Partial Order Semantics of Types of Nets

Robert Lorenz[1], Gabriel Juhás[2], and Sebastian Mauser[3]

[1] Department of Computer Science, University of Augsburg, Germany
robert.lorenz@informatik.uni-augsburg.de
[2] Faculty of Electrical Engineering and Information Technology
Slovak University of Technology, Bratislava, Slovakia
gabriel.juhas@stuba.sk
[3] Department of Applied Computer Science
Catholic University of Eichstätt-Ingolstadt, Germany
sebastian.mauser@ku-eichstaett.de

Abstract. In this paper we define partial order semantics of types of nets. Types of nets are a parametric definition of Petri nets originally developed for a general presentation of the synthesis of Petri nets from (step) transition systems. Partial order semantics of a concrete net (of a certain type) usually are given by the set of labelled partial orders (LPOs) enabled w.r.t. the net. For classical place/transition nets there are several equivalent characterizations of enabled LPOs. We discuss in which way the general notion of types of nets has to be restricted such that these characterizations can also be formulated for nets of such type. In particular we consider under which requirements enabled LPOs can be defined through token flows, which have been proven to be useful for efficient synthesis and verification of Petri nets. The presented concepts form the basis for a general presentation of the synthesis of Petri nets from sets of LPOs.

1 Introduction

The so-called synthesis problem of Petri nets consists of deciding whether there exists some unlabelled net which exhibits a given behaviour and in the positive case constructing such net. The behavioural description can be given by a (step) transition system [4] or by a language, i.e. a set, of (step) occurrence sequences [4] or partially ordered executions [13, 5]. The synthesis problem has been solved in the literature for various classes of nets, ranging from elementary nets [9] to place/transition nets [4] (p/t-nets) and inhibitor nets [14, 13], w.r.t. various kinds of behavioural descriptions. The common principle for the synthesis is the idea of regions of behavioural descriptions, representing places of nets, which allow to reproduce the behaviour given by the specification. There are industrial applications of Petri net synthesis in hardware system design, control of manufacturing systems and workflow design (see e.g. [13] for references).

In [2] so called types of nets were introduced to present the synthesis of Petri nets from transition systems parametric w.r.t. the Petri net class. Types of nets are a parametric definition of Petri nets determining a net class by a transition system. This transition system represents all possible states of places of a net of the considered net class and all possible modifications of such states by transitions of the net. It is shown in [2] that, given a type of nets, regions of a transition system may be identified with the morphisms from the transition system to the transition system given by the considered type. All the earlier known synthesis results for transition systems may be retrieved as instances of the general result established in [2] by translating the considered net class into a type of nets. The types of nets definition was extended in [3] to include a step firing rule. In this richer context the concept of regions as morphisms is generalized to step transition systems leading to a comprehensive presentation of the synthesis of Petri nets from step transition systems. This presentation can also be adapted to cover the synthesis from languages of (step) occurrence sequences [4]. The types of nets definition was further extended in [1] to cover some specific finer behavioural aspects.

Partially ordered executions truly representing the concurrency of transition occurrences are often considered the most appropriate representation of behaviour of Petri net models of concurrent systems. In the case of p/t-nets, synthesis [13, 5] and verification [10] based on partially ordered executions given by labelled partial orders (LPOs) proved useful and nicely applicable for system design (see e.g. research in the context of VipTool at viptool.ku-eichstaett.de). Therefore, we are interested in a generalization of these methods to types of nets, in particular including the development of a general presentation of the synthesis of Petri nets from languages of LPOs. For this it is necessary to first define partially ordered executions of nets parametric in their type. The aim of this paper is to develop and examine such partial order semantics of types of nets. The paper provides a comprehensive grounding in the theory of this topic, but beyond no effective algorithms are discussed.

For classical p/t-nets there are several equivalent characterizations of partially ordered executions given by LPOs [12, 10, 5], namely several variants of the classical definition of so called *enabled LPOs*, LPOs fulfilling the so called *token flow property* and LPOs sequentializing LPOs underlying a *process net*. The classical definitions of enabled LPOs either consider each step sequentialization, each co-set or each cut of the LPO. In this paper we generalize the various variants of the definition of enabled LPOs as well as the token flow property of LPOs to the setting of types of nets. In particular, we discuss in which way the general notion of types of nets has to be restricted such that these characterizations of partially ordered executions can be formulated. It turns out that the restrictions introduced here do not rule out any important extension of classical p/t-nets whose non-sequential behaviour can be described by LPOs.

We are especially interested in the token flow property [10], because in the case of p/t-nets this property proved useful for efficient synthesis and verification. The token flow property nicely represents the actual dependencies of transition occurrences of a Petri net by the flow of tokens between the transition occurrences. We do not provide a characterization of partially ordered runs based on process nets of types of nets in this paper. But the concepts of process nets and token flows are roughly speaking only dialects of the same principle. As known from p/t-nets [10], token flows just abstract from the individuality of conditions of a process net by encoding their relations in a token flow function. Following these ideas, the considerations about token flows in this paper in a natural way also lead to a definition of process nets of types of nets.

Although the main motivation for the considerations in this paper is to prepare the ground for a general presentation of the synthesis of Petri nets from LPOs parameterized by net types, the presented theory about enabled LPOs for types of nets is of value on its own. It gives precise insights into the limitations of expressing non-sequential behaviour of concurrent system models through LPOs by identifying appropriate conditions. In particular, the existence of efficient verification and synthesis methods w.r.t. partial order semantics of Petri nets (which are based on token flows in the case of p/t-nets) is clarified on an abstract level. Such methods can be developed for net types similar to those of p/t-nets; that means this paper can be seen as the main step towards the mentioned general presentation of synthesis of Petri nets. In particular, the different characterizations of partially ordered executions of types of nets can straightforwardly be used to define regions for languages of LPOs, where each characterization yields another region definition. Given such a region definition, synthesis algorithms may be developed for some concrete type of net following standard computation principles known from the theory of regions for p/t-nets [13].

Finally, it remains to mention that there are also alternative parametric Petri net definitions such as algebraic $(\mathcal{M}, \mathcal{I})$-nets [7], Petri nets over a group [8] or token free nets [15]. Since we are in particular interested in synthesis, we here consider types of nets. The local flavour of the type of nets definition makes it an appropriate uniform definition as a basis for region theories as shown in previous works [1–4]. In [4] it is discussed that the nets in [8] are more problematic for a general presentation of synthesis. Algebraic definitions [7] do not have a local focus, and are thus complicated in the setting of region-based approaches. Token-free nets [15] are very similar to types of nets, but types of nets are better tuned for synthesis.

The remainder of the paper is organized as follows: In Section 2 we introduce types of nets. In Section 3 we discuss the notion of enabled LPOs for types of nets and in Section 4 we develop the token flow property for types of nets.

Because of lack of space we omitted all proofs and illustrating examples from the main text of the paper and included them in a technical report [11] on our homepage.

2 Types of Nets

In this section we introduce the basic definitions of types of nets [2, 3]. We start with some mathematical notions.

By $\mathbb{N}$ we denote the *nonnegative integers* and by ω an infinite value, i.e. $n < \omega$ for $n \in \mathbb{N}$, and $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$. For a set A, by id_A we denote the *identity function* on A, by 0_A the *zero function* on A and by 1_X the *characteristic function* of a subset $X \subseteq A$.

A triple $(A, +, 0)$, where A is a set, $+ : A \times A \to A$ is a binary operation on A and $0 \in A$, is called an *abelian monoid* if the following holds: $\forall a, b, c \in A : \ (a + b) + c = a + (b + c)$ (*associativity*), $\forall a \in A : \ a + 0 = 0 + a = a$ (*identity element*) and $\forall a, b \in A : \ a + b = b + a$ (*commutativity*). For $n \in \mathbb{N}$ and $a_1 \ldots a_n, a \in A$, $\sum_{i=1}^{n} a_i$ denotes the element $a_1 + \ldots + a_n \in A$ and $n \cdot a$ denotes the element $\sum_{i=1}^{n} a \in A$. A *morphism* between two monoids $(A, +, 0)$ and $(A', +', 0')$ is a function $f : A \to A'$ such that $f(a + b) = f(a) +' f(b)$ for all $a, b \in A$ and $f(0) = 0'$.

The *free abelian monoid* over the set of generators A is given by the abelian monoid $(\mathbb{N}^A, +, 0_A)$ of the set of *multi-sets* over A. A multi-set over A is a function $m : A \to \mathbb{N} \in \mathbb{N}^A$. Addition $+$ on multi-sets is defined as usual by $(m+m')(a) = m(a)+m'(a)$. We write $a \in m$ if $m(a) > 0$. We do not distinguish between a subset $X \subseteq A$ and the multi-set 1_X. The multi-set $1_{\{a\}}$ for $a \in A$ is often abbreviated by a. Each multi-set $m \in \mathbb{N}^A$ can be written as $\sum_{a \in A} m(a)a$. An order relation $\leqslant$ on $\mathbb{N}^A$ is defined by $m \leqslant m' \iff \forall a \in A : \; m(a) \leqslant m'(a)$. We denote $m < m'$ if $m \leqslant m'$ and $m \neq m'$. If $m \leqslant m'$ then $m' - m$ is the uniquely defined multi-set such that $m + (m' - m) = m'$ (given by $(m' - m)(a) = m'(a) - m(a)$ for $a \in A$).

Types of nets [2, 3] allow a parametric definition of Petri nets (in [2] only sequential semantics and in [3] step semantics is considered) covering most of the relevant Petri net classes, including e.g. elementary nets, p/t-nets and p/t-nets with weighted inhibitor (or read) arcs (called pti-nets) equipped with the a-priori as well as the a-posteriori semantics (see [11], Example 1). Each type determines a Petri net class.

We consider Petri nets as usual consisting of a set of places P and a set of transitions T. The states (markings) of the net and the occurrence rule for (steps of) transitions is defined parametric w.r.t. the considered type of nets. A type of nets τ is a deterministic (step) transition system over a set of local events LE with set of states LS and transition relation τ [3]. The marking of a place is given by a local state from LS. The occurrence of transitions is determined by local events from LE assigned to every pair $(p, t) \in P \times T$ by a weight function W. Roughly speaking, a step of transitions is enabled to occur in a marking, if for each place the corresponding step of local events is enabled w.r.t. τ in the local state given by the marking.

Definition 1 (Type of nets). *A* type of nets *is a triple $\tau = (LS, LE, \tau)$, where LS is a set of* local states, *LE is an abelian monoid $(LE, +, 0)$ of* local events *and $\tau \subseteq LS \times LE \times LS$ defines the* partial action *of local events on local states. The partial action is required to be deterministic, i.e. $(s, e, s'), (s, e, s'') \in \tau \Longrightarrow s' = s''$.*

As usual $(s, e, s') \in \tau$ is abbreviated by $s \xrightarrow{e} s'$. A local event e is enabled *in a local state s, if $\exists s' : s \xrightarrow{e} s'$, denoted by $s \xrightarrow{e}$. The identity event 0 is enabled in each local state s with $s \xrightarrow{0} s$.*

Definition 2 (Net of type τ). *A* (Petri) net of type τ *is a triple $N = (P, T, W)$, where P is a finite set of* places, *T is a finite set of* transitions *and $W : P \times T \to LE$ is a* weight function. *A* marking *of N is a mapping $m : P \to LS$. A marked net of type τ is a structure (P, T, W, m_0), where (P, T, W) is a net of type τ and $m_0 : P \to LS$ is an* initial marking.

Definition 3 (Occurrence rule). *Let $N = (P, T, W)$ be a net of type $\tau = (LS, LE, \tau)$. A multi-set of transitions $x \in \mathbb{N}^T$, called a* step of transitions, *is* enabled (to occur) in a marking *m of N if for each place p:*

$$\exists s' \in LS : (m(p), \sum_{t \in T} x(t) \cdot W(p, t), s') \in \tau.$$

In this case, its occurrence leads to the marking m' uniquely defined by $m'(p) = s'$. We write $m \xrightarrow{x} m'$ to denote that x is enabled in m and that its occurrence leads to m'.

A finite sequence of steps $\sigma = x_1 \ldots x_n, \; n \in \mathbb{N}$, is called a *step (occurrence) sequence enabled in a marking m and leading to the follower marking m_n*, denoted by $m \xrightarrow{\sigma} m_n$, if there exists a sequence of markings $m_1, \ldots, m_n$ such that $m \xrightarrow{x_1} m_1 \xrightarrow{x_2} \ldots \xrightarrow{x_n} m_n$. The marking m_n is said to be *reachable from the marking m*.

3 Enabled LPOs in Types of Nets

In this section, for the first time partial order semantics of types of nets is introduced. The partial order semantics of a net of some type is given by the set of labelled partial orders (LPOs) *enabled* in this net. For p/t-nets, there are several different characterizations of enabled LPOs. It turns out that for each characterization we have to restrict the very general definition of types of nets adequately in order to adapt the characterization to the introduced parametric net definition. The considerations give new detailed insights into the notion of LPOs enabled w.r.t. Petri nets.

We first recall basic notions of partial orders. A *partial order* is a pair $(V, <)$, where V is a finite *set of nodes* and $<\subseteq V \times V$ is an irreflexive and transitive binary relation over V called the *set of arcs*. A node $v \in V$ such that $\forall v' \in V : v' \not< v$ $(v' \not> v)$ is called *minimal (maximal)*. Two nodes $v, v' \in V$, are called *independent* if $v \not< v'$ and $v' \not< v$. By $\mathrm{co}_< \subseteq V \times V$ we denote the set of all pairs of independent nodes of V. A *co-set* is a subset $C \subseteq V$ fulfilling $\forall v, v' \in C : v \,\mathrm{co}_< v'$. A *cut* is a maximal co-set. For a co-set C of a partial order $(V, <)$ and a node $v \in V \setminus C$ we write $v < C$ $(C < v)$, if $v < v'$ $(v' < v)$ for some $v' \in C$, and $v \,\mathrm{co}_< C$, if $v \,\mathrm{co}_< v'$ for all $v' \in C$. If $\mathrm{co}_< = id_V$ then $(V, <)$ is called *total*. If $\mathrm{co}_<$ is transitive then $(V, <)$ is called to be in *step form*.

Given a set $V' \subseteq V$, a partial order $(V', <')$ is called a *prefix* of $(V, <)$ if $(v' \in V' \wedge v < v') \Longrightarrow (v \in V')$ and $<' = <|_{V' \times V'}$. In this case the prefix $(V', <')$ is said to be *defined by* V'. Given a co-set C of $(V, <)$, a *prefix of* C is a prefix $(V', <')$ of $(V, <)$ fulfilling $\{v \in V \mid v < C\} \subseteq V'$ and $C \cap V' = \emptyset$. A cut has only one prefix. 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 $<_1 \subseteq <_2$. If po_2 is total it is called a *linearization* and if po_2 is in step form it is called a *step sequentialization*.

Definition 4 (Labelled partial order). *A* labelled partial order *(LPO) is a triple* $\mathrm{lpo} = (V, <, l)$*, where* $(V, <)$ *is a partial order, and* $l : V \to T$ *is a* labelling function *with* set of labels T.

We use the above notations defined for partial orders also for LPOs. We consider LPOs only up to isomorphism. In the case that T is a set of transitions, nodes in V model transition occurrences, called *events*. The arcs of lpo model a dependency relationship between events and independency of events expresses concurrency. For a set of events $V' \subseteq V$, we define the step of transitions $|V'|_l \subseteq \mathbb{N}^T$ by $|V'|_l(t) = |\{v \in V' \mid l(v) = t\}|$. If lpo is in step form, i.e. $V = \bigcup_{i=1}^n V_i$ and $<= \bigcup_{i<j} V_i \times V_j$, we call the step sequence $\sigma_{\mathrm{lpo}} = |V_1|_l \ldots |V_n|_l$ *associated to* lpo. Given a step sequentialization (linearization) lpo' of lpo, we call $\sigma_{\mathrm{lpo}'}$ a *step sequence (linear sequence) of* lpo.

An LPO over a set of labels T can be enabled or not enabled in a marked net $N = (P, T, W, m_0)$ of type τ. An enabled LPO models valid behaviour of the net. That means, an LPO is enabled in a net, if the events of the LPO can occur in the net respecting the concurrency and dependency relations of the LPO. In particular, it must be possible that concurrent events can occur in any order as well as in one step, while ordered events have to occur in the respective order. Thus formally, enabledness of an LPO means that every step sequence of the LPO is enabled in the given net (observe that every sequentialization and prefix of an enabled LPO is also enabled).

Definition 5 (Enabledness). *Let $N = (P, T, W, m_0)$ be a marked net of type $\tau = (LS, LE, \tau)$. An LPO* $\mathrm{lpo} = (V, <, l)$ *with* $l : V \to T$ *is called* enabled (to occur) in N *if for every step sequentialization* lpo' *of* lpo*, the associated step sequence* $\sigma_{\mathrm{lpo}'}$ *is enabled to occur in N. The marking m' given by* $m_0 \xrightarrow{\sigma_{\mathrm{lpo}'}} m'$ *is a* final marking *of* lpo.

Note that different step sequences of an enabled LPO may produce different final markings of the LPO (see [11], Example 2).

In the case of p/t-nets, enabled LPOs can be equivalently characterized through requiring that for each co-set C of the LPO and each prefix of C, the final marking of this prefix enables the step of transitions $|C|_l$. This cannot analogously be formulated for types of nets for two reasons. First the final marking of a prefix is not unique, i.e. for types of nets each final marking of the prefix should enable $|C|_l$. Second, a final marking is only defined if the prefix itself is enabled – for p/t-nets a final marking can also be defined by the events in the prefix whether or not it is enabled. Both problems can be solved through restricting the definition of types of nets adequately. But this is discussed later on. Without a further restriction we can state:

Lemma 1. *Let $N = (P, T, W, m_0)$ be a marked net of type $\tau = (LS, LE, \tau)$. An LPO* $\mathrm{lpo} = (V, <, l)$ *with $l : V \to T$ is enabled in N if and only if for every non-empty co-set C of* lpo *and every prefix* $\mathrm{lpo}' = (V', <', l')$ *of C,* lpo' *is enabled and $|C|_l$ is enabled in each final marking of* lpo'.

This characterization can be simplified for nets, whose enabled LPOs always have a unique final marking. There is a nice and natural property of nets ensuring such unique final markings. We call it *weak intermediate state property (WISP)*. A marked net N fulfills the WISP if all reachable markings m, m', m'' together with steps of transitions x_1, x_2 satisfy

$$m \xrightarrow{x_1+x_2} m' \wedge m \xrightarrow{x_1 x_2} m'' \Longrightarrow m' = m''$$

This means, if an enabled step can be divided into a sequence of two steps which is also enabled, this sequence yields the same marking.

Lemma 2. *Let $N = (P, T, W, m_0)$ be a marked net of type $\tau = (LS, LE, \tau)$ satisfying the WISP. Then each LPO* $\mathrm{lpo} = (V, <, l)$ *enabled in N has a unique final marking.*

The characterization of enabled LPOs by co-sets now reads as follows showing that only one step sequence has to be considered for each prefix of each co-set.

Lemma 3. *Let $N = (P, T, W, m_0)$ be a marked net of type $\tau = (LS, LE, \tau)$ satisfying the WISP and let* $\mathrm{lpo} = (V, <, l)$ *with $l : V \to T$ be an LPO. If* lpo *is enabled in N, then for every non-empty co-set C of* lpo *and every prefix* $\mathrm{lpo}' = (V', <', l')$ *of C, every step sequence σ of* lpo' *is enabled and $|C|_l$ is enabled in the follower marking of σ. If for every non-empty co-set C of* lpo *and every prefix* $\mathrm{lpo}' = (V', <', l')$ *of C, there is an enabled step sequence σ of* lpo' *such that $|C|_l$ is enabled in the follower marking of σ, then* lpo *is enabled in N.*

Unique final markings of enabled LPOs are required in Lemma 3 (by the WISP), because there are simple examples showing that otherwise the second statement of Lemma 3 does not hold (see [11], Example 3).

It is interesting to have a requirement for types of nets which ensures unique final markings of enabled LPOs for all nets of a type satisfying this requirement. The WISP can also be formulated for types of nets. A type of nets (LS, LE, τ) fulfills the WISP if all combinations of local states s, s', s'' and local events e_1, e_2 satisfy

$$s \xrightarrow{e_1+e_2} s' \wedge s \xrightarrow{e_1e_2} s'' \Longrightarrow s' = s''$$

Lemma 4. *If (LS, LE, τ) is a type of nets satisfying the WISP then each marked net of this type satisfies the WISP. If (LS, LE, τ) does not satisfy the WISP then there is a marked net of this type not satisfying the WISP.*

Consider now the case of two different enabled LPOs having exactly the same multi-set of transition occurrences. Although each of these LPOs has a unique final marking if WISP holds, these final markings need not be equal (see [11], Example 4). Types of nets can be restricted in such a way, that final markings of enabled LPOs only depend from the numbers of transition occurrences, but not longer from the ordering. This simplifies testing enabledness of LPOs, because then obviously in the second statement of Lemma 3 only the numbers of transition occurrences in lpo' need to be considered (instead of a step sequence of lpo'). A property generalizing the WISP in this sense is the *Parikh image property (PIP)*. The PIP can be formulated for nets and types of nets.

A marked net N fulfills the PIP if all reachable markings m, m', m'' together with steps of transitions $x_1, \ldots, x_n$ and $x'_1, \ldots, x'_m$ satisfy:

$$m \xrightarrow{x_1\ldots x_n} m' \wedge m \xrightarrow{x'_1\ldots x'_m} m'' \wedge x_1 + \ldots + x_n = x'_1 + \ldots + x'_m \Longrightarrow m' = m''.$$

A type of nets (LS, LE, τ) fulfills the PIP if all combinations of local states s, s', s'', a multi-set of local events u and two partitions $u = u_1 + \ldots + u_n = u'_1 + \ldots + u'_m$ of u satisfy for $e_i = \sum_{e \in u_i} u_i(e)e$ and $e'_i = \sum_{e \in u'_i} u'_i(e)e$:

$$s \xrightarrow{e_1\ldots e_n} s' \wedge s \xrightarrow{e'_1\ldots e'_m} s'' \Longrightarrow s' = s''.$$

This property is a typical feature of Petri nets. The property means that the marking reached after firing an enabled step sequence is determined by the number each single transition occurs in the step sequence (a generalization of the Parikh image). Note that the PIP implies the WISP.

Lemma 5. *If (LS, LE, τ) is a type of nets satisfying the PIP then each marked net of this type satisfies the PIP. If (LS, LE, τ) does not satisfy the PIP then there is a marked net of this type not satisfying the PIP.*

Lemma 6. *Let N satisfy the PIP and let $\mathrm{lpo} = (V, <, l), \mathrm{lpo}' = (V', <', l')$ be LPOs enabled in N with $|V|_l = |V'|_{l'}$. Then the final markings of lpo and lpo' coincide.*

In more restricted cases, the characterization of enabled LPOs by co-sets can further be simplified by not taking every co-set (and each prefix of the co-sets) into account, but only every cut (and the prefix of the cut). We formulate a sufficient condition for this. Most net classes exhibit the well-known *intermediate state property (ISP)* [3, 4] (which is stronger than the WISP, but incomparable with the PIP). A marked net N fulfills the ISP if for all reachable markings m, m', m'' and steps of transitions x_1, x_2:

$$m \xrightarrow{x_1+x_2} m' \Longrightarrow m \xrightarrow{x_1x_2} m'.$$

The ISP can also be formulated for types of nets. A type of nets (LS, LE, τ) fulfills the ISP if all combinations of local states s, s', s'' and all local events e_1, e_2 satisfy:

$$s \xrightarrow{e_1+e_2} s' \Longrightarrow s \xrightarrow{e_1 e_2} s'.$$

Lemma 7. *If (LS, LE, τ) is a type of nets satisfying the ISP then each marked net of this type satisfies the ISP. If (LS, LE, τ) does not satisfy the ISP then there is a marked net of this type not satisfying the ISP.*

Lemma 8. *Let $N = (P, T, W, m_0)$ be a marked net of type $\tau = (LS, LE, \tau)$ fulfilling the ISP and let $\mathrm{lpo} = (V, <, l)$ with $l : V \to T$ be an LPO. If lpo is enabled in N, then for every cut C of lpo, every step sequence σ of the prefix of C is enabled and $|C|_l$ is enabled in the follower marking of σ. If for every cut C of lpo, there is an enabled step sequence σ of the prefix of C such that $|C|_l$ is enabled in the follower marking of σ, then lpo is enabled in N.*

The ISP is necessary for this characterization in the sense that, if a net does not fulfill the ISP, but it fulfills the WISP, then the second statement of Lemma 8 does not hold (see [11], Example 5). In general, using LPOs as a behavioural model for nets satisfying the WISP but not the ISP, which means that a sequential decomposition of some enabled step of transitions is not enabled, is problematic.

Simple examples show that there are types of nets not satisfying the WISP, satisfying the WISP but not the ISP and the PIP, satisfying the ISP but not the PIP, satisfying the PIP but not the ISP, and satisfying both the PIP and the ISP (see [11], Example 6).

4 Token Flows in Types of Nets

In this section we generalize the notion of *token flows*, originally developed for p/t-nets, to types of nets. The existence of appropriate token flows of an LPO gives an equivalent characterization of enabledness.

The notion of token flows was introduced for p/t-nets in [10] as a compact representation of process nets. Namely, token flows abstract from the individuality of conditions of a process net and encode the flow relation of the process net by natural numbers. For each place a natural number is assigned to each arc of the LPO underlying a process net. Such a natural number assigned to an arc (e, e') represents the number of tokens produced by the event e and consumed by the event e' in the respective place. As shown in [10] an LPO is enabled in a p/t-net if and only if it fulfills the so called *token flow property*. This characterization of enabledness has in particular been successfully applied for the efficient verification [10] and the synthesis [5, 6] of p/t-nets.

In types of nets, markings given by the numbers of tokens in places are generalized to local states of places. In general, types of nets do not exhibit some kind of flow of local states based on ”consuming” and ”producing” local states by transition occurrences. Therefore, we define a subclass of types of nets, called *flow types of nets*. The main idea is to equip the set of local states with an appropriate algebraic structure, such that local states can be added and related to each other and such that there is a set of generators. Moreover, the sets of local events and local states are related through appropriate morphisms modelling a local flow. The partial action of local events on local states has to be consistent to the local flow. After the definition we will make plausible that each of these restrictions is actually necessary to allow a reasonable notion of token flow.

Definition 6. *A flow type of nets (τ, f) is a type of nets $\tau = (LS, LE, \tau)$ together with a flow map $f = (f_1, f_2) : LE \to LS \times LS$ fulfilling:*
- *LS is a free abelian monoid,*
- *f_1, f_2 are monoid morphisms,*
- *$(s, e, s') \in \tau \Longrightarrow f_1(e) \leq s \wedge s' = s - f_1(e) + f_2(e)$.*

The effect of the occurrence of concurrent transitions onto a local state is given by the effect of the corresponding local events. We assume that if a step of local events (which of course itself defines a local event) is enabled in a local state, then each of the local events in the step consumes a "part" of the local state, modelled by f_1, and produces a "part" of the follower local state, modelled by f_2. That means that the local state can be partitioned into several components (which are themselves local states) and that these components can be composed to the "whole" local state. Of course the composition of local states should be associative and commutative, since the effect of the occurrence of concurrent transitions onto a local state should be independent from any ordering of these concurrent transitions. Also a zero-flow, i.e. an identity local state is necessary. Thus, it is reasonable to assume the set of local states to be an abelian monoid. The need to partition local states also shows that generator local states are reasonable, i.e. to consider a free abelian monoid of local states. The mappings f_1 and f_2 are assumed to be morphisms to guarantee that the local states consumed and produced by a step of transitions are consistent to the local states consumed and produced by the single transitions in the step. The last property of the flow type of nets definition ensures that the change $(s, e, s') \in \tau$ of a local state s to a local state s' by a local event e is consistent with the flow $f(e)$ of e and that s' can be computed in this way. Later on, token flows in LPOs will be based on the flow map f.

All net classes we are interested in can be represented as flow types of nets by equipping the types of nets instantiation in a natural way with an appropriate flow map. Examples are p/t-nets, pti-nets equipped with the a-priori and the a-posteriori semantics as well as elementary nets (see [11], Example 7).

The requirements of the flow type of nets definition ensure the PIP introduced in the last section. In particular, the final marking of an enabled LPO is unique and solely depends on the numbers of transition occurrences in the LPO.

Lemma 9. *A flow type of nets (τ, f), $\tau = (LS, LE, \tau)$, fulfills the PIP.*

The final marking of an enabled LPO can be computed as follows.

Lemma 10. *The final marking m of an enabled LPO $\mathrm{lpo} = (V, <, l)$ of a marked net $N = (P, T, W, m_0)$ of a flow type (τ, f) is given by*

$$m(p) = m_0(p) + \sum_{v \in V} f_2(W(p, l(v))) - \sum_{v \in V} f_1(W(p, l(v))), p \in P.$$

Let $\mathrm{lpo} = (V, <, l)$ be an LPO. A *token flow function (on* lpo*)* is a function $x : < \to LS$ (LS a free abelian monoid). For $v \in V$, $In_x(v) = \sum_{v' < v} x(v', v)$ is the *intoken flow of v (w.r.t. x)*, and $Out_x(v) = \sum_{v < v'} x(v, v')$ is the *outtoken flow of v (w.r.t. x).*

Definition 7 (Token flow LPO). *A* token flow LPO *is a pair* $(\mathrm{lpo}, \mathbf{x})$*, where* $\mathrm{lpo} = (V, <, l)$ *is an LPO and* $\mathbf{x} = (x_1, \ldots, x_k) :<\rightarrow LS^k$ *is a function satisfying*
• lpo *has a unique minimal node* v_{min} *(*initial node*) with* $l(v_{min}) \notin l(V \setminus \{v_{min}\})$*.*
• lpo *has a unique maximal node* v_{max} *(*final node*) with* $l(v_{max}) \notin l(V \setminus \{v_{max}\})$*.*
• *Each* x_i *is a token flow function on* lpo *satisfying* $l(v) = l(w) \implies In_{x_i}(v) = In_{x_i}(w) \wedge Out_{x_i}(v) = Out_{x_i}(w)$ *for each pair of nodes* $v, w \in V$*.*

The LPO underlying a token flow LPO $(\mathrm{lpo}, \mathbf{x})$ *is the LPO* $\mathrm{lpo}' = (V', < |_{V' \times V'}, l|_{V'})$ *for* $V' = V \setminus \{v_{min}, v_{max}\}$*.*

A token flow LPO fulfills the token flow property if, roughly speaking, each token flow function corresponds to a place of the net. A token flow assigned to an arc (v, v') represents the local state which is produced by the occurrence of $l(v)$ and consumed by the occurrence of $l(v')$. The initial node is intended to produce the initial marking of the net, while the final node represents a transition consuming the final marking reached after the occurrence of all previous events given by the LPO (initial and final node are omitted in the LPO underlying a token flow LPO).

Definition 8 (Token flow property). *A token flow LPO* $(\mathrm{lpo}, \mathbf{x})$*,* $\mathrm{lpo} = (V, <, l)$ *and* $\mathbf{x} = (x_1, \ldots, x_k) :<\rightarrow LS^k$*, fulfills the* token flow property *w.r.t. a marked net* $N = (P, T, W, m_0)$ *of a flow type* (τ, f)*, if* $l(V \setminus \{v_{min}, v_{max}\}) \subseteq T$ *and if there is a bijective mapping* $\phi : \{1, \ldots, k\} \rightarrow P$ *satisfying*
• $\forall i :\ Out_{x_i}(v_{min}) = m_0(\phi(i))$*.*
• $\forall i, \forall v \neq v_{min}, v_{max} : Out_{x_i}(v) = f_2(W(\phi(i), l(v))) \wedge In_{x_i}(v) = f_1(W(\phi(i), l(v)))$*.*

For $\phi(i) = p$ *we also denote* $x_p = x_i$*,* $Out_p() = Out_{x_i}()$ *and* $In_p() = In_{x_i}()$*. We say that an LPO* lpo' *fulfills the token flow property if there is a token flow LPO* $(\mathrm{lpo}, \mathbf{x})$ *fulfilling the token flow property, such that* lpo' *is the LPO underlying* $(\mathrm{lpo}, \mathbf{x})$*.*

This notion of token flow property is a generalization of the token flow property for p/t-nets [10]. The only restriction in the flow type of nets definition not directly motivated as necessary by the previous definitions of token flow is the consideration of the set of local states as a free abelian monoid instead of an arbitrary abelian monoid. But there are examples showing that the characteristic of a free abelian monoid to have a subset of generator elements is necessary to guarantee that for each enabled LPO it is possible to find an appropriate token flow distribution ensuring the token flow property (see [11], Example 8).

Now we can show the central theorem of the paper that each LPO enabled w.r.t. a net of a flow type fulfills the token flow property. This result generalizes the result from [12] that each enabled LPO of a p/t-net sequentializes an LPO underlying a process net.

Theorem 1. *Let* $N = (P, T, W, m_0)$ *be a marked net of flow type* (τ, f)*,* $\tau = (LS, LE, \tau)$ *and let the LPO* $\mathrm{lpo}' = (V', <, l)$ *with* $l : V' \rightarrow T$ *be enabled w.r.t.* N*. Then* lpo' *fulfills the token flow property w.r.t.* N*.*

Next we consider the reverse statement to this theorem in order to actually get an equivalent characterization of enabledness by token flows. By considering several net classes, we have seen that for this we need different additional requirements depending on the net class. The simplest situation is to consider flow types of nets also satisfying the reverse implication of the third requirement in Definition 6, comprising e.g. p/t-nets:

Lemma 11. *Let (τ, f), $\tau = (LS, LE, \tau)$, be a flow type satisfying: $f_1(e) \leq s \wedge s' = s - f_1(e) + f_2(e) \Longrightarrow (s, e, s') \in \tau$ (note that this implies the ISP). Let $N = (P, T, W, m_0)$ be a marked net of flow type (τ, f) and let the LPO $\mathrm{lpo}' = (V', <, l)$ with $l : V' \to T$ fulfill the token flow property w.r.t. N. Then lpo' is enabled w.r.t. N.*

The requirement in Lemma 11 is quite restrictive, since e.g. inhibitor nets do not satisfy this restriction. One possibility to cover such net classes could be to equip each flow type (τ, f), $\tau = (LS, LE, \tau)$ with a *blocking function* $b : LS \times LE \to \{0, 1\}$ such that

$$(f_1(e) \leq s \wedge s' = s - f_1(e) + f_2(e) \wedge b(s, e) = 1) \Longleftrightarrow (s, e, s') \in \tau.$$

Canonical blocking functions for net classes such as pti-nets can easily be defined (see [11], Example 9). In this setting the enabledness of an LPO is determined by the flow and the blocking function. By Lemma 11 and Theorem 1, the flow part is encoded in the token flow property. For the blocking part an additional *non-blocking property* has to be considered such that an LPO is enabled if and only if the token flow property and the non-blocking property is satisfied. The non-blocking property is straightforwardly defined by requiring for every non-empty co-set (resp. cut if the ISP is fulfilled) C of an LPO lpo and every prefix lpo' of C that there is an enabled step sequence of lpo' with follower marking m such that $b(m(p), \sum_{t \in T} |C|_l(t) \cdot W(p, t)) = 1$ for each place p.

Corollary 1. *Let (τ, f), $\tau = (LS, LE, \tau)$, be a flow type equipped with a blocking function b. Let $N = (P, T, W, m_0)$ be a marked net of flow type (τ, f) and let $\mathrm{lpo} = (V, <, l)$ be an LPO with $l : V \to T$, then lpo is enabled if and only if the token flow property and the non-blocking property is satisfied.*

The non-blocking property can often be simplified by using specific features of the considered type of nets. We developed a simplified non-blocking property e.g. for inhibitor nets [14] and nets with capacities. Together with the token flow property these properties lead to efficient verification methods and are useful for synthesis.

5 Conclusion

In this paper we introduced several restrictions to types of nets in order to be able to introduce enabledness of LPOs w.r.t. nets of some type in a meaningful way. We first adapted the classical characterizations of enabled LPOs w.r.t. p/t-nets. The restrictions WISP, PIP and ISP introduced in this context seem to be natural for concrete Petri nets and are satisfied by all classes of net types, whose causal behaviour can reasonably be described through LPOs (including e.g. read arcs, inhibitor arcs and capacities w.r.t. most known semantics). There are some exceptions for net classes whose causal behaviour cannot any more be captured by LPOs, e.g. inhibitor nets w.r.t. the a-priori semantics and (usually) nets with step firing policies such as localities do not satisfy the ISP. Secondly, we generalized the newly developed characterization of enabled LPOs based on token flows (called token flow property) to net types. For this we considered so called flow types of nets, which still include the above mentioned net classes. For such types, the token flow property is a necessary condition for enabledness. In order to get equivalence of enabledness and token flow property, flow types of nets need to be further restricted. At this point it turns out that it is not clear how to define such a restriction in a general way without being too restrictive in the sense that p/t-nets are almost re-invented. The non-blocking property is a first proposal here.

We have already mentioned the main follow-up question for future research in the introduction: We want to analyze several notions of regions of types of nets for languages of LPOs and examine computation principles for concrete types of nets, with the aim of an integration into a general synthesis framework for types of nets.

Another interesting application of the central notion of token flows is the development of efficient verification algorithms to test, whether a given LPO is enabled in a given net of some flow type. For this purpose we plan to apply flow theory (generalized to free abelian monoids having a finite set of generators) as in the case of p/t-nets [10].

Note finally that the theory presented in this paper can also be formulated for infinite sets of transitions and places

References

1. E. Badouel, M. A. Bednarczyk, and P. Darondeau. Generalized Automata and Their Net Representations. In *Unifying Petri Nets, LNCS 2128*, pages 304–345, 2001.
2. E. Badouel and P. Darondeau. Dualities Between Nets and Automata Induced by Schizophrenic Objects. In *CTCS, LNCS 953*, pages 24–43, 1995.
3. E. Badouel and P. Darondeau. On the Synthesis of General Petri Nets. Technical Report 3025, Inria, 1996.
4. E. Badouel and P. Darondeau. Theory of Regions. In *Petri Nets, LNCS 1491*, pages 529–586, 1996.
5. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri Nets from Finite Partial Languages. *To appear in Fundam. Inform.*, 2008.
6. R. Bergenthum and S. Mauser. Comparison of Different Algorithms to Synthesize a Petri Net from a Partial Language. In *Proceedings of Workshop CHINA @ICATPN*, 2008.
7. J. Desel, G. Juhás, and R. Lorenz. Petri Nets over Partial Algebra. In *Unifying Petri Nets, LNCS 2128*, pages 126–172, 2001.
8. M. Droste and R. M. Shortt. From Petri Nets to Automata with Concurrency. *Applied Categorical Structures*, 10(2):173–191, 2002.
9. A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-Structures. Part I: Basic Notions and the Representation Problem / Part II: State Spaces of Concurrent Systems. *Acta Inf.*, 27(4):315–368, 1989.
10. G. Juhás, R. Lorenz, and J. Desel. Can I Execute My Scenario in Your Net? In *ICATPN 2005, LNCS 3536*, pages 289–308, 2005.
11. G. Juhás, R. Lorenz, and S. Mauser. Examples and Proofs: Partial Order Semantics of Types of Nets. Technical report, http://www.ku-eichstaett.de/Fakultaeten/MGF/Informatik/Mitarbeiter/Mauser/Publikationen.de, 2008.
12. A. Kiehn. On the Interrelation Between Synchronized and Non-Synchronized Behaviour of Petri Nets. *Elektronische Informationsverarbeitung und Kybernetik*, 24(1/2):3–18, 1988.
13. R. Lorenz, G. Juhás, and S. Mauser. How to Synthesize Nets from Languages - a Survey. In *Proceedings of the Wintersimulation Conference (WSC)*, pages 637–647, 2007.
14. R. Lorenz, S. Mauser, and R. Bergenthum. Theory of Regions for the Synthesis of Inhibitor Nets from Scenarios. In *ICATPN, LNCS 4546*, pages 342–361, 2007.
15. A. Mazurkiewicz. Petri Nets Without Tokens. In *ICATPN, LNCS 4546*, pages 20–23, 2007.