Complete Process Semantics for Inhibitor Nets

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

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

Abstract. In this paper we complete the semantical framework proposed in [13] for process and causality semantics of Petri nets by an additional aim and develop process and causality semantics of place/transition Petri nets with weighted inhibitor arcs (pti-nets) satisfying the semantical framework including this aim. The aim was firstly mentioned in [8] and states that causality semantics deduced from process nets should be *complete* w.r.t. step semantics in the sense that *each* causality structure which is consistent with the step semantics corresponds to some process net. We formulate this aim in terms of *enabled* causality structures.

While it is well known that process semantics of place/transition Petri nets (p/t-nets) satisfy the additional aim, we show that the most general process semantics of pti-nets proposed so far [13] does not and develop our process semantics as an appropriate generalization.

1 Introduction

The study of concurrency as a phenomenon of system behavior attracted much attention in recent years. There is an increasing number of distributed systems, multiprocessor systems and communication networks, which are concurrent in their nature. An important research field is the definition of non-sequential semantics of concurrent system models to describe concurrency among events in system executions, where events are considered concurrent if they can occur at the same time and in arbitrary order. Such non-sequential semantics is usually deduced from the so called step semantics of a concurrent system model.

For the definition of step semantics it is generally stated which events can occur in a certain state of the system *at the same time* (synchronously) and how the system state is changed by their occurrence. Such events form a *step (of events)*. Given an initial state, from this information all sequences of steps which can occur from the initial marking can easily be computed. The set of all possible such *step sequences* defines the step semantics of a concurrent system model. A step sequence can be interpreted as a possible *observation* of the systems behavior, where the event occurrences in one step are observed at the same time and the event occurrences in different steps are observed in the order given by the step sequence.

Non-sequential semantics are based on causal structures – we will also call them scenarios in the following – which allow to specify arbitrary concurrency relations among events. Non-sequential semantics for this paper is a set of scenarios. A scenario allows (generates) several different observations, since the occurrence of events which are concurrent in the scenario can be observed synchronously or also in arbitrary order. Therefore, a given scenario only represents behavior of the system if it is consistent with the step semantics in the sense that all of its generated observations belong to the step semantics of the system. Non-sequential semantics which consists only of scenarios satisfying this property we call *sound w.r.t. step semantics*. On the other hand, *all* scenarios which are consistent with the step semantics represent behavior of the system. Non-sequential semantics which contains *all* such scenarios we call *complete w.r.t. the step semantics*. In other words, a complete non-sequential semantics includes each causal structure satisfying that all observations generated by the causal structure are possible observations of the system. Note here that if we add causality to a causal structure which is consistent with the step semantics the resulting causal structure is again consistent with the step semantics (since it generates less observations). Thus, a complete non-sequential semantics can be given by such causal structures consistent with the step semantics satisfying that removing causality from the causal structure results in a causal structure *not* consistent with the step semantics. Such causal structures express *minimal* causal dependencies among events. Altogether, complete non-sequential semantics represent minimal causalities.

Therefore, an important aim of each semantical framework for the definition of a non-sequential semantics of particular formalisms for concurrent systems is that a non-sequential semantics is defined *sound and complete w.r.t. the step semantics* of the formalism. In this paper we consider this aim for Petri nets. These are one of the most prominent formalisms for understanding the concurrency phenomenon on the theoretical as well as the conceptual level and for modeling of real concurrent systems in many application areas [7]. The most important and well-known concept of non-sequential semantics of Petri nets are process semantics based on occurrence nets [4,5]. From the very beginning of Petri net theory processes were based on partial orders relating events labeled by transitions (an event represents the occurrence of a transition): Any process directly defines a respective partial order among events, called the associated *run*, in which unordered events are considered to be concurrent. Since adding causality to a run still leads to possible system behavior, a non-sequential semantics of a Petri net can also be given as the set of sequentializations of runs (a sequentialization adds causality) of the net. This set is also called causal semantics of the net, since it describes its causal behavior. Note that in most cases partial orders are suitable to describe such behavior but sometimes generalizations of partial orders are needed as appropriate causal structures. In the case of inhibitor nets under the so-called a-priori semantics [6], so called stratified order structures (so-structures) represent the causal semantics.

Since the basic developments of Petri nets, more and more different *Petri net classes* for various applications have been proposed. It turned out to be not easy to define process semantics and related causality semantics in the form of runs for such net classes. Therefore, in [13] (in the context of defining respective semantics for inhibitor nets) a semantical framework aiming at a systematic presentation of process and causality semantics of different Petri net models was developed (see Figure 3 in Section 3): Any process semantics should fulfill the reasonable aims stated by the framework.

These aims are reduced to several properties that have to be checked in a particular practical setting. The most important of these aims is the soundness of process semantics and causality semantics w.r.t. step semantics as described above. For Petri nets, soundness means that each observation generated by a process or a run is a possible step occurrence sequence of the Petri net. But this general framework – as well as many other particular process definitions for special Petri net classes – does not regard the described aim of completeness. In the Petri net context, process and causality semantics are complete w.r.t. step semantics if each causality structure consistent with the step semantics adds causality to or is equal to some run of the Petri net. Instead another aim of the framework from [13] requires a kind of weak completeness, saying that each step occurrence sequence should be generated by some process.

For place/transition nets (p/t-nets) a labeled partial order (LPO) which is consistent with the step semantics is called *enabled* [17,18,8]. It was shown in [11] that an LPO is enabled if and only if it is a sequentialization of a run corresponding to a process (see also [17,18,8]). Thus, process and causality semantics of p/t-nets are sound and complete w.r.t. step semantics. In particular, from the completeness we deduce that enabled LPOs with minimal causal dependencies between events (thus maximal concurrency) – so called *minimal enabled LPOs* – are generated by processes.[1] This is an essential property of p/t-net processes and justifies their success as non-sequential semantics describing system behavior.

Therefore, the aim of completeness should also hold for process semantics of other Petri net classes. To this end, we included it in the semantical framework of [13]. We will discuss the aim of completeness for process definitions of inhibitor nets. As stated in [15], ”Petri nets with inhibitor arcs are intuitively the most direct approach to increasing the modeling power of Petri nets”. Moreover inhibitor nets have been found appropriate in various application areas [1,3]. Accordingly, for these net classes various authors proposed process definitions regarding different interpretations of the occurrence rule of inhibitor nets. In this paper we will focus on the most general class of pti-nets and its process definition from [13].[2] We show that the general a-priori process definition of [13] does not fulfill the aim of completeness and propose appropriate changes of the process semantics. Thus we develop an alternative process definition which fulfills the complete semantical framework of Figure 3 including the aim of completeness.

As mentioned in the context of the a-priori semantics, LPOs are not expressive enough to describe the causal behavior of a pti-net. Instead, so-structures are used on the causal level. Thus the aim of completeness can be formulated for this net class in the following way: For any enabled so-structure there is a process with associated run in the form of an so-structure such that the enabled so-structure sequentializes the run. As in the case of LPOs, an so-structure is enabled if it is consistent with the step semantics of pti-nets in the above described sense.

The paper is structured as follows: First the basic notions of pti-nets, processes of pti-nets, so-structures (see [13]) and enabled so-structures are introduced (section 2). Then in section 3 the semantical framework of [13] will be discussed in the context of

[1] In case of p/t-nets and their processes (runs), not each enabled LPO is a run and there are also non-minimal runs, but each minimal enabled LPO is a minimal run.

[2] We will briefly consider alternative process definitions for inhibitor nets in the conclusion.

introducing a new requirement – the aim of completeness. Subsequently in the main part of the paper (section 4) we will show why the a-priori process semantics for pti-nets in [13] does not fulfill the aim of completeness. Based on these considerations we propose an alternative process semantics implementing the complete semantical framework including the aim of completeness.

2 Preliminaries

In this section we recall the basic definitions of *so-structures*, *pti-nets (equipped with the a-priori semantics)* and *process nets of pti-nets*, and finally define *enabled so-structures*.

Given a set X we will denote the set of all subsets of X by 2^X and the set of all multi-sets over X by $\mathbb{N}^X$. A set can always be viewed as a multi-set m with $m \leq 1$ and correspondingly a multi-set $m \leq 1$ can always be viewed as a set. We further denote the identity relation over X by id_X, the reflexive, transitive closure of a binary relation R over X by R^*, the transitive closure of R by R^+ and the composition of two binary relations R, R' over X by $R \circ R'$.

Inhibitor nets are an extension of classical Petri nets enhanced with inhibitor arcs. In their simplest version inhibitor arcs test whether a place is empty in the current marking (zero-testing) as an enabling condition for transitions. In the most general version of pti-nets, inhibitor arcs test if a place contains *at most* a certain number of tokens given by weights of the inhibitor arcs (instead of zero-testing). In pictures inhibitor arcs are depicted by arcs with circles as arrowheads. Figure 1 shows a pti-net, where the transitions t and v test a place to be empty and transition w tests a place to hold at most one token. As explained in [6,12,13], "earlier than" causality expressed by LPOs is not enough to describe causal semantics of pti-nets w.r.t. the a-priori semantics. In Figure 1 this phenomenon is depicted: In the a-priori semantics the testing for absence of tokens (through inhibitor arcs) precedes the execution of a transition. Thus t cannot occur later than u, because after the occurrence of u the place connected with t by an inhibitor arc (with weight 0 representing zero-testing) is marked. Consequently the occurrence of t is prohibited by this inhibitor arc. Therefore t and u cannot occur concurrently or sequentially in order $u \rightarrow t$. But they still can occur synchronously or sequentially in order $t \rightarrow u$, because of the occurrence rule "testing before execution" (details on the occurrence rule can be found later on in this section). This is exactly the behavior described by "t not later than u". After firing t and u we reach the marking in which every non-bottom and non-top place of the net NI contains one token. With the same arguments as above the transitions v and w can occur in this marking synchronously but not sequentially in any order. The relationship between v and w can consequently be expressed by a symmetric "not later than" relation between the respective events - none may occur later than the other. The described causal behavior of NI is illustrated through the run $\kappa(\mathrm{AON})$ on the right side of Figure 1. The solid arcs represent a (common) "earlier than" relation. Those events can only occur in the expressed order but not synchronously or inversely. Dashed arcs depict the "not later than" relation explained above. Partial orders can only model the "earlier than" relation, but it is not possible to describe relationships as in the example between t and u as well as between v and w, where synchronous occurrence is possible but concurrency is not existent.

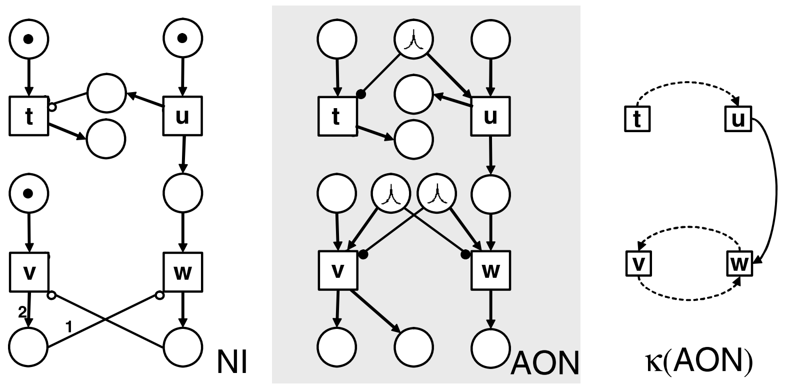

Fig. 1. A pti-net NI (inhibitor arcs have circles as arrowheads), an a-process AON of NI and the associated run $\kappa(\mathrm{AON})$

Altogether there exist net classes including inhibitor nets where synchronous and concurrent behavior has to be distinguished.[3] In [6] causal semantics based on so-structures (like the run $\kappa(\mathrm{AON})$) consisting of a combination of an "earlier than" and a "not later than" relation between events were proposed to cover such cases.

Before giving the definition of *stratified order structures* (*so-structures*), we recall the notion of a *directed graph*. This 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*. 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. $\rightarrow$ 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 *(strict) 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 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 satisfying certain properties. A *relational structure* (*rel-structure*) is a triple $\mathcal{S} = (V, \prec, \sqsubset)$, where V is a 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 1 (Stratified order structure). *A rel-structure $\mathcal{S} = (V, \prec, \sqsubset)$ is called* stratified order structure *(*so-structure*) if the following conditions are satisfied for all $u, v, w \in V$:*

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

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

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

[3] Further examples of such net classes are briefly mentioned in the conclusion.

can be deduced by (C3) and (C4). It is shown in [6] that $(V, \prec)$ is a partial order (thus a partial order can always be interpreted as an so-structure with $\sqsubset\, = \,\prec$). Therefore, so-structures are a generalization of partial orders. They turned out to be adequate to model the causal relations between events of complex systems regarding sequential, concurrent and synchronous behavior. In this context $\prec$ represents the ordinary "earlier than" relation (as in partial order based systems) while $\sqsubset$ models a "not later than" relation (recall the example of Figure 1).

Similar to the notion of the transitive closure of a binary relation the *$\diamondsuit$-closure* $\mathcal{S}^{\diamondsuit}$ of a rel-structure $\mathcal{S} = (V, \prec, \sqsubset)$ is defined by $\mathcal{S}^{\diamondsuit} = (V, \prec_{\mathcal{S}^{\diamondsuit}}, \sqsubset_{\mathcal{S}^{\diamondsuit}}) = (V, (\prec \cup \sqsubset)^* \circ \prec \circ (\prec \cup \sqsubset)^*, (\prec \cup \sqsubset)^* \setminus id_V)$. A rel-structure $\mathcal{S}$ is called *$\diamondsuit$-acyclic* if $\prec_{\mathcal{S}^{\diamondsuit}}$ is irreflexive. The *$\diamondsuit$-closure* $\mathcal{S}^{\diamondsuit}$ of a rel-structure $\mathcal{S}$ is an so-structure if and only if $\mathcal{S}$ is $\diamondsuit$-acyclic (for this and further results on the $\diamondsuit$-closure see [6]).

For our purposes we will only consider *labeled so-structures* (*LSOs*). Nodes of an LSO represent transition occurrences of a Petri net (constituted by node labels as in Figure 1). Formally LSOs are so-structures $\mathcal{S} = (V, \prec, \sqsubset)$ together with a *set of labels* T and a *labeling function* $l : V \to T$. A labeling 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 use the notations defined for so-structures also for LSOs.

We introduce an important subclass of so-structures similar to the subclass of total orders in the case of partial orders.

Definition 2 (Total linear so-structure). *An so-structure $\mathcal{S} = (V, \prec, \sqsubset)$ 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 1. Therefore the linearizations $lin(\mathcal{S}')$ of an so-structure $\mathcal{S}'$ are its maximal extensions. Note that a total linear so-structure $lin = (V, \prec, \sqsubset)$ represents a sequence of (synchronous) steps $\tau_1 \dots \tau_n$ (we also write $lin = \tau_1 \dots \tau_n$). A (synchronous) step is a set of cyclic $\sqsubset$-ordered events (forming a so called $\sqsubset$-clique – such events can only occur synchronously as explained in the context of Figure 1) and the sequential ordering is caused by $\prec$-relations between these steps. That means $\tau_1 \dots \tau_n$ and $(V, \prec, \sqsubset)$ are related through $V = \bigcup_{i=1}^{n} \tau_i$, $\prec\, = \bigcup_{i<j} \tau_i \times \tau_j$ and $\sqsubset\, = ((\bigcup_{i=1}^{n} \tau_i \times \tau_i) \setminus id_V) \cup \prec$. For example, the linearizations of the run $\kappa(AON)$ in Figure 1 are the sequences of (synchronous) steps $tu\{v, w\}$ and $\{t, u\}\{v, w\}$. By abstracting from the nodes of a total linear LSO $lin = (V, \prec, \sqsubset, l)$ representing $\tau_1 \dots \tau_n$, every step (set) of events τ_i can be interpreted as a step (multi-set) $l(\tau_i)$ of transitions using the labeling function. This is a general principle. That means we will interpret such a (synchronous) step sequence $\tau_1 \dots \tau$ of events based on a total linear LSO $lin = (V, \prec, \sqsubset, l)$ as a sequence $\sigma_{lin} = l(\tau_1) \dots l(\tau_n)$ of (synchronous) transition steps in a Petri net. Thus, we often do not distinguish total linear LSOs and respective sequences of transition steps in a Petri net. Lastly we need the notion of prefixes of so-structures. These are defined by subsets of nodes which are downward closed w.r.t. the $\sqsubset$-relation:

Definition 3 (Prefix). *Let $\mathcal{S} = (V, \prec, \sqsubset)$ be an so-structure and let $V' \subseteq V$ be a set of events such that $u' \in V'$, $u \sqsubset u' \Longrightarrow u \in V'$. Then V' is called* prefix *w.r.t. $\mathcal{S}$. A* prefix V' of $u \in V \setminus V'$ *is a prefix w.r.t. $\mathcal{S}$ satisfying $(v \prec u \Longrightarrow v \in V')$.*

The prefixes w.r.t. $\kappa(AON)$ in Figure 1 are the event sets $\{t\}$, $\{t, u\}$ and $\{t, u, v, e\}$. The only prefix of w is $\{t, u\}$, since v and w may not occur in a prefix of w ($w \sqsubset v$) and u has to occur in a prefix of w ($u \prec w$). We have the following relation between prefixes and linearizations of so-structures:

Lemma 1. *Let V' be a prefix (of $u \in V$) w.r.t. an so-structure $\mathcal{S} = (V, \prec, \sqsubset)$, then there exists $lin \in lin(\mathcal{S})$ such that V' is a prefix (of u) w.r.t. lin.*

Proof. $lin = \tau_1 \dots \tau_n$ can be constructed as follows: $\tau_1 = \{v \in V' \mid \forall v' \in V' : v' \not\prec v\}$, $\tau_2 = \{v \in V' \setminus \tau_1 \mid \forall v' \in V' \setminus \tau_1 : v' \not\prec v\}$ and so on, i.e. we define $\tau_i \subseteq V'$ as the set of nodes $\{v \in V' \setminus (\bigcup_{j=1}^{i-1} \tau_j) \mid \forall v' \in V' \setminus (\bigcup_{j=1}^{i-1} \tau_j) : v' \not\prec v\}$ which are minimal w.r.t. the restriction of $\prec$ onto the node set $V' \setminus (\bigcup_{j=1}^{i-1} \tau_j)$, as long as $V' \setminus (\bigcup_{j=1}^{i-1} \tau_j) \neq \emptyset$. Then continue with the same procedure on $V \setminus V' = V \setminus (\bigcup_{j=1}^{i} \tau_j)$, i.e. $\tau_{i+1} = \{v \in V \setminus (\bigcup_{j=1}^{i} \tau_j) \mid \forall v' \in V \setminus (\bigcup_{j=1}^{i} \tau_j) : v' \not\prec v\}$ and so on. By construction V' is a prefix (of u) w.r.t. lin. A straightforward computation also yields $lin \in lin(\mathcal{S})$. $\square$

A prefix V' w.r.t. a total linear so-structure $lin = \tau_1 \dots \tau_n$ always represents a primary part of the respective (synchronous) step sequence, i.e. $V' = \bigcup_{j \leq i} \tau_j$ for some $i \in \{0, \dots, n\}$. If V' is a prefix of u, then $u \in \tau_{i+1}$.

Next we present the net class of pti-nets (p/t-nets with weighted inhibitor arcs). As usual, a *p/t-net* is a triple $N = (P, T, W)$, where P is a finite set of places, T is a finite set of transitions and $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the weight function representing the flow relation. The pre- and post-multi-set of a transition $t \in T$ are the multi-sets of places given by ${}^\bullet t(p) = W(p, t)$ and $t^\bullet(p) = W(t, p)$ for all $p \in P$. This notation can be extended to $U \in \mathbb{N}^T$ by ${}^\bullet U(p) = \sum_{t \in U} U(t)\, \dot{{}^\bullet} t(p)$ and $U^\bullet(p) = \sum_{t \in U} U(t) \dot{t}^\bullet(p)$ for all $p \in P$. Analogously we can define pre- and post-multi-sets of multi-sets of places as multi-sets of transitions. Each $m \in \mathbb{N}^P$ is called a *marking* of N and each $U \in \mathbb{N}^T$ is called a step of N. U is *enabled to occur* in m if and only if $m \geq {}^\bullet U$. In this case, its occurrence leads to the marking $m' = m - {}^\bullet U + U^\bullet$.

Definition 4 (Pti-net). *A marked* pti-net *is a quadruple $NI = (P, T, W, I, m_0)$, where* $\mathrm{Und}(NI) = (P, T, W)$ *is a p/t-net (the* underlying net *of NI), m_0 the* initial marking *of NI and $I : P \times T \to \mathbb{N} \cup \{\infty\}$ is the* inhibitor (weight) function *(we assume $\infty > n$ for every $n \in \mathbb{N}$). For a transition t the negative context ${}^-t \in (\mathbb{N} \cup \{\infty\})^P$ is given by ${}^-t(p) = I(p, t)$ for all $p \in P$. For a step of transitions U, ${}^-U \in (\mathbb{N} \cup \{\infty\})^P$ is given by ${}^-U(p) = min(\{\infty\} \cup \{{}^-t(p) \mid t \in U\})$. A place p with ${}^-t(p) \neq \infty$ is called* inhibitor place *of t.*

A step of transitions U is (synchronously) enabled to occur *in a marking m if and only if it is enabled to occur in the underlying p/t-net* $\mathrm{Und}(NI)$ *and in addition $m \leq {}^-U$. The occurrence of U leads to the marking $m' = m - {}^\bullet U + U^\bullet$. This is denoted by $m \xrightarrow{U} m'$. A finite sequence of steps of transitions $\sigma = U_1 \dots U_n$, $n \in \mathbb{N}$, 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* $m \xrightarrow{U_1} m_1 \xrightarrow{U_2} \ldots \xrightarrow{U_n} m_n$. *By* $\mathcal{EX}(NI)$ *we denote the set of all step sequences of a marked pti-net* NI.

Note that $I(p,t) = k \in \mathbb{N}$ implies that t can only occur if p does not contain more than k tokens (as explained in the context of the inhibitor arc connected with w in Figure 1); $k = 0$ coincides with zero-testing. Accordingly $I(p,t) = \infty$ means that the occurrence of t is not restricted through the presence of tokens in p. Thus a p/t-net can always be interpreted as a pti-net with $I \equiv \infty$. In graphic illustrations, inhibitor arcs are drawn with circles as arrowheads and annotated with their weights (see Figure 1). Inhibitor arcs with weight ∞ are completely omitted and the inhibitor weight 0 is not shown in diagrams. The definition of enabledness in Definition 4 reflects the considerations about the a-priori testing explicated above: the inhibitor constraints are obeyed before the step of transitions is executed. For an example, see Figure 1 and the explanations at the beginning of this section.

Now we introduce the process semantics for pti-nets as presented in [13]. The problem is that the absence of tokens in a place – this is tested by inhibitor arcs – cannot be directly represented in an occurrence net. This is solved by introducing local extra conditions and read arcs – also called activator arcs – connected to these conditions. These extra conditions are introduced "on demand" to directly represent dependencies of events caused by the presence of an inhibitor arc in the net. The conditions are artificial conditions without a reference to inhibitor weights or places of the net. They only focus on the dependencies that result from inhibitor tests. Thus, activator arcs represent local information regarding the lack of tokens in a place. The process definition of [13] is based on the usual notion of occurrence nets extended by activator arcs. These are (labeled) acyclic nets with non-branching places (conditions) (since conflicts between transitions are resolved). By abstracting from the conditions one obtains an LSO representing the causal relationships between the events. In the following definition B represents the finite set of *conditions*, E the finite set of *events*, R the flow relation and Act the set of activator arcs of the occurrence net.

Definition 5 (Activator occurrence net). *A* labeled activator occurrence net *(ao-net) is a five-tuple* $\mathrm{AON} = (B, E, R, Act, l)$ *satisfying:*

- B *and* E *are finite disjoint sets,*
- $R \subseteq (B \times E) \cup (E \times B)$ *and* $Act \subseteq B \times E$,
- $|{}^\bullet b|, |b^\bullet| \leq 1$ *for every* $b \in B$,
- *the relational structure* $\mathcal{S}(\mathrm{AON}) = (E, \prec_{loc}, \sqsubset_{loc}, l|_E) = (E, (R \circ R)|_{E \times E} \cup (R \circ Act), (Act^{-1} \circ R) \setminus id_E, l|_E)$ *is* $\diamondsuit$*-acyclic,*
- l *is a labeling for* $B \cup E$.

The LSO generated by AON *is* $\kappa(\mathrm{AON}) = (E, \prec_{\mathrm{AON}}, \sqsubset_{\mathrm{AON}}, l|_E) = \mathcal{S}(\mathrm{AON})^{\diamondsuit}$.

The relations $\prec_{loc}$ and $\sqsubset_{loc}$ represent the local information about causal relationships between events. Figure 2 shows their construction rule. $\kappa(\mathrm{AON})$ captures all (not only local) causal relations between the events (see also Figure 1). Note that Definition 5 is a conservative extension of common occurrence nets by read arcs.

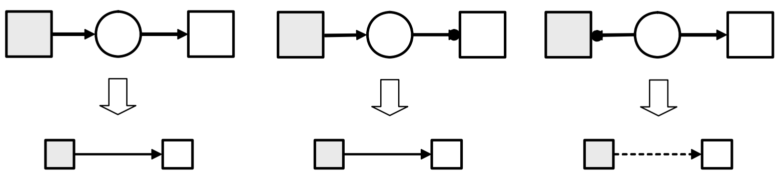

Fig. 2. Generation of the orders $\prec_{loc}$ and $\sqsubset_{loc}$ in ao-nets

The initial marking $\mathrm{MIN_{AON}}$ of AON consists of all conditions without incoming flow arcs (the minimal conditions w.r.t. R). The final marking $\mathrm{MAX_{AON}}$ of AON consists of all conditions without outgoing flow arcs (the maximal conditions w.r.t. R). There are two different notions of configurations and slices for ao-nets. A set of events $D \subseteq E$ is a *strong configuration* of AON, if $e \in D$ and $f \prec_{loc}^{+} e$ implies $f \in D$. D is called a *weak configuration* of AON, if $e \in D$ and $f(\prec_{loc} \cup \sqsubset_{loc})^{+}e$ implies $f \in D$. A *strong slice* of AON is a maximal (w.r.t. set inclusion) set of conditions $S \subseteq B$ which are incomparable w.r.t. the relation $R \circ \prec_{loc}^{*} \circ R$, denoted by $S \in \mathrm{SSL(AON)}$. A *weak slice* of AON is a maximal (w.r.t. set inclusion) set of conditions $S \subseteq B$ which are incomparable w.r.t. the relation $R \circ (\prec_{loc} \cup \sqsubset_{loc})^{*} \circ R$, denoted by $S \in \mathrm{WSL(AON)}$. In the example occurrence net from Figure 1 $|\mathrm{WSL}| = 4$ and $|\mathrm{SSL}| = 12$.

Every weak configuration is also a strong configuration and every weak slice is also a strong slice. In [13] it is shown that the set of strong slices of AON equals the set of all sets of conditions which are generated by firing the events of a strong configuration. An analogous result holds for weak slices and weak configurations. $\mathrm{SSL(AON)}$ equals the set of all sets of conditions reachable from the initial marking $\mathrm{MIN_{AON}}$ in AON and $\mathrm{WSL(AON)}$ equals the set of all sets of conditions from which the final marking $\mathrm{MAX_{AON}}$ is reachable in AON (using the standard a-priori occurrence rule of elementary nets with read arcs [13]). By $\mathrm{MAR}(C)$ we denote the marking resulting from the initial marking of a net by firing the multi-set of transitions corresponding to a (weak or strong) configuration C.

Now we are prepared to define processes of pti-nets as in [13]. The mentioned artificial conditions are labeled by the special symbol $\curlywedge$. They are introduced in situations, when a transition $t \in T$ tests a place in the pre- or post-multi-set of another transition $w \in T$ for absence of tokens, i.e. when $I(p,t) \neq \infty$ and ${}^{\bullet}w(p) + w^{\bullet}(p) \neq 0$ for some $p \in P$. Such situations are abbreviated by $w \multimap t$. If $w \multimap t$ holds, then any two occurrences f of w and e of t are adjacent to a common $\curlywedge$-condition representing a causal dependency of f and e. That means there exists a condition $b \in \widetilde{B}$ such that $(b,e) \in Act$ and ${}^{\bullet}f(b) + f^{\bullet}(b) \neq 0$ (remember that ${}^{\bullet}f, f^{\bullet} \in B^{\mathbb{N}}$ are multi-sets over B) – abbreviated by $f \multimap\!\!\bullet\, e$ (see requirement 6. in Definition 6). Thus the axiomatic process definition in [13] is as follows:

Definition 6 (Activator process). *An* activator process *(a-process) of NI is an ao-net* $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ *satisfying:*

1. *$l(B) \subseteq P$ and $l(E) \subseteq T$.*
2. *The conditions in $\widetilde{B} = \{b \mid \exists e \in E : (b,e) \in Act\}$ are labelled by the special symbol $\curlywedge$.*

3. *$m_0 = l(\mathrm{MIN}_{\mathrm{AON}} \cap B)$.*
4. *For all $e \in E$, ${}^\bullet l(e) = l({}^\bullet e \cap B)$ and $l(e)^\bullet = l(e^\bullet \cap B)$.*
5. *For all $b \in \widetilde{B}$, there are unique $g, h \in E$ such that ${}^\bullet b + b^\bullet = \{g\}$, $(b, h) \in Act$ and $l(g) \multimap l(h)$.*
6. *For all $e, f \in E$, if $l(f) \multimap l(e)$ then there is exactly one $c \in \widetilde{B}$ such that $f \multimap\!\!\bullet\, e$ through c.*
7. *For all $e \in E$ and $S \in \mathrm{SSL}(\mathrm{AON})$, if ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b, e) \in Act\} \subseteq S$ then $l(S \cap B) \leq {}^- l(e)$.*

The set of a-processes of NI (given by this axiomatic definition) is denoted by $\alpha(NI)$. For $\mathrm{AON} \in \alpha(NI)$ the generated so-structure $\kappa(\mathrm{AON})$ is called a run (associated to AON).

The occurrence net AON in Figure 1 is indeed an a-process: All $\curlywedge$-labeled conditions satisfy 5. All $\curlywedge$-labeled conditions which are necessary according to 6. are drawn. Condition 7. must be simply verified for the strong slices produced by strong configurations, e.g. $\mathrm{MAR}(\emptyset)$, $\mathrm{MAR}(\{t\})$, $\mathrm{MAR}(\{u\})$, $\mathrm{MAR}(\{u, t\})$ and so on. Thus, $\kappa(\mathrm{AON})$ is a run.

The requirements 1., 3., 4. in Definition 6 represent common features of processes well-known from p/t-nets. They ensure that a-processes constitute a conservative generalization of common p/t-net processes. That means, the set of processes of $\mathrm{Und}(NI)$ coincides with the set of processes resulting from $\alpha(NI)$ by omitting the $\curlywedge$-labeled conditions (omitting the $\curlywedge$-conditions from an a-process AON leads to the so called underlying process UAON of AON). If NI has no inhibitor arcs (thus $NI = \mathrm{Und}(NI)$) a-processes coincide with common processes. Thus, Definition 6 can also be used to define processes of p/t-nets. The properties 2. and 5. together with the rule 6. – describing when $\curlywedge$-conditions have to be inserted – constitute the structure of the $\curlywedge$-conditions. The requirement 7. expresses that in the strong slices of AON the inhibitor constraints of the pti-net have to be properly reflected. That means, for events enabled in a certain slice of AON the respective transitions are also enabled in the respective marking in the pti-net NI.

We finally formally define, when we consider an LSO $\mathcal{S}$ to be consistent with the step semantics $\mathcal{EX}$ of a given pti-net (Definition 4). Such LSOs we call *enabled* (w.r.t. the given pti-net). Intuitively it is clear what enabledness means: The transitions associated to the events of an LSO can be executed in the net regarding all given concurrency and dependency relations. For the formal definition the concurrency and dependency relations described by $\mathcal{S}$ are reduced to the set of step sequences sequentializing $\mathcal{S}$ (given by $lin(\mathcal{S})$). Such step sequences can be considered as observations of $\mathcal{S}$, where transition occurrences within a step are observed at the same time (synchronously), and step occurrences are observed in the order given by the step sequence. If each such observation of $\mathcal{S}$ is an enabled step occurrence sequences of the pti-net, $\mathcal{S}$ is consistent with the step semantics.

Definition 7 (Enabled LSO). *An LSO $\mathcal{S} = (V, \prec, \sqsubset, l)$ is enabled w.r.t. a marked pti-net $NI = (P, T, W, I, m_0)$ if and only if every $lin \in lin(\mathcal{S})$ represents an enabled (synchronous) step sequence σ_{lin} in $\mathcal{EX}(NI)$ (of NI). $\mathcal{ELCS}(NI)$ is the set of all so-structures enabled w.r.t. a given marked pti-net NI.*

With this definition one can easily check that the run $\kappa(AON)$ in Figure 1 is enabled w.r.t. NI: The two linearizations of $\kappa(AON)$ represent the sequences of synchronous steps $tu\{v,w\}$ and $\{t,u\}\{v,w\}$ which are both executable in NI.

Definition 7 is consistent with and a proper generalization of the notion of enabled LPOs in the context of p/t-nets: An LPO $\mathrm{lpo} = (V, \prec, l)$ with $l : V \to T$ is *enabled w.r.t. a marked p/t-net* (P, T, W, m_0) if each step sequence which extends 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 labeled by transitions (transition occurrences) which are concurrent.

Beside the consistency of Definition 7 with the definition of enabled LPOs, there are two general semantical arguments justifying this definition: First the set of total linear LSOs $lin(\mathcal{S})$, which are tested for enabledness in the Petri net, represents $\mathcal{S}$. This is shown in [6] by the following generalization of Szpilrajns theorem [16] to so-structures: $\mathcal{S} = (V, \bigcap_{(V,\prec,\sqsubset)\in lin(\mathcal{S})} \prec, \bigcap_{(V,\prec,\sqsubset)\in lin(\mathcal{S})} \sqsubset)$. Second the set $lin(\mathcal{S})$ can express arbitrary concurrency relations between transition occurrences of a pti-net, since concurrency equals the possibility of sequential occurrence in any order and synchronous occurrence. Thus, considering more generally sequences of concurrent steps of synchronous steps instead of simply sequences of synchronous steps does not lead to a higher expressivity of concurrency. These two arguments justify the choice of synchronous step sequences as the operational semantics (of executions) of pti-nets. Thus the definition of enabled LSOs based on synchronous step sequences and total linear LSOs constitutes the adequate causal semantics.

3 The Semantical Framework

In [13] a general framework for dealing with process semantics of Petri nets was proposed (see Figure 3, left part). It aims at a support for a systematic development of process and causality semantics for various Petri net classes using a common scheme.

In Figure 3 the abbreviations mean the following. $\mathcal{PN}$ represents a Petri net model together with an operational occurrence rule. $\mathcal{EX}$ are executions such as step sequences in accordance to the occurrence rule employed by $\mathcal{PN}$. $\mathcal{LAN}$ represents the process semantics given by labeled acyclic nets such as occurrence nets. $\mathcal{LEX}$ are labeled executions such as step sequences of nets in $\mathcal{LAN}$. Finally, $\mathcal{LCS}$ are labeled causal structures describing net behavior through causality relations between events. The arrows indicate functions that define and relate the different semantical views. They represent the consistency requirements for process semantics according to this framework. ω yields the set of executions (step sequences) providing the operational semantics (Definition 4 for pti-nets). α defines the axiomatic process definition (Definition 6). κ associates so called runs to the process definition (Definition 6); $\kappa(\mathcal{LAN}) \subseteq \mathcal{LCS}$ defines the set of runs of a net. λ represents the operational semantics of the process definition given by labeled step sequences (defined through a slight modification of the step occurrence rule of elementary nets with read arcs under the a-priori semantics [13]). Through ϕ a labeled execution can be interpreted as an ordinary execution (defined as trivial modification omitting labels). ϵ and ι relate a labeled causal structure with its generated

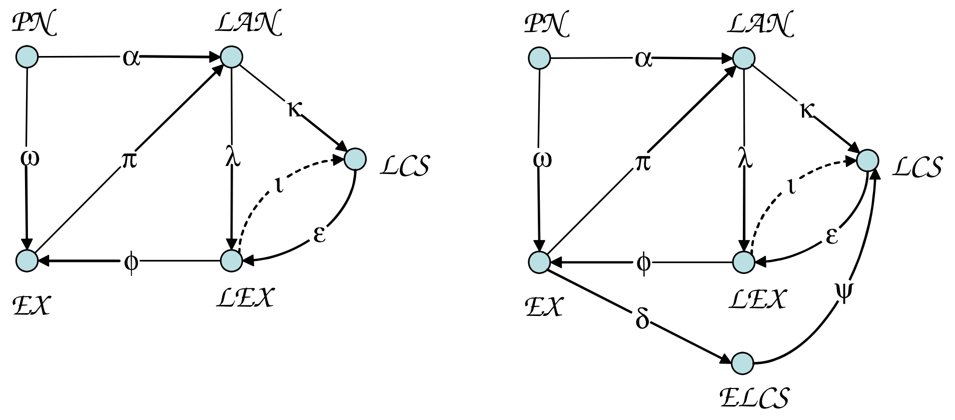

Fig. 3. Left: The semantical framework of [13]. Right: The left semantical framework extended by the completeness-requirement that any enabled causal structure has to be a sequentialization of a run; this is depicted through $\mathcal{ELCS}$ and the adjacent arcs labeled by δ and ψ

labeled executions (ϵ respectively ι are given as linearizations respectively intersections in the case of LSOs). Finally, π represents the operational process definition starting from executions.

This framework defines reasonable requirements for process semantics. It provides a schematic approach to ensure that process and causality semantics developed for a special Petri net class are consistently defined. In [13] the framework is condensed to five properties that have to be checked in each particular setting. Two of these properties state that all mappings in Figure 3 are total and all mappings returning sets do not return the empty set. *Consistency* is formulated there as the following separated properties:

Soundness: The process definition $\mathcal{LAN}$ should be *sound* w.r.t. the step semantics $\mathcal{EX}$ in the sense that every run should be consistent with the step semantics.
Weak completeness: $\mathcal{LAN}$ should be *weak complete* w.r.t. $\mathcal{EX}$ in the sense that $\mathcal{EX}$ should be reproducible from $\mathcal{LAN}$.
Construction of processes from step sequences: A process in $\mathcal{LAN}$ should be constructible from each step sequence in $\mathcal{EX}$ generated by the process (by π).
Consistency of runs and processes (called *Fitting* in [13])): Processes and corresponding runs should generate the same step sequences.
Runs are reconstructible from step sequences (called *Representation* in [13])): Runs from $\mathcal{LCS}$ should be reconstructible from step sequences in $\mathcal{EX}$ by $\iota \circ \epsilon$.

But an important feature of process semantics relating runs and step semantics is not present in this framework. On the one hand, $\phi \circ \epsilon$ ensures that each run is consistent with the step semantics (soundness). On the other hand, there is no requirement guaranteeing the converse, that each causal structure which is consistent with the step semantics is generated by a run through adding causality to it (completeness). For p/t-nets this is fulfilled (as mentioned in the Introduction), since every enabled LPO is a sequentialization of a run [11]. Together with the reverse statement that runs are enabled (soundness), completeness guarantees that there are runs and processes which express all valid causal behavior of the net regarding as much concurrency as possible. That means, the minimal causal dependencies in a net are reflected in the process semantics. To represent such an aim of completeness, we add new relations to the semantical framework (Figure 3, right part) by the introduction of enabled causal structures $\mathcal{ELCS}$. The arc labeled by δ represents the definition of enabled labeled causal structures $\mathcal{ELCS}$ from the operational semantics $\mathcal{EX}$. The arc labeled with ψ relates enabled labeled causal structures ($\mathcal{ELCS}$) and runs ($\kappa(\mathcal{LAN}) \subseteq \mathcal{LCS}$) in the above sense by assigning a run with less causality to each enabled labeled causal structure (for which such a run exists). Formally, a labeled causal structure is said to have *less causality* then a second one, if each labeled execution in $\mathcal{EX}$ generated by the second one is also generated by the first one (where the labeled executions generated by a labeled causal structure are given by ϵ). Thus, through $\psi \circ \delta$ we add an additional property to the process framework that we call the aim of completeness.

Definition 8 (Aim of completeness). *The mapping δ assigns a set of step sequences $\mathcal{EX}$ onto the set of causal structures $\mathcal{ELCS}$ enabled w.r.t. $\mathcal{EX}$. The mapping ψ assigns a run $\mathcal{LCS}$ with less causality to each enabled causal structure in $\mathcal{ELCS}$ for which such a run exists.*

The aim of completeness *states that the mapping ψ is total, i.e. that each enabled causal structure adds causality to some run.*

The absence of the aim of completeness in the framework of [13] leads to process definitions that do not have to represent minimal causal behavior. According to [13] a process definition that equals the operational step semantics (processes are step sequences) is a valid process semantics. But the set of step sequences is not a reasonable process semantics and process definitions not producing the minimal causalities are not really useful. The aim of completeness in our framework solves this problem. It implies that minimal enabled labeled causal structures coincide with (minimal) runs: On the one hand a minimal enabled labeled causal structure has to be a sequentializations of a run, on the other hand runs have to be enabled – so runs cannot have less causalities than minimal enabled labeled causal structures.

4 Process Semantics of Pti-nets

The definition of a-processes from section 2 meets all requirements of the left semantical framework in Figure 3 as shown in [13]. In the setting of pti-nets the additional aim of completeness states that each enabled so-structure extends some run of the pti-net. We show in this section that a-processes do not fulfill the aim of completeness. Moreover, we develop an alternative process definition preserving all the other requirements of the semantical framework, such that the aim of completeness is fulfilled.

The basic intuition behind the fact that the a-processes from Definition 6 do not generate minimal causalities is as follows: The definition uses constraints introduced through artificial $\curlywedge$-labeled conditions. They do not have counterparts on the pti-net level, but rather represent dynamic causal relationships between events. Therefore, it is possible that the definition of the $\curlywedge$-conditions does not reflect the causalities in the original pti-net such that too many constraints are introduced in the runs generated by

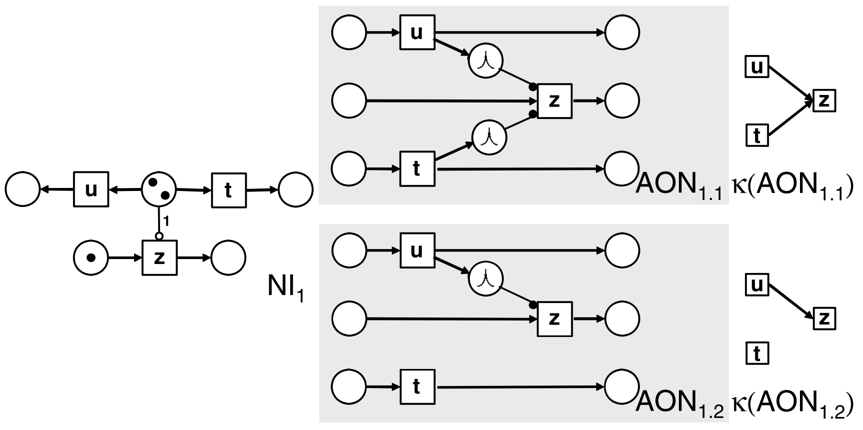

Fig. 4. A pti-net NI_1, an a-process $\mathrm{AON}_{1.1}$ of NI_1 and the associated run $\kappa(\mathrm{AON}_{1.1})$ together with an ao-net $\mathrm{AON}_{1.2}$ that is a candidate to be a process of NI_1, and the associated run $\kappa(\mathrm{AON}_{1.2})$. This example from [13] shows that a-processes (mandatory) introduce unnecessary causalities.

a-processes. In this section we will step by step illustrate via examples why the aim of completeness does not hold for a-processes and adapt their definition such that this aim is finally fulfilled (all the other requirements will be preserved).

In the following we give two examples of LSOs enabled w.r.t. a marked pti-net, which do not extend a run of the considered net. Each of these examples leads to a specific modification of Definition 6. We assume that events in these examples are labeled by the identity mapping, i.e. u, t and z are events representing the occurrence of the transitions $l(u) = u$, $l(t) = t$ and $l(z) = z$. The place connected to z by an inhibitor arc in each example we denote by p.

The first example gave the authors of [13] themselves. The a-process $\mathrm{AON}_{1.1}$ in Figure 4 shows that the technique of introducing $\curlywedge$-labeled conditions according to Definition 6 in general generates too many constraints in the associated run $\kappa(\mathrm{AON}_{1.1})$: ”One may easily verify that we can safely delete one of the activator arcs (but not both), which leads to another a-process generating weaker constraints than $AON_{1.1}$”. Indeed, deleting for example the $\curlywedge$-condition between t and z the resulting ao-net $\mathrm{AON}_{1.2}$ is a reasonable process. The other $\curlywedge$-condition orders u and z in sequence $u \longrightarrow z$ and t can occur concurrently to this sequence. On the other hand, omitting the $\curlywedge$-condition between t and z contradicts 6. of Definition 6 because there holds $t \multimap z$. That means $\mathrm{AON}_{1.2}$ is not an a-process (in particular the quoted statement is not exactly true). Thus, the LSO $\kappa(\mathrm{AON}_{1.2})$ is enabled but does not sequentialize a run (since it can only be generated by an ao-net without a $\curlywedge$-condition adjacent to t and z). An analogous observations holds symmetrically when deleting the $\curlywedge$-condition between u and z instead between t and z. Consequently, the first modification of Definition 6 is to replace requirement 6. by requirement 6.’. According to 6.’, the unique condition $c \in \widetilde{B}$ is only possible instead of required. Then the problem discussed above is solved and the ao-net $\mathrm{AON}_{1.2}$ is actually a process.

6.’ For all $e, f \in E$, if $f \multimap\!\!\bullet\, e$ then there is exactly one $c \in \widetilde{B}$ such that $f \multimap\!\!\bullet\, e$ through c.

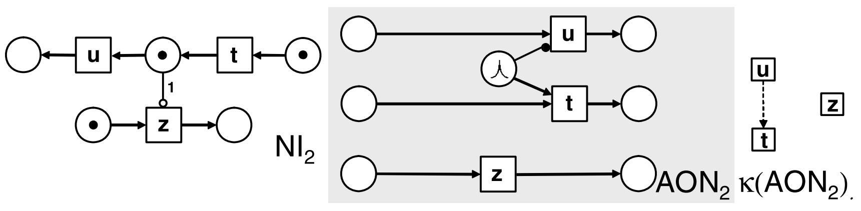

Fig. 5. A pti-net NI_2, an *ao*-net AON_2 that is a candidate to be a process of NI_2, and the associated run $\kappa(\mathrm{AON}_2)$. The *ao*-net models executable causalities that cannot be generated with a-processes.

The net NI_2 of Figure 5 shows that the aim of completeness is still not fulfilled: If u and t occur causally ordered in sequence $u \rightarrow t$ then z can fire concurrently to this sequence because the place p never contains more than one token. It is even possible to fire z concurrently to the synchronous step $\{u, t\}$. Consequently $\kappa(\mathrm{AON}_2)$, requiring solely that u occurs "not later than" t, is enabled (check Definition 7). The only possibility to introduce such a causal dependency between u and t on the process level is through a $\curlywedge$-condition between u and t. This is illustrated by the ao-net AON_2 (compare Figure 2). But according to 5. of Definition 6, AON_2 is not an a-process, since $l(u) \not\multimap l(t)$. Thus, a run which is extended by $\kappa(\mathrm{AON}_2)$ has no ordering between u, t and z. This is not possible because such a run is not enabled (the step sequence $t \rightarrow z \rightarrow u$ cannot be fired). That means $\kappa(\mathrm{AON}_2)$ does not sequentialize a run. Altogether, in 5. an important possibility of generating causal dependencies from inhibitor arcs via $\curlywedge$-conditions is not present. Allowing $\curlywedge$-conditions as in AON_2 solves this problem leading to a process having $\kappa(\mathrm{AON}_2)$ as its associated run. This $\curlywedge$-condition represents the causal dependency of u and t caused by the inhibitor arc (p, z). It reflects the inhibitor testing of z and not of u or t. A generalization of 5. allowing $\curlywedge$-conditions also in situations as in this example is a next necessary step towards the aim of completeness. Loosely speaking, we will allow to insert $\curlywedge$-conditions additionally in the following situation: If a transition, testing some place via an inhibitor arc, occurs concurrently to transitions consuming and producing tokens in this place, these transition occurrences must eventually be ordered via a $\curlywedge$-condition. This $\curlywedge$-conditions is intended to ensure that tokens are consumed not later than produced in order to restrict the maximal number of tokens in this place according to the inhibitor weight. To this end, we replaces 5. by the weaker requirement 5.'. It introduces a more general structural construction rule of $\curlywedge$-conditions using this intuition as follows:

5.' For all $b \in \widetilde{B}$, there are unique $g, h \in E$ such that ${}^\bullet b + b^\bullet = \{g\}$, $(b, h) \in Act$ and additionally $l(g) \multimap l(h)$ or ${}^\bullet l(h) \cap l(g)^\bullet \cap {}^- z \neq \emptyset$ for a $z \in T$.

But the modifications proposed so far still do not ensure that AON_2 is a process, since AON_2 does not fulfill 7. of Definition 6: The conditions resulting from only firing t in the initial marking establish a strong slice S and z fulfills ${}^\bullet z \cup \{b \in \widetilde{B} \mid (b, z) \in Act\} \subseteq S$. That means that using the standard occurrence rule of elementary nets with read arcs under the a-priori semantics [13] S constitutes a reachable marking in the process net and z is enabled in this marking in the process net. But obviously in the pti-net z is not enabled in the marking resulting from firing t. This problem can be resolved as follows: In AON_2 the event t can fire in the initial marking, although the $\curlywedge$-condition generates the ordering "u not later than t". Thus, firing t in the initial marking disables u. This means that we could have omitted u from AON_2 which leads to a different *ao*-net. Consequently, it is a proper assumption that *ao*-nets should model only such behavior in which every event of the *ao*-net actually occurs. Under this assumption, firing t in the initial marking is not a valid behavior of the *ao*-net and therefore the problematic marking S is not a marking of interest. The markings of interest are the markings reachable from the minimal conditions ($\mathrm{MIN}_{\mathrm{AON}_2}$) in the *ao*-net from which we can reach the maximal conditions ($\mathrm{MAX}_{\mathrm{AON}_2}$). That means, all events of the *ao*-net not fired yet can still be executed starting in the respective marking. These markings are represented by the weak slices of the *ao*-net. Therefore, we replace 7. by 7.', where SSL (strong slices) are replaced by WSL (weak slices) reflecting the above assumption:

7.' For all $e \in E$ and $S \in \mathrm{WSL}(\mathrm{AON})$, if ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b,e) \in \mathit{Act}\} \subseteq S$ then $l(S \cap B) \leq {}^- l(e)$.

This is a generalization of Definition 6 since WSL $\subseteq$ SSL. From the intuitive point of view the two alternative formulations 7. and 7.' focus on different aspects: While the consideration of SSL completely reflects the occurrence rule of elementary nets with read arcs, the consideration of WSL additionally postulates that no event of the *ao*-net may completely be disabled. This second assumption is also used in [13] for defining the executions $\mathcal{LEX}$ through the mapping λ in the semantical framework of Figure 3: λ represents all step sequences of an a-process in $\mathcal{LAN}$ in which every event of the process occurs. In this sense the change of the occurrence rule of *ao*-nets explained above is an adaption to the idea of mandatory regarding all events used in the operational semantics of *ao*-nets anyway. Therefore, this slightly altered occurrence rule of *ao*-nets (that we will use) is completely consistent to the executions of *ao*-nets and thus even fits better into the semantical framework.

Replacing 5., 6. and 7. by 5.', 6.' and 7.' in Definition 6 as described here ensures that the *ao*-net AON_2 is a process. So the above considerations lead to the following alternative process definition and thus a change of the mapping α in Figure 3 (denoted by α' instead of α in Definition 9):

Definition 9 (Complete activator process). *A* complete activator process *(ca-process) of NI is an ao-net* $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, \mathit{Act}, l)$ *satisfying:*

1. $l(B) \subseteq P$ and $l(E) \subseteq T$.
2. The conditions in $\widetilde{B} = \{b \mid \exists e \in E : (b,e) \in \mathit{Act}\}$ are labelled by the special symbol $\curlywedge$.
3. $m_0 = l(\mathrm{MIN}_{\mathrm{AON}} \cap B)$.
4. For all $e \in E$, ${}^\bullet l(e) = l({}^\bullet e \cap B)$ and $l(e)^\bullet = l(e^\bullet \cap B)$.
5.' For all $b \in \widetilde{B}$, there are unique $g, h \in E$ such that ${}^\bullet b + b^\bullet = \{g\}$, $(b,h) \in \mathit{Act}$ and additionally $l(g) \multimap l(h)$ or ${}^\bullet l(h) \cap l(g)^\bullet \cap {}^- z \neq \emptyset$ for a $z \in T$.
6.' For all $e, f \in E$, if $f \multimap\!\!\bullet\, e$ then there is exactly one $c \in \widetilde{B}$ such that $f \multimap\!\!\bullet\, e$ through c.
7.' For all $e \in E$ and $S \in \mathrm{WSL}(\mathrm{AON})$, if ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b,e) \in \mathit{Act}\} \subseteq S$ then $l(S \cap B) \leq {}^- l(e)$.

The set of ca-processes of NI is denoted by $\alpha'(NI)$. For $\mathrm{AON} \in \alpha'(NI)$ *the generated so-structure* $\kappa(\mathrm{AON})$ *is called a run (associated to* AON*).*

Note that the requirements 1.,3.,4. of Definition 6 are preserved in Definition 9 and thus also ca-processes constitute a conservative generalization of common p/t-net processes. Omitting the $\curlywedge$-conditions from a ca-process AON leads to the so called underlying process $\mathrm{Und}(\mathrm{AON})$ of AON, which is a process of $\mathrm{Und}(NI)$. We will show now as the main result of this paper that the ca-process definition actually fulfills the aim of completeness. Due to lack of space, we only give a sketch of the proof (which has three pages). The complete proof can be found in the technical report [10].

Theorem 1. *For every enabled LSO* $\mathcal{S} = (E, \prec, \sqsubset, l)$ *of a pti-net* NI *there exists a ca-process* $\mathrm{AON} \in \alpha'(NI)$ *whereas* $\mathcal{S}$ *is an extension of the run* $\kappa(\mathrm{AON})$.

Proof (Sketch). The LPO $\mathrm{lpo}_{\mathcal{S}} = (E, \prec, l)$ underlying $\mathcal{S}$ is enabled w.r.t. $\mathrm{Und}(NI)$. Thus there exists a process $\mathrm{UAON} = (B, E, R', l')$ of $\mathrm{Und}(NI)$ fulfilling that $\mathrm{lpo}_{\mathcal{S}}$ sequentializes the run $\kappa(\mathrm{UAON})$. The basic idea is now to construct an ao-net AON from UAON by adding all $\curlywedge$-conditions to UAON which can be added according to property 5.' while not producing causal dependencies contradicting $\mathcal{S}$. Then this ao-net $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ is the sought ca-process. It is clear that AON satisfies 1. - 4., 5.' and 6.'. Thus, it only remains to show that AON meets condition 7.' of Definition 9, i.e. that given $e \in E$ and $S \in \mathrm{WSL}(\mathrm{AON})$ with ${}^{\bullet}e \cup \{b \in \widetilde{B} \mid (b, e) \in Act\} \subseteq S$ it holds that $l(S \cap B) \leq {}^{-}l(e)$. For this, we fix a weak configuration C of AON with $S = \mathrm{MAR}(C)$ and show that $l(e)$ is executable in the pti-net after the occurrence of the transitions corresponding to events in C. To this end, we define a prefix C_{pre} of e in $\mathcal{S}$ containing as many events from C as possible. Using that $\mathcal{S}$ is enabled, we can deduce that $l(e)$ is executable in the pti-net after the occurrence of the transitions corresponding to events in C_{pre}: By Lemma 1 there is $lin \in lin(\mathcal{S})$ such that C_{pre} is a prefix of e w.r.t. lin. Because $\mathcal{S}$ is enabled the total linear so-structure $lin = \tau_1 \ldots \tau_n$ represents an enabled synchronous step sequence of NI with $C_{pre} = \bigcup_{j=1}^{i-1} \tau_j$ and $e \in \tau_i$ (for $i \in \{1 \ldots n\}$). This implies that e can occur after C_{pre}. Finally C_{pre} can be transformed in several steps into the set C and in each step it can be shown that the transformation does not disable $l(e)$. $\square$

In the following we briefly explain that the other aims of the semantical framework are still fulfilled by the new process definition:

Soundness: Using Proposition 5.19 of [13] it is easy to see that every run is enabled, i.e. if $\mathrm{AON} \in \alpha'(NI)$, then $\phi(\epsilon(\kappa(\mathrm{AON}))) \subseteq \omega(NI)$.
Consistency of runs and processes: Processes and runs generate the same step sequences, i.e. if $\mathrm{AON} \in \alpha'(NI)$, then $\epsilon(\kappa(\mathrm{AON})) = \lambda(\mathrm{AON})$ (that means the rules for constructing causal relationships between events from processes as shown in Figure 2 are correct). This follows since in proposition 5.19 of [13] this relation was shown for arbitrary ao-nets (note here that the construction rules of the involved mappings λ, κ and ϵ have not changed in contrast to [13], only the process definition constituting the starting point of this relation is changed).

Weak completeness: Any execution of the pti-net ($\mathcal{EX}$) given by $\omega(NI)$ is generated from a ca-process, i.e. for any execution $\sigma \in \mathcal{EX}$ there exists an ca-process $\mathrm{AON} \in \alpha'(NI)$ with $\sigma \in \phi(\lambda(\mathrm{AON}))$ ($\omega(NI) \subseteq \bigcup_{\mathrm{AON} \in \alpha'(NI)} \phi(\lambda(\mathrm{AON}))$). This also holds for ca-processes, because this is the relation generalized in comparison to a-processes (the aim of completeness is a generalization of the weak completeness property).
Runs are reconstructible from step sequences: Each run is the intersection of all observations it generates, i.e. $\iota \circ \epsilon$ reconstructs a run. This relation holds because of the generalization of Szpilrajns theorem to so-structures described in the preliminaries (note that in this context nothing is changed in contrast to [13]).
Construction of processes from step sequences: There is no obvious way to generalize the constructive definition of π from [13] because especially the new requirement 6.' of Definition 9 is problematic: Now it is no more mandatory but optional to introduce $\curlywedge$-conditions between certain transitions (the transition candidates can be identified with 5.') and one has to check whether 7.' holds (7. holds by construction). There is the following constructive process definition that is based directly on the axiomatic definition: Given an enabled step sequence σ of NI a ca-processes can be generated as follows:

- Construct a usual p/t-net process of $Und(NI)$ (based on an occurrence net) starting from σ.
- Introduce arbitrary $\curlywedge$-labeled conditions in accordance with 5.' and 6.' of Definition 9.
- Check 7.' of Definition 9: if it is fulfilled the construction is finished, else perform the next step.
- Introduce further $\curlywedge$-labeled conditions in accordance with 5.' and 6.' of Definition 9, then go back to the previous step.

All processes constructible with this algorithm produce the set of ca-processes $\pi'(\sigma)$ generated by σ. Moreover, the ca-processes generated from a step sequence σ are the ca-processes having σ (provided with respective labels) as an execution. This algorithm always terminates because there are only finite many possible $\curlywedge$-labeled conditions in accordance with 5.' and 6.' of Definition 9. Introducing *all* such possible $\curlywedge$-conditions obviously leads to a ca-process, i.e. 7.' is then fulfilled in step 3. More precisely, the number of possible $\curlywedge$-conditions is at most quadratic in the number of events which means that the number of repetitions of the steps 3 and 4 of the algorithm is polynomial. Thus, only checking 7.' in step 3 may be not efficient, since there exists an exponential number of (weak) slices in the number of nodes. But current research results on a similar topic summarized in [14] show that there exists an algorithm polynomial in time solving this problem: In [14] we present an algorithm (based on flow theory) that can be used to calculate step 3 in polynomial time (of degree $O(n^3)$). Therefore, with this construction the requirements interrelated with the mapping π in the semantical framework of Figure 3 are also fulfilled.

5 Conclusion

In this paper we have developed a general semantical framework that supports the definition of process semantics and respective causal semantics for arbitrary Petri net classes.

The framework is based on the semantical framework from [13] additionally requiring that process semantics should be complete w.r.t. step semantics: Each causal structure which is consistent to step semantics – such causal structures we call enabled – should be generated from a process net. Since for the description of causal net behavior of pti-nets under the a-priori semantics labeled so-structures are applied, the notion of enabled so-structures has been introduced. We were able to show that the process definition for pti-nets from [13] is not complete w.r.t. step semantics and to identify a structural generalization of this process definition which is complete (while still satisfying all the other requirements of the framework of [13]).

Possible further applications of the results of this paper are on the one hand the usage of the semantical framework on further Petri net classes in order to check existing process semantics and to evolve new process semantics. In the context of the paper, this is in particular interesting for existing inhibitor net semantics [19,6,2,12,13,8]: While most aims of [13] are checked for those process semantics, the new aim of completeness is not (probably because this is the most complicated aim). Nevertheless a lot of these process semantics seem to satisfy the aim of completeness (at least for the process semantics of elementary nets with inhibitor arcs under the a-priori semantics as well as the a-posteriori semantics there are formal proofs [9]). On the other hand the ca-processes of this paper constitute a process definition for pti-nets under the a-priori semantics expressing minimal causalities and can thus be useful e.g. for model checking algorithms based on unfoldings.

References

1. Billington, J.: Protocol specification using p-graphs, a technique based on coloured petri nets. In: Reisig, W., Rozenberg, G. [20] pp. 293–330
2. Busi, N., Pinna, G.M.: Process semantics for place/transition nets with inhibitor and read arcs. Fundam. Inform. 40(2-3), 165–197 (1999)
3. Donatelli, S., Franceschinis, G.: Modelling and analysis of distributed software using gspns. In: Reisig, W., Rozenberg, G. [20], pp. 438–476
4. Goltz, U., Reisig, W.: The non-sequential behaviour of petri nets. Information and Control 57(2/3), 125–147 (1983)
5. Goltz, U., Reisig, W.: Processes of place/transition-nets. In: Díaz, J. (ed.) Automata, Languages and Programming. LNCS, vol. 154, pp. 264–277. Springer, Heidelberg (1983)
6. Janicki, R., Koutny, M.: Semantics of inhibitor nets. Inf. Comput. 123(1), 1–16 (1995)
7. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. In: Monographs in Theoretical Computer Science, vol. 1-3, Springer, Heidelberg (1992) (1994) (1997)
8. Juhas, G.: Are these events independend? it depends! Habilitation (2005)
9. Juhas, G., Lorenz, R., Mauser, S.: Synchronous + concurrent + sequential = earlier than + not later than. In: Proceedings of ACSD 2006, pp. 261–270 (2006)
10. Juhás, G., Lorenz, R., Mauser, S.: Complete process semantics of inhibitor net (2007) Technical report http://www.informatik.ku-eichstaett.de/mitarbeiter/lorenz/techreports/complete.pdf
11. Kiehn, A.: On the interrelation between synchronized and non-synchronized behaviour of petri nets. Elektronische Informationsverarbeitung und Kybernetik 24(1/2), 3–18 (1988)

12. Kleijn, H.C.M., Koutny, M.: Process semantics of p/t-nets with inhibitor arcs. In: Nielsen, M., Simpson, D. (eds.) ICATPN 2000. LNCS, vol. 1825, pp. 261–281. Springer, Heidelberg (2000)
13. Kleijn, H.C.M., Koutny, M.: Process semantics of general inhibitor nets. Inf. Comput. 190(1), 18–69 (2004)
14. Lorenz, R., Bergenthum, R., Mauser, S.: Testing the executability of scenarios in general inhibitor nets. In: Proceedings ACSD 2007 (2007)
15. Peterson, J.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, Englewood Cliffs (1981)
16. Szpilrajn, E.: Sur l'extension de l'ordre partiel. Fundamenta Mathematicae 16, 386–389 (1930)
17. Vogler, W.: Modular Construction and Partial Order Semantics of Petri Nets. LNCS, vol. 625. Springer, Heidelberg (1992)
18. Vogler, W.: Partial words versus processes: a short comparison. In: Rozenberg, G. (ed.) Advances in Petri Nets: The DEMON Project. LNCS, vol. 609, pp. 292–303. Springer, Heidelberg (1992)
19. Vogler, W.: Partial order semantics and read arcs. In: Privara, I., Ruzicka, P. (eds.): MFCS 1997. LNCS, vol. 1295, pp. 508–517. Springer, Heidelberg (1997)
20. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets II: Applications, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996. LNCS, vol. 1492. Springer, Heidelberg (1998)