Fundam. Inform. 50(2), 175–203 (2002)
13. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: A survey of issues and approaches. Data Knowl. Eng. 47(2), 237–267 (2003)
14. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)
15. Zhou, M., Cesare, F.D.: Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Kluwer, Dordrecht (1993)