Fundamenta Informaticae XX (2008) 1–37
IOS Press

Complete Process Semantics of Petri Nets

Gabriel Juhás
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology, Bratislava, Slovakia
gabriel.juhas@stuba.sk

Robert Lorenz, Sebastian Mauser *
Department of Applied Computer Science
Catholic University of Eichstätt-Ingolstadt, Germany
{*robert.lorenz, sebastian.mauser*}*@ku-eichstaett.de*

Abstract. In the first part of this paper we extend the semantical framework proposed in [22] for process and causality semantics of Petri nets by an additional aim, firstly mentioned in the habilitation thesis [15]. The aim states that causality semantics deduced from process nets should be *complete* w.r.t. step semantics of a Petri net in the sense that *each* causality structure which is *enabled* w.r.t. step semantics corresponds to some process net.

In the second part of this paper we examine several process semantics of different Petri net classes w.r.t. this aim. While it is well known that it is satisfied by the process semantics of place/transition Petri nets (p/t-nets), we show in particular that the process semantics of p/t-nets with weighted inhibitor arcs (PTI-nets) proposed in [22] does not satisfy the aim. We develop a modified process semantics of PTI-nets fulfilling the aim of completeness and also all remaining axioms of the semantical framework. Finally, we sketch results in literature concerning the aim of completeness for process definitions of various further Petri net classes.

The paper is a revised and extended version of the conference paper [18].

Keywords: Petri Net, Inhibitor Net, Process Semantics, Causal Semantics, Completeness

*This paper was supported by the German research Council, Project *Synops*

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, synchronicity and causal dependency among events in system executions. Events are considered to be *concurrent* in a certain state of the system if they can occur at the same time and in arbitrary order in this state. They are *synchronous* in a certain state if they only can occur at the same time. Non-sequential semantics can be given as the set of executions of the system, where executions are represented by appropriate causal structures relating events. Therefore, such non-sequential semantics is also called *causal semantics*. Whether a given causal structure is an execution of a given system or not can be deduced from the so called *step semantics* of the system model.

For the definition of step semantics it is stated which actions can occur in a certain state of the system *at the same time*, and how the system state is changed by their occurrence (yielding the step occurrence rule). Such actions form a *step (of actions)*. Given an initial state of a system, from the step occurrence rule sequences of steps which can occur in this state 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 action occurrences in one step are observed at the same time and the action occurrences in different steps are observed in the order given by the step sequence.

Causal semantics aim at representing arbitrary concurrency among action occurrences (or events). Since step sequences can only describe a very restricted class of concurrency, a causal structure usually allows (generates) several different observations in the form of step sequences. In particular, the occurrence of events, which are concurrent in a causal structure, can be observed synchronously or also in arbitrary order. Therefore, a given causal structure 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. Here, only "full" observations are considered, i.e. observations which contain all events of a causal structure. Such causal structures are called *enabled (w.r.t. step semantics)*.[1] If causality is added to an enabled causal structure, resulting in a so called *extension* of the causal structure, the extension is again enabled, since it generates fewer observations. The other way round, if causality is removed from an enabled causal structure, the resulting causal structure need not be again enabled, since it generates more observations. If removing causality always leads to causal structures which are not enabled, an enabled causal structure is called *minimally enabled*. Minimally enabled causal structures express *minimal* causal dependencies among events.

Figure 1 shows examples of enabled, minimally enabled and not enabled causal structures w.r.t. a given step semantics. Part (a) shows a set of step sequences over the action names a and c. The same action may occur several times at the same time. Thus a step can be formally given as a multi-set over the set of actions $\{a, c\}$. Actions in one step are considered to be synchronous. Observe that prefixes are not considered in the shown set of step sequences. Since often step semantics is prefix-closed, e.g. for Petri nets, this set may be interpreted as a fragment of some step semantics. In the example a causal

[1] If step semantics of place/transition Petri nets (p/t-nets) is considered, causal structures are given by labeled partial orders (LPOs) (also called pomsets [28] or partial words [11]). In [20], *enabled LPOs* w.r.t. the step semantics of p/t-nets were defined.

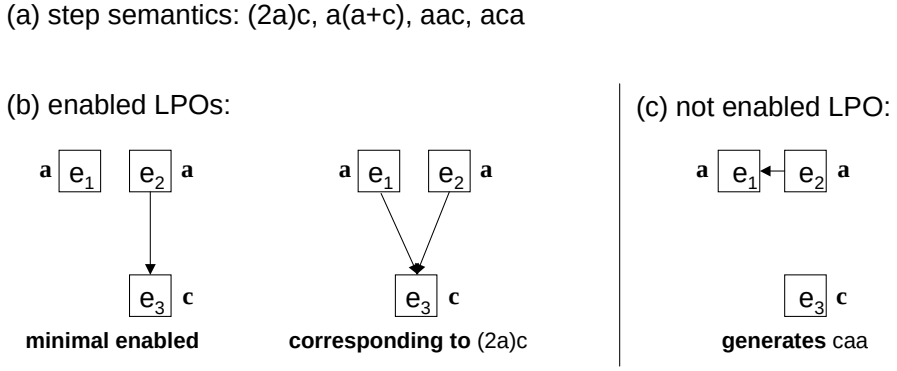

Figure 1. Causal structures which are enabled, minimally enabled and not enabled, given by labeled partial orders.

structure is given by a partial order between the events e_1, e_2 and e_3 labeled by the action names a and c, a so called *LPO (labeled partial order)*. LPOs may or may not be *enabled* w.r.t. the given step semantics. They model an "earlier than"-relation between events, expressed by solid arcs. Unordered events are considered to be concurrent. Note that in LPOs it is not possible to distinguish concurrent from synchronous behavior. Therefore, synchronous transition occurrences in step sequences correspond to concurrent transition occurrences in partial orders. The right LPO in part (b) corresponds to the step sequence $(2a)c$. It is enabled, but not minimally enabled, since removing the arc between e_1 and e_3 gives the left LPO in part (b), which is also enabled. This left LPO cannot be expressed through a step sequence. It is *minimally enabled* and generates all shown step sequences in (a). The LPO in part (c) is not enabled because it generates the step sequence caa not belonging to the step semantics given in part (a).

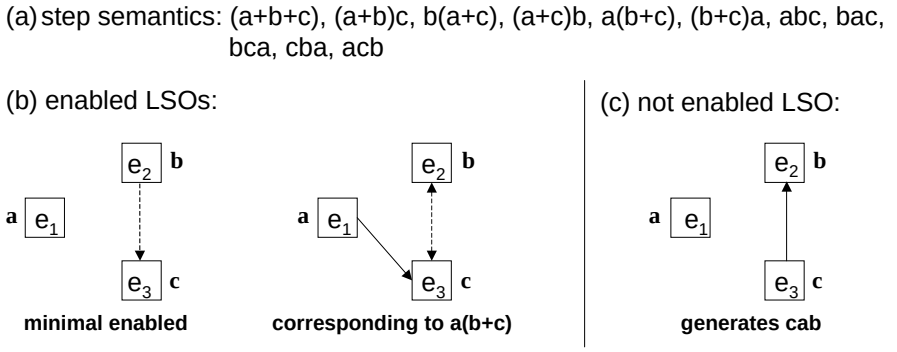

Figure 2. Causal structures which are enabled, minimally enabled and not enabled, given by labeled stratified order structures.

Figure 2 shows further examples of enabled, minimally enabled and not enabled causal structures w.r.t. a given step semantics. Here, a causal structure is given by a stratified order structure (so-structure) over the set of events $\{e_1, e_2, e_3\}$ labeled by action names a, b and c, a so called *LSO (labeled so-structure)*. LSOs generalize LPOs. They model an "earlier than"-relation (solid arcs) and a "not later

than"-relation between events (dashed arcs). Two "not later than"-ordered events can be observed in the respective order or also synchronously, but not in the reverse order. Synchronicity can be expressed by cyclic "not later than"-relations. The right LSO in part (b) represents the step sequence $a(b+c)$, where e_2 and e_3 can only occur synchronously, but not sequentially. It is enabled, but not minimally enabled, since removing the dashed arc from e_3 to e_2 and the solid arc from e_1 to e_3 gives the left LSO in part (b), which is also enabled. This left LSO in part (b) cannot be expressed through a step sequence. It is minimally enabled.

Causal semantics consisting only of enabled causal structures we call *sound (w.r.t. step semantics)*. On the other hand, *all* enabled causal structures represent valid behavior of the system. Among the set of all enabled causal structures, the minimally enabled causal structures give full information on causal dependencies and concurrency. Causal semantics which contains *all minimally* enabled causal structures we call *complete (w.r.t. step semantics)*. Given a complete causal semantics, each enabled causal structure is extension of some minimally enabled causal structure in the causal semantics. An important aim for the definition of causal semantics of particular formalisms describing concurrent systems is soundness and completeness w.r.t. step semantics.

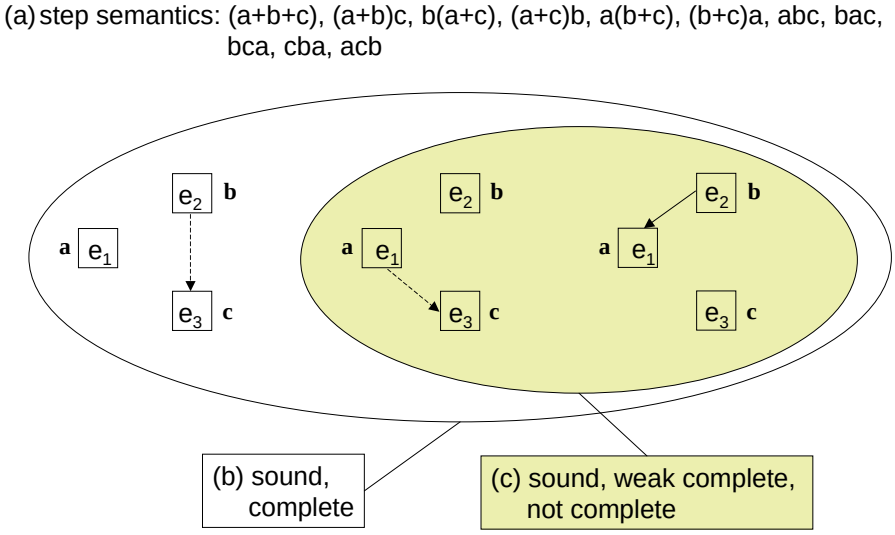

Figure 3. Causal semantics, which are complete and not complete.

Figure 3 shows examples of causal semantics which are complete and not complete w.r.t. a given step semantics. Both causal semantics are represented by a set of minimally enabled LSOs. The smaller set shown in (c) is not complete, since it does not include the minimally enabled LSO on the left side. But it satisfies a weaker form of completeness we call *weak completeness* here: The LSOs in the smaller set still generate *all* step sequences shown in part (a). The bigger set shown in (b) is complete, i.e. there is no other enabled LSO which is not an extension of one of the shown LSOs.

The completeness aim can be carried over to any non-sequential semantics (of concurrent system models) which define causal semantics. In this paper we consider soundness and completeness for process semantics and causal semantics of Petri nets. Petri nets are one of the most prominent formalisms for understanding the concurrency phenomenon and for modeling of real concurrent systems in many

application areas [14]. The most important and well-known concept of non-sequential semantics of Petri nets is process semantics [9, 10]. Process semantics are given by sets of process nets, which are Petri nets representing transition occurrences by events (transitions of process nets) with explicit pre-, post- and side-conditions (places of process nets). These conditions represent token occurrences (in places of the original net) and other causal dependencies (for example context arcs).

Process semantics were first developed for place/transition Petri nets (p/t-nets) [9, 10]. For such nets, a process net can be translated to a partial order between transition occurrences (events labeled by transition names) by removing all conditions and keeping the partial order (given by the flow relation of the process net) for the events. Such a *labeled partial order (LPO)* is called *run associated to the process*. In a run, unordered events are considered to be concurrent. Through the definition of runs, the process semantics yields a causal semantics: A run describes valid causal behavior of the p/t-net. It was shown in [20] that an LPO is enabled if and only if it is a (partial order) extension of a run (see also [34, 35, 15]). This implies, that each run is enabled and that the set of runs includes all minimally enabled LPOs, i.e. process semantics from [9, 10] and induced causal semantics given by runs are sound and complete. Thus, the process net based causality semantics of p/t-nets satisfies strong consistency properties w.r.t. step semantics. Moreover, they have an intuitive graphical representation, can be efficiently constructed and only reflect causal dependencies among transition occurrence which are existent in the net.[2] These are the essential properties of p/t-net processes justifying their success as non-sequential semantics describing system behavior.

Since the basic developments of Petri nets, more and more different *Petri net classes* for various applications have been proposed, extending their modeling features by additional structural elements which modify the step occurrence rule. It turned out to be not easy to define process semantics (and related causality semantics in the form of runs), having all the advantages of p/t-net processes, for such net classes. Especially the completeness aim is very hard to prove already in the most simple case of p/t-nets (see [20, 34, 35, 15], whether completeness holds for p/t-net processes was an open question for many years). For several of the process semantics proposed so far for certain Petri net classes, completeness is still an open problem (for details see Section 4).

An important p/t-net extension is that by inhibitor arcs, proposed in several variants. As stated in [27], "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, 8]. Accordingly, for these net classes various authors proposed process definitions regarding different interpretations of the occurrence rule of inhibitor nets. In [22] a-priori processes for PTI-nets (the most general class of p/t-nets with inhibitor arcs) are defined. In the case of inhibitor nets under the so-called a-priori semantics [12], so called *(labeled) so-structures (LSOs)* represent the causal semantics (Figures 2, 3). Recently, we could show in [18] that the a-priori process definition of [22] is not complete by identifying a minimally enabled LSO of a PTI-net not being a run of the net. We developed an alternative complete process definition which fulfills also all other semantical consistency properties stated in [22].

In order to provide a common scheme for the definition of process semantics of Petri nets, in [22] (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. Any process semantics should fulfill the reasonable aims stated by the framework. These aims are reduced to

[2] In contrast, in enabled LPOs transition occurrences may be ordered which are not causally dependent in the net, since also extensions of runs are enabled LPOs

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. But this general framework does not regard the described aim of completeness. Instead another aim of the framework from [22] requires a kind of *weak completeness*, saying that each step sequence in the step semantics should be generated by some process net (Figure 3). We extend this framework by adding the described completeness aim.

The paper is structured as follows. In Section 2 we recall the semantical framework from [22] in a new terminology, formally add the property of completeness and illustrate the modified framework by a small example concerning process semantics of p/t-nets. In Section 3 we consider the modified framework for PTI-nets w.r.t. the a-priori semantics. After introducing step semantics of such nets, we introduce causal semantics in form of enabled LSOs. Then we show that the process semantics from [22] is not complete and develop a new complete process semantics. This process semantics is then shown to fulfil the whole semantical framework from Section 2. Finally, in Section 4 further Petri net classes are discussed in the context of the presented semantical framework focusing on the new property of completeness.

2. The Semantical Framework

In [22] a general framework for dealing with process semantics of Petri nets was proposed (see Figure 4, left part). It aims at a support for a systematic, consistent development of process and causality semantics for various Petri net classes using a common scheme. In this section we restate this framework proposing a new terminology and formally add the aim of completeness as described in the introduction.

In Figure 4 the nodes represent different semantics of a given Petri net model. The arrows indicate functions that define and relate the different semantics. They represent the consistency requirements for process semantics according to this framework. The abbreviations mean the following:

- $N \in \mathcal{PN}$ represents a *Petri net model* of a given Petri net class $\mathcal{PN}$ together with an operational step occurrence rule.
- $\mathcal{EX}$ is the set of executions generated by a net $N \in \mathcal{PN}$ in form of *enabled step sequences* in accordance to its step occurrence rule. $\omega(N) = \mathcal{EX}$ assigns to a net N its set of executions.
- $\mathcal{LAN}$ defines (axiomatically) the *process semantics* of a net $N \in \mathcal{PN}$ given by process nets. Process nets are labeled acyclic occurrence nets which have assigned an operational step occurrence rule. Labels of places and transitions of an occurrence net refer to places and transitions of N. $\alpha(N) = \mathcal{LAN}$ assigns to a net N its set of process nets.
- $\mathcal{LEX}$ is the set of *labeled step sequences* generated by process nets $O \in \mathcal{LAN}$ in accordance to their step occurrence rule. For an occurrence net O, $\lambda(O)$ is the set of *labeled step sequences* generated by O, which contain all events of O. That means $\mathcal{LEX} = \bigcup_{O \in \mathcal{LAN}} \lambda(O)$. In labeled step sequences from $\mathcal{LEX}$, actions refer to event names of a process net, whereas labels of actions refer to transitions of $N \in \mathcal{PN}$. Observe that λ is generally defined for occurrence nets, not only for process nets.
- $\mathcal{RUN}$ defines the set of *runs* associated to process nets in $\mathcal{LAN}$ describing net behavior through causality relations between events. For an occurrence net O, $\kappa(O)$ is the labeled causal structure associated to O. That means $\mathcal{RUN} = \{\kappa(O) \mid O \in \mathcal{LAN}\}$. Observe that κ is generally defined for occurrence nets, not only for process nets.
- The mapping ϕ abstracts from action names in a labeled step sequence, producing a step sequence over the set of its labels. Observe that ϕ is defined for arbitrary labeled step sequences, not only for labeled step sequences in $\mathcal{LEX}$.
- For a labeled causal structure R, $\epsilon(R)$ defines the set of *labeled step sequences* generated by R, which contain all events of R. Observe that ϕ is generally defined for labeled causal structures, not only for runs.
- ι defines a method to construct a labeled causal structure from a set of labeled step sequences having the same set of actions with the same labeling.
- π defines a method to construct a set of process nets from an enabled step sequence. Note that π not only depends on $\mathcal{EX}$ but also on $N \in \mathcal{PN}$.

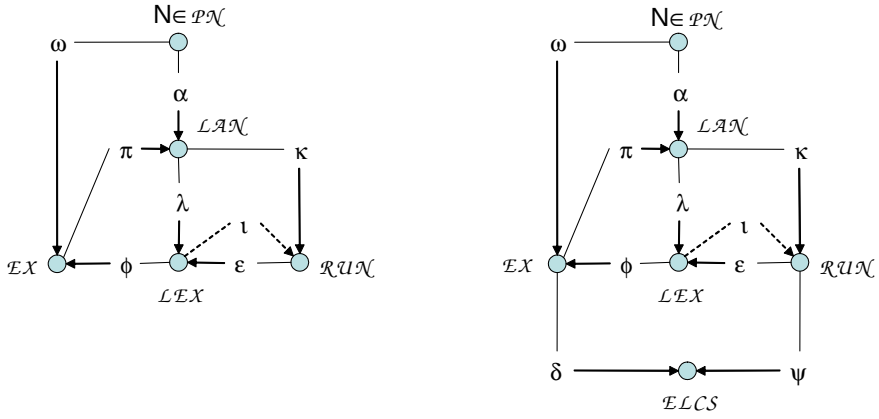

Figure 4. Adding the completeness aim to the semantical framework of [22].

The framework in [22] is condensed to five properties that have to be checked in each particular setting:

- The mappings in Figure 4 (a) returning sets (namely $\omega, \alpha, \lambda, \epsilon, \pi$) do not return the empty set.
- The mappings in Figure 4 (a) are total.
- The mappings in Figure 4 (a) commute (called *consistency* in [22]), i.e.
  - *Consistency of runs and processes*: $\forall O \in \mathcal{L}AN : \ \lambda(O) = \epsilon(\kappa(O))$ (called *fitting* in [22]).
  - *Soundness*: $\forall R \in \mathcal{RUN} : \ \phi(\epsilon(R)) \subseteq \mathcal{EX}$.
  - *Weak completeness*: $\mathcal{EX} \subseteq \bigcup_{R \in \mathcal{RUN}} \phi(\epsilon(R))$.

  Note that weak completeness implies for $S \in \mathcal{EX}$ that $S \in \bigcup_{R \in \mathcal{RUN}} \phi(\epsilon(R))$. If $\kappa(O) = R$, then from the consistency of runs and processes we get that $\lambda(O) = \epsilon(R)$, i.e. $S \in \bigcup_{O \in \alpha(N)} \phi(\lambda(O))$. The other way round, for $S \in \phi(\lambda(O))$ we get $S \in \phi(\epsilon(R))$ for $R = \kappa(O)$ from the consistency of runs and processes. From soundness this implies $S \in \mathcal{EX}$. Altogether, the above consistency properties imply
  $$\bigcup_{O \in \alpha(N)} \phi(\lambda(O)) = \mathcal{EX} = \omega(\mathcal{PN}).$$
- *Runs are reconstructible from step sequences*: $\forall R \in \mathcal{RUN} : \ \iota(\epsilon(R)) = R$ (called *representation* in [22]).
- *Construction of processes from step sequences*: $\forall S \in \mathcal{EX} : \ \pi(S) = \{O \in \mathcal{L}AN \mid S \in \phi(\lambda(O))\}$

On the one hand, *soundness* ensures that each run is consistent with the step semantics in the sense defined by $\phi \circ \epsilon$. On the other hand, *each* labeled causal structure, which is consistent with the step semantics in this sense, represents valid behavior of the given Petri net model. But there is no property in the framework of [22] requiring that such causal structures are modeled by the process semantics. In the following, we call such causal structures *enabled*.

Definition 2.1. (Enabled labeled causal structure)
A labeled causal structure R is called *enabled (w.r.t. step semantics)* if $\phi(\epsilon(R)) \subseteq \mathcal{EX}$. The mapping δ assigns to a set of step sequences $\mathcal{S}$ the set of labeled causal structures $\{R \mid \phi(\epsilon(R)) \subseteq \mathcal{S}\}$. $\mathcal{ELCS} = \delta(\mathcal{EX})$ defines the set of enabled labeled causal structures.

With this new terminology, soundness means that each run is enabled. Each labeled causal structure which has *more causality than a run* is enabled, too. Formally, we define the relation of "having more causality" among labeled causal structures within the framework as follows:

Definition 2.2. (Extension of a labeled causal structure)
A labeled causal structure R has *more causality than* a labeled causal structure R' if $\phi(\epsilon(R)) \subseteq \phi(\epsilon(R'))$. In this case, R is called *extension of* R'. If $\phi(\epsilon(R)) \subsetneq \phi(\epsilon(R'))$ then R is called *strict extension of* R'. For a labeled causal structure R, $\psi(R)$ denotes the set of all extensions of R.

It follows by definition that extensions of enabled causal structures are enabled.

Definition 2.3. (Minimally enabled labeled causal structure)
An enabled labeled causal structure R is *minimally enabled* if it is not a (strict) extension of another enabled labeled causal structure.

It holds by construction that a process semantics is sound if and only if all extensions of runs are enabled, i.e.

$$(\forall R \in \mathcal{RUN} :\ \phi(\epsilon(R)) \subseteq \mathcal{EX}) \Longleftrightarrow (\delta(\mathcal{EX}) = \mathcal{ELCS} \supseteq \bigcup_{R \in \mathcal{RUN}} \psi(R)).$$

But there still may be enabled causal structures which are not extensions of a run. This is the case if and only if there is a minimally enabled causal structures which is not extension of a run, since extensions of enabled causal structures are enabled. Completeness means, that there are no such minimally enabled causal structures.

Definition 2.4. (Aim of completeness)
Process semantics is called *complete (w.r.t. step semantics)* if $\delta(\mathcal{EX}) = \mathcal{ELCS} \subseteq \bigcup_{R \in \mathcal{RUN}} \psi(R)$.

Note that completeness implies weak completeness, since each step sequence of the net corresponds to a step sequence generated by an enabled causal structure. The aims of completeness, weak completeness and soundness can be considered as properties of the set $\mathcal{RUN}$. Together with soundness, completeness can also be characterized through minimally enabled causal structures:

Theorem 2.1. Let $\mathcal{RUN}$ be sound. Then $\mathcal{RUN}$ is complete if and only if the set of minimally enabled causal structures coincides with the set of minimal runs.[3]

[3] A run is *minimal* if it is not extension of another run.

Thus, if a process semantics is sound and complete, then for arbitrary valid causal behavior of the net, there are runs and processes which express this behavior. In other words, minimal causal dependencies in a net are reflected in the process semantics.

To integrate the aim of completeness into the semantical framework, we add new relations labeled by δ and ψ and the new node $\mathcal{ELCS}$ to Figure 4 (right part). The absence of the aim of completeness in the framework of [22] (Figure 4 left part) allows process definitions that do not necessarily represent minimal causal behavior. According to [22] a process definition that equals the operational step semantics ("processes are step sequences") is a valid process semantics, because it is sound and weakly complete (but not complete). But the set of step sequences would be a reasonable process semantics only if it reflects minimal causal dependencies. Generally, process definitions not producing minimal causalities are less expressive and do not give all possible system runs. In this sense, the property of weak completeness, only requiring that each step sequence is modeled in the process definition, is not enough. Therefore in our new version of the semantical framework (Figure 4 right part) the aim of completeness is introduced solving this problem.

Remark 2.1. There is the following equivalent formalization of the completeness aim, which avoids the notion of enabled causal structures:

Process semantics is called complete (w.r.t. step semantics) if each labeled causal structure S with $\phi(\epsilon(S)) \subseteq \mathcal{EX}$, which is minimal with this property, is a run.

This formulation is more compact and could be illustrated in the framework by an arc from $\mathcal{EX}$ to $\mathcal{RUN}$. On the other side, it does not conform with the historical development of non-sequential semantics of Petri nets. Namely, in the case of p/t-nets, the notion of enabled LPOs as a description of non-sequential behavior [28, 11, 20] was introduced independently from occurrence nets and process nets [9, 10]. Enabled LPOs have the advantage that they provide a causal semantics which is complete "by construction". The exact relation between enabled LPOs and runs underlying process nets was unclear for several years. Finally, it could be shown that an LPO is an extension of a run if and only if it is enabled [20]. This result corresponds to the notion of completeness we chose. Altogether, the concepts of runs and enabled LPOs as a description of non-sequential behavior existed independently from each other and we decided to follow this terminology in this paper. In particular, we did not want to disregard the research on enabled causal structures.

We finish this section with a small example discussing process semantics for p/t-nets.

Example 2.1. (p/t-nets)
For p/t-nets the completeness aim is fulfilled (as mentioned). Figure 5 shows an example of the semantics of a p/t-net as occurring in the framework of Figure 4 right part; in this figure, we neglect "prefixes", i.e. always considering processes, runs, enabled causal structures and (labeled) step sequences of maximal length. The mappings ω, α and δ define the semantics $\mathcal{EX}$, $\mathcal{LAN}$ and $\mathcal{ELCS}$ as shown. The mapping κ assigns to each process net in $\mathcal{LAN}$ a run in $\mathcal{RUN}$ by omitting the places in the process net and keeping the order between transition occurrence given by the flow relation. It holds $\kappa(O_i) = R_i$ for $i = 1, 2$. The mapping λ assigns labeled step sequences of the set $\mathcal{LEX}$ to a process net in $\mathcal{LAN}$ through considering a process net as a marked elementary net (initially the conditions, which are minimal w.r.t. the flow relation, are marked) and applying the step occurrence rule for such nets. The mapping ϵ assigns the set of all labeled step sequences (from $\mathcal{LEX}$) to an LPO in $\mathcal{RUN}$, which add causality to the LPO (events in one step are considered to be concurrent). The event names indicate which labeled step sequences

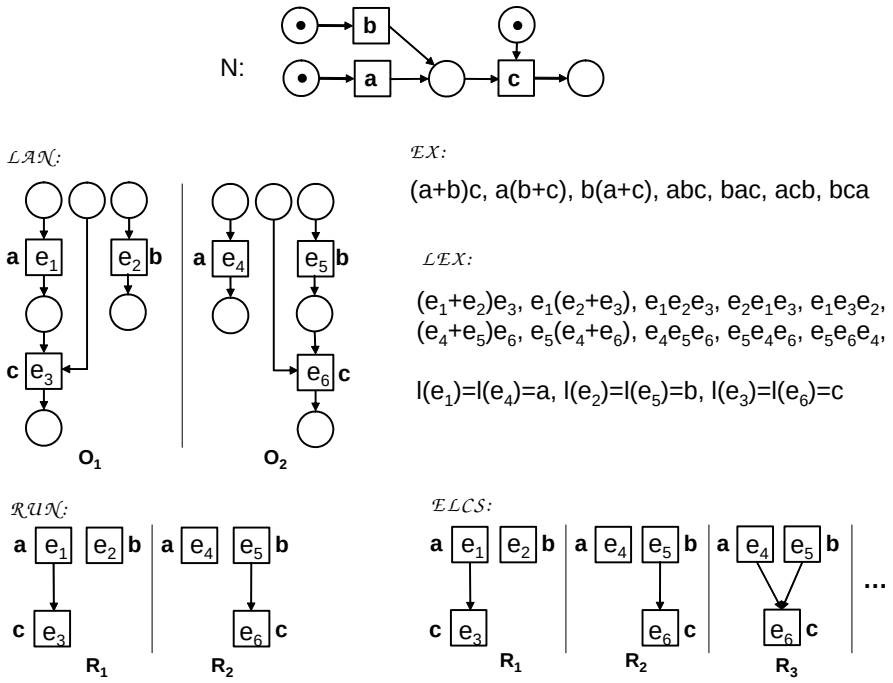

Figure 5. Different semantics of a p/t-net.

are generated by which process net resp. run. The mapping ι assigns an LPO to a set of labeled step sequences Σ (with set of events V and labeling function l) in $\mathcal{LEX}$ by $\iota(\Sigma) = (V, \{(v, v') \in V \times V \mid \forall \sigma = \tau_1 \dots \tau_n \in \Sigma : (\exists i < j : v \in \tau_i, v' \in \tau_j)\}, l)$. According to Szpilrajn's theorem [32] there holds $\iota(\epsilon(R_1)) = R_1$ and $\iota(\epsilon(R_2)) = R_2$. The mapping ϕ simply abstracts from the individuality of events in a labeled step sequence of $\mathcal{LEX}$, for example $\phi((e_1 + e_2)e_3) = (a + b)c$. The mapping ψ assigns to an LPO in $\mathcal{RUN}$ the set of its extensions. For example, R_3 is an extension of R_2 via the additional arc (e_4, e_6). It is easy to observe that R_1 and R_2 are the only two minimally enabled LPOs of the shown p/t-net (as defined by δ). Thus $\mathcal{ELCS}$ is given by the set of all extensions of R_1 or R_2. Finally, π is an algorithm for the construction of a set of process nets from a step sequence. Namely, for a step sequence all process nets generating the given step sequence are computed. It holds, for example, $\pi(a(b + c)) = \{O_1\}$ and $\pi((a + b)c) = \{O_1, O_2\}$.

3. PTI-nets

In this section we recall the formal definitions of *PTI-nets* and their step semantics w.r.t. *the a-priori step occurrence rule*. We introduce *enabled labeled causal structures* corresponding to these step semantics. These are given by so called *labeled so-structures (LSOs)*. Then we show that *process nets of PTI-nets* w.r.t. the a-priori semantics according to [22] are not complete and develop a new complete process semantics. Finally, we prove that this new process semantics also fulfils all other requirements of the framework presented in the last section. Altogether, compared to [22], we redefine the mappings α and π and the sets $\mathcal{LAN}$ and consequently $\mathcal{RUN}$, and introduce new mappings δ and ψ and the new set $\mathcal{ELCS}$ in order to extend the framework of [22] as mentioned in the last section for PTI-nets.

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 $\mathbb{N}^X$ ($\mathbb{N}$ denotes the non-negative integers). 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 $Rel \subseteq X \times X$ by Rel^*, the transitive closure of Rel by Rel^+ and the composition of two binary relations Rel, Rel' over X by $Rel \circ Rel'$. Two elements $x, y \in X$ are called Rel-independent if $(x,y),(y,x) \notin Rel$, the set of all Rel-independent pairs of elements is denoted by $\mathrm{co}_{Rel} \subseteq X \times X$. A binary relation $Rel \subseteq X \times X$ is a partial order if $\forall x \in X : (x,x) \notin Rel$ (Rel is irreflexive) and $\forall x,y,z \in X : (x,y),(y,z) \in Rel \Longrightarrow (x,z) \in Rel$ (Rel is transitive). If Rel is a partial order, we also say that (X, Rel) is a partial order.

3.1. Basic Definitions

In this subsection we introduce PTI-nets together with their a-priori step semantics. That means, we specify the set $\mathcal{E}X$ and the mapping ω of the presented semantical framework.

Inhibitor nets are an extension of classical place/transition nets (p/t-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).

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 ${}^\bullet U(p) = \sum_{t \in U} U(t) \cdot {}^\bullet t(p)$ and $U^\bullet(p) = \sum_{t \in U} U(t) \cdot t^\bullet(p)$ for all $p \in P$. Analogously we can define pre- and post-multi-sets of multi-sets of places as multi-sets of transitions. We assume that each transition has non-empty pre- and post-multi-set. 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 3.1. (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 ${}^- U(p) = min(\{{}^- t(p) \mid t \in U\})$. A place p with ${}^- t(p) \neq \infty$ is called *inhibitor place* of t.

Note that $I(p,t) = k \in \mathbb{N}$ implies that t can only occur if p does not contain more than k tokens; $k = 0$ 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 graphical illustrations, inhibitor arcs are drawn with circles as arrowheads and annotated with their weights (Figure 6). Inhibitor arcs with weight ∞ are completely omitted and the inhibitor weight 0 is not shown in diagrams. In the a-priori semantics, the testing of inhibitor restrictions precedes the occurrence of (steps of) transitions.

Definition 3.2. (Step semantics $\omega(\mathcal{PN}) = \mathcal{EX}$)
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 $m_1, \dots, m_n$ such that $m \xrightarrow{U_1} m_1 \xrightarrow{U_2} \dots \xrightarrow{U_n} m_n$.

By $\omega(NI) = \mathcal{EX}_{NI}$ we denote the set of all enabled step sequences of a marked PTI-net NI.

3.2. Enabled Labeled Causal Structures

In this subsection we introduce enabled labeled causal structures for PTI-nets. That means, we define the set $\mathcal{ELCS}$ and the mappings δ, ϵ, ι, ϕ and ψ from the framework presented in the last section.

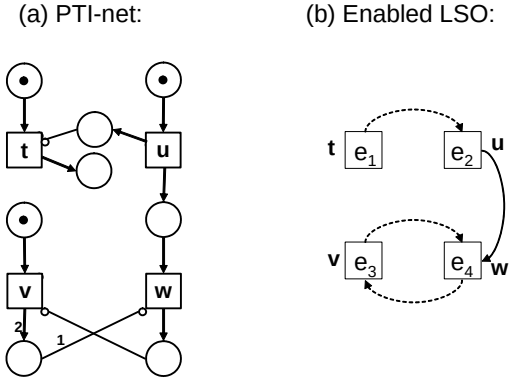

Figure 6. A PTI-net NI and an enabled LSO w.r.t. NI.

Figure 6 (a) 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 [12, 21, 22], "earlier than"-causality expressed by partial orders is not enough to describe causal semantics of PTI-nets w.r.t. the a-priori semantics. In Figure 6 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 $u \to t$. But they still can occur sequentially in order $t \to u$

or even synchronously, because of the occurrence rule "testing before execution". 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.

Such causal relationships between events can be described by so called so-structures. 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 as described above. Figure 6 (b) illustrates the above explained causal relationships in form of an so-structure. 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. 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 finite set (of *events*), and $\prec \subseteq V \times V$ and $\sqsubset \subseteq V \times V$ are binary relations on V. A rel-structure $\mathcal{S}' = (V, \prec', \sqsubset')$ is said to be an *extension* (or sequentialization) of another rel-structure $\mathcal{S} = (V, \prec, \sqsubset)$, written $\mathcal{S} \subseteq \mathcal{S}'$, if $\prec \subseteq \prec'$ and $\sqsubset \subseteq \sqsubset'$.

Definition 3.3. (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 $u, v, w \in V$:

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

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

In figures, $\prec$ is graphically expressed by solid arcs and $\sqsubset$ by dashed arcs. According to (C2) a dashed arc is omitted if there is already a solid arc. Moreover, we omit arcs which can be deduced by (C3) and (C4). It is shown in [12] 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.

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 $\prec_{S^\diamondsuit}$ 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 [12]).

For our purposes we will only consider *labeled so-structures* (*LSOs*). Events of an LSO represent transition occurrences of a Petri net. Formally LSOs are so-structures $\mathcal{S} = (V, \prec, \sqsubset)$ together with a *set of labels* T and a *labeling function* $l : V \to T$. The 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. In particular:

Definition 3.4. (The mapping ψ)
The mapping ψ assigns to each LSO $(V, \prec, \sqsubset, l)$ the set of all LSOs $(V, \prec', \sqsubset', l)$ such that $(V, \prec', \sqsubset')$ is an extension of $(V, \prec, \sqsubset)$.

For the definition of enabled LSOs according to the presented semantical framework, we need to define a mapping ϵ assigning to each LSO the set of labeled step sequences "generated" by this LSO and a mapping ϕ assigning a step sequence to a labeled step sequence. A *labeled step sequence over a finite set* V is a pair (σ, l) where $\sigma = U_1 \dots U_n$ is a sequence of disjoint subsets $U_i \subseteq V$ with $\bigcup_{i=1}^{n} U_i = V$ and $l : V \to T$ is a labeling function from V to a set of labels T. A labeled step sequences (σ, l) can be identified with special LSOs $\mathcal{S}_{(\sigma,l)}$. (σ, l) is generated by an LSO $\mathcal{S}$ if $\mathcal{S}_{(\sigma,l)}$ is an extension of $\mathcal{S}$. A labeled step sequence $(U_1 \dots U_n, l)$ corresponds to the step sequence $l(U_1) \dots l(U_n)$, where $l(U_i)$ is defined as a multi-set.

Definition 3.5. (The mappings ϵ and ϕ)
Let (σ, l) be a labeled step sequence over V with $\sigma = U_1 \dots U_n$. Define $\mathcal{S}_{(\sigma,l)} = (V, \prec_{(\sigma,l)}, \sqsubset_{(\sigma,l)}, l)$ by $\prec_{(\sigma,l)} = \bigcup_{i<j} U_i \times U_j$ and $\sqsubset_{(\sigma,l)} = ((\bigcup_{i=1}^{n} U_i \times U_i) \setminus id_V) \cup \prec_{(\sigma,l)}$.

For an LSO $\mathcal{S} = (V, \prec, \sqsubset, l)$, we denote $\epsilon(\mathcal{S}) = \{(\sigma, l) \mid \mathcal{S}_{(\sigma,l)} \text{ is an extension of } \mathcal{S}\}$.

For a labeled step sequence (σ, l) over V with $\sigma = U_1 \dots U_n$, we denote $\phi((\sigma, l)) = l(U_1) \dots l(U_n)$.

The step sequences in $\phi(\epsilon(\mathcal{S}))$ can be considered as observations of $\mathcal{S}$, where events within a step are observed at the same time (synchronously), and step occurrences are observed in the order given by the step sequence.

An LSO $\mathcal{S}$ is consistent with the step semantics $\mathcal{EX}$ of a given PTI-net if each such observation of $\mathcal{S}$ is a step occurrence sequence of the PTI-net in $\mathcal{EX}$. Such LSOs we call *enabled* (w.r.t. the given PTI-net).

Definition 3.6. (Enabled LSO)
An LSO $\mathcal{S}$ is enabled w.r.t. a marked PTI-net NI if $\phi(\epsilon(\mathcal{S})) \subseteq \mathcal{EX}_{NI}$. $\mathcal{ELCS}_{NI} = \delta(\mathcal{EX}_{NI})$ is the set of all enabled LSOs.

With this definition one can easily check that the LSO in Figure 6 is enabled. It generates the step sequences $tu(v + w)$ and $(t + u)(v + w)$, which are both enabled.

Finally, we define the mapping ι for the reconstruction of an LSO from the set of labeled step sequences generated by the LSO. The reconstruction works through intersection of causal relations.

Definition 3.7. (The mapping ι)
Let Σ be a set of labeled step sequences over a set V with labeling function l. We define $\iota(\Sigma) = (V, \bigcap_{(\sigma,l)\in\Sigma} \prec_{(\sigma,l)}, \bigcap_{(\sigma,l)\in\Sigma} \sqsubset_{(\sigma,l)}, l)$.

In [29, 22] it was shown (by a generalization of Szpilrajn's theorem to so-structures) that $\iota(\epsilon(\mathcal{S})) = \mathcal{S}$ for arbitrary LSOs $\mathcal{S}$.

3.3. Process Semantics of [22] is not Complete

In this subsection we introduce the process semantics for PTI-nets (given by the set $\mathcal{L}AN$ and the mapping α of the semantical framework) as presented in [22]. Moreover, we define the set $\mathcal{R}UN$ and the mappings κ and λ relating process semantics to step semantics. We show, that $\mathcal{L}AN$ as defined in this subsection is not complete, i.e. does not fulfil $\delta(\mathcal{E}X) \subseteq \bigcup_{R\in\mathcal{R}UN} \psi(R)$.

The problem of defining process nets for PTI-nets 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 [22] is based on the usual notion of occurrence nets extended by activator arcs. Occurrence nets are (labeled) acyclic nets with non-branching places (conditions) whose underlying causal relationship between events is described by LSOs. 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 3.8. (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 $B \cup E$.

For $x \in B \cup E$ and $X \subseteq B$ or $X \subseteq E$ we denote ${}^\bullet x = \{y \mid (y,x) \in R\}$, $x^\bullet = \{y \mid (x,y) \in R\}$, ${}^\bullet X = \bigcup_{x\in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x\in X} x^\bullet$. For $x \in E$ and $X \subseteq E$ we denote $x^+ = \{y \mid (y,x) \in Act\}$ and $X^+ = \bigcup_{x\in X} x^+$.

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 7 shows their construction rule. $\kappa(\mathrm{AON})$ captures all (not only local) causal relations between the events (see also Figure 6). Note that Definition 3.8 is a conservative extension of standard occurrence nets by read arcs.

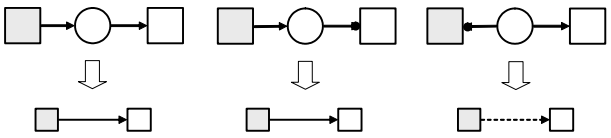

Figure 7. Generation of the orders $\prec_{loc}$ and $\sqsubset_{loc}$ in ao-nets.

There are the notions of weak and strong configurations and slices for ao-nets. We formally introduce only weak configurations and slices, since strong configurations and slices are only used when referring to [22]. A set of events $D \subseteq E$ is called a *weak configuration* of AON, if $e \in D$ and $f(\prec_{loc} \cup \sqsubset_{loc})^+ e$ implies $f \in D$. A *weak slice* of AON is a maximal (w.r.t. set inclusion) set of conditions $S \subseteq B$ which are $R \circ (\prec_{loc} \cup \sqsubset_{loc})^* \circ R$-independent. $\mathrm{WSL}(\mathrm{AON})$ denotes the set of all weak slices. The set $\mathrm{MIN}_{\mathrm{AON}}$ of all conditions without incoming flow arcs (the minimal conditions w.r.t. R) and the set $\mathrm{MAX}_{\mathrm{AON}}$ of all conditions without outgoing flow arcs (the maximal conditions w.r.t. R) are weak slices. For a weak

configuration C, the marking reached from the initial marking after the transitions occurrences from C is represented by the weak slice $S_C = (C^\bullet \cup MIN_{\mathrm{AON}}) \setminus {}^\bullet C$. In [22] it is shown that the set of weak slices S of AON equals the set of weak slices S_C for weak configurations C. In Figure 8, the weak configurations of the ao-net O_1 are $\{e_1\}$, $\{e_1, e_2\}$, $\{e_1, e_3\}$ and $\{e_1, e_2, e_3\}$. Each defines a weak slice. The set $\{e_2\}$ is not a weak configuration, since $e_1(\prec_{loc} \cup \sqsubset_{loc})^+ e_2$.

An activator occurrence net generates a set of labeled step sequences describing its dynamics using the standard a-priori occurrence rule of elementary nets with read arcs [22], where such labeled step sequences are required to contain all events of the occurrence net. Formally, this set is defined by the mapping λ. The set of slices is the union of the sets of weak and strong slices.

Definition 3.9. (The mapping λ)
Let $\mathrm{AON} = (B, E, R, Act, l)$ be an ao-net. A slice S *enables* a *step* of $\prec_{loc}^+$-independent events $\tau \subseteq E$ if ${}^\bullet\tau \cup \tau^+ \subseteq S$. The *occurrence* of τ yields the slice $S' = (S \setminus {}^\bullet\tau) \cup \tau^\bullet$. We write $S \xrightarrow{\tau} S'$.

$\lambda(\mathrm{AON})$ is the set of all labeled finite sequences of steps of $\prec_{loc}^+$-independent events $(\tau_1 \ldots \tau_n, l)$, such that there is a sequence of weak slices $S_0, \ldots, S_n$ with $S_0 = \mathrm{MIN}_{\mathrm{AON}}$, $S_n = \mathrm{MAX}_{\mathrm{AON}}$ and $S_0 \xrightarrow{\tau_1} S_1 \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} S_n$.

Note that a slice S is a weak slice if and only if there is a finite sequence of steps of $\prec_{loc}^+$-independent events $\tau_1 \ldots \tau_n$ with $S \xrightarrow{\tau_1} S_1 \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} \mathrm{MAX}_{\mathrm{AON}}$.

Now we are prepared to define processes of PTI-nets as in [22]. The mentioned artificial conditions in such processes are labeled by the special symbol $\lightning$. 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 in a process are adjacent to a common $\lightning$-condition representing a causal dependency of f and e. That means there exists a $\lightning$-labeled condition b such that $(b, e) \in Act$ and $b \in ({}^\bullet f \cup f^\bullet)$. This is abbreviated by $f \multimap\!\bullet\, e$ (see requirement (Cond6) in Definition 3.10). The axiomatic process definition in [22] is as follows:

Definition 3.10. (Activator process)
An *activator process* (a-process) of NI is an ao-net $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ satisfying:

(Cond1) $l(B) \subseteq P$, $l(E) \subseteq T$ and $l(\widetilde{B}) = \{\lightning\}$.

(Cond2) $\widetilde{B} = \{b \mid \exists e \in E : (b, e) \in Act\}$.

(Cond3) $m_0 = l(\mathrm{MIN}_{\mathrm{AON}} \cap B)$.

(Cond4) For all $e \in E$, ${}^\bullet l(e) = l({}^\bullet e \cap B)$ and $l(e)^\bullet = l(e^\bullet \cap B)$.

(Cond5) For all $b \in \widetilde{B}$, there are unique $g, h \in E$ such that ${}^\bullet b \cup b^\bullet = \{g\}$, $(b, h) \in Act$ and $l(g) \multimap l(h)$.

(Cond6) 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.

(Cond7) For all $e \in E$ and all strong slices S, 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 is denoted by $\mathcal{LAN}_{NI} = \alpha(NI)$. For $\mathrm{AON} \in \alpha(NI)$ the LSO $\kappa(\mathrm{AON})$ is called a *run (associated to* AON*)*. $\mathcal{RUN}_{NI}$ denotes the set of all runs.

The requirements (Cond1), (Cond3), (Cond4) in Definition 3.10 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 λ-labeled conditions (omitting the λ-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 3.10 can also be used to define processes of p/t-nets. The properties (Cond2) and (Cond5) together with the rule (Cond6) – describing when λ-conditions have to be inserted – constitute the structure of the λ-conditions. The requirement (Cond7) 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.

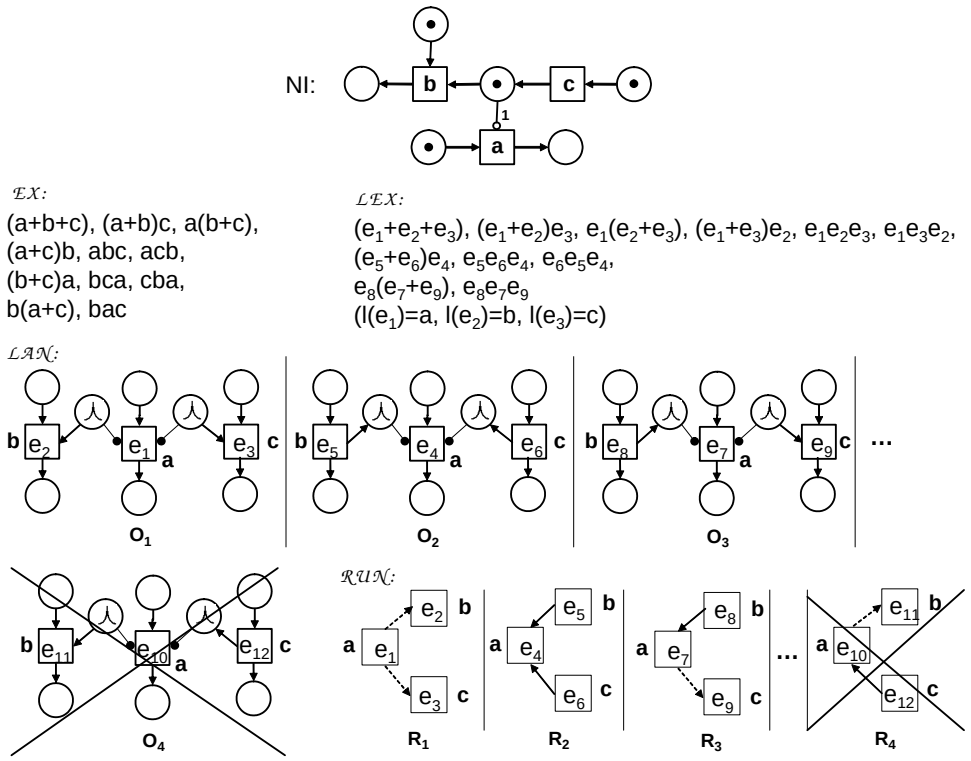

Figure 8. Different semantics of PTI-nets.

Example 3.1. Figure 8 shows an example of the semantics of a PTI-net as occurring in the framework of Figure 4 (b); in this figure, we neglect "prefixes", i.e. always considering processes, runs, enabled causal structures and (labeled) step sequences of maximal length.

The mapping ω defines the step semantics $\mathcal{EX}$. Transition a can fire within a step if and only if there are 0 or 1 tokens in the place connected to a via an inhibitor arc. That means, it is not possible to fire a if c has occurred before, but b has not occurred before. In other words, the only step sequences which are not valid are cab and $c(a+b)$.

The mapping α defines the process semantics $\mathcal{LAN}$ according to Definition 3.10. In Figure 8 only processes with underlying minimal runs are shown. Namely, the processes where b consumes a token produced by c are omitted. Since $b \multimap a$ and $c \multimap a$, each of the process nets in $\mathcal{LAN}$ has two $\curlywedge$-conditions. There are four possible combinations to introduce these two $\curlywedge$-conditions, since both can belong to the pre-set or the post-set of the b- resp. c-labeled event. Only the possibility shown in the occurrence net O_4 is not a process net, since it contradicts property (Cond7): The strong configuration $C = \{e_{12}\}$ yields a slice S_C which enables e_{10} but satisfies $l(S_C \cap B) \not\leq {}^{-}a$.

The mapping κ assigns to each process net in $\mathcal{LAN}$ a run in $\mathcal{RUN}$ through omitting the places in the process net and keeping the causal relations between the events (transition occurrences) according to Figure 7. It holds $\kappa(O_i) = R_i$ for $i = 1, 2, 3, 4$.

The mapping λ assigns labeled step sequences of the set $\mathcal{LEX}$ to a process net in $\mathcal{LAN}$ according to Definition 3.9. The mapping ϵ assigns the set of all labeled step sequences in $\mathcal{LEX}$ to an LSO in $\mathcal{RUN}$, which add causality to the LSO (events in one step are considered to be synchronous). The event names indicate which labeled step sequences are generated by which process resp. run.

The mapping ι assigns an LSO to a set of labeled step sequences Σ (with set of events V and labeling function l) as shown in Definition 3.7. There holds $\iota(\epsilon(R_i)) = R_i$ for $i = 1, 2, 3$.

The mapping ϕ simply abstracts from the individuality of events in a labeled step sequence in $\mathcal{LEX}$, for example $\phi((e_1 + e_2)e_3) = (a + b)c$.

Finally, π is an algorithm for the construction of a set of process nets from a step sequence. Namely, for a step sequence all process nets generating the given step sequence are computed. It holds, for example, $\pi((a + b + c)) = \{O_1\}$, $\pi((b + c)a) = \{O_2\}$ and $\pi(bac) = \{O_3\}$.

In [22], it is proven that the a-process definition given in Definition 3.10 fulfills all properties of the semantical framework of Figure 4, left part. That means the process semantics for PTI-nets given in [22] is in particular sound and weakly complete. But it is not complete as shown in the next theorem.

Theorem 3.1. The process semantics defined in Definition 3.10 is not complete.

Proof:
Figure 9 gives an example of a PTI-net NI and an LSO $\mathcal{S}$ satisfying:

(i) $\mathcal{S}$ is enabled w.r.t. NI ($\phi(\epsilon(\mathcal{S})) \subseteq \mathcal{EX}$): This is an easy computation using the step semantics $\mathcal{EX}$ of NI shown in Figure 8 (remember that only cab and $c(a + b)$ are not in $\mathcal{EX}$).

(ii) $\mathcal{S}$ is not an extension of a run ($\mathcal{S} \notin \bigcup_{R \in \mathcal{RUN}} \psi(R)$): This can easily be verified by checking all minimal runs of NI (shown in Figure 8). The reason that the enabled LSO $\mathcal{S}$ is not extension of a run follows from the considerations in Example 3.1 on the construction of $\curlywedge$-conditions. There are two problems, each being sufficient to produce a contradiction. First, according to (Cond6), it is necessary to introduce two $\curlywedge$-conditions connecting to the b- and a-labeled events resp. the c- and a-labeled events. Therefore, the b- and a-labeled event resp. the c- and a-labeled event labeled event cannot be concurrent in some process net. Second, the only possibility to establish a "not later than"-relation between b and c is through a $\curlywedge$-condition connecting the b- and c-labeled event. Such a condition cannot be introduced according to (Cond5), since $b \not\multimap c$.

Altogether, $\mathcal{S} \in \mathcal{ELCS} \setminus \bigcup_{R \in \mathcal{RUN}} \psi(R)$. □

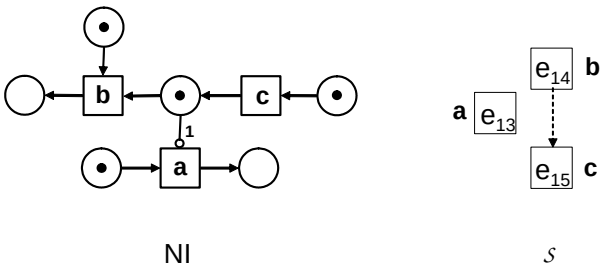

Figure 9. Example contradicting the completeness of the process definition from [22].

3.4. Complete Process Semantics

In this subsection we propose a new definition of a-priori process nets for PTI-nets and show that the resulting process semantics is complete. Formally, we change the definition of α and thus $\mathcal{LAN}$ in the semantical framework (while this also changes the set $\mathcal{RUN}$ all other notations remain the same). The new definition is based on a modification of the properties (Cond5), (Cond6) and (Cond7) of Definition 3.10. Briefly, following to the arguments in the proof of Theorem 3.1, (Cond5') allows for additional $\curlywedge$-conditions and according to (Cond6') adding of $\curlywedge$-conditions is more flexible. (Cond7') covers a technical difficulty after modifying (Cond5) and (Cond6).

Definition 3.11. (Complete activator process)
A *complete activator process* (ca-process)[4] of NI is an ao-net $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ satisfying:

(Cond1) $l(B) \subseteq P$, $l(E) \subseteq T$ and $l(\widetilde{B}) = \{\curlywedge\}$.

(Cond2) $\widetilde{B} = \{b \mid \exists e \in E : \ (b,e) \in Act\}$.

(Cond3) $m_0 = l(\mathrm{MIN}_{\mathrm{AON}} \cap B)$.

(Cond4) For all $e \in E$, ${}^\bullet l(e) = l({}^\bullet e \cap B)$ and $l(e)^\bullet = l(e^\bullet \cap B)$.

(Cond5') For all $b \in \widetilde{B}$, there are unique $g, h \in E$ such that ${}^\bullet b \cup b^\bullet = \{g\}$, $(b,h) \in Act$ and $l(g) \multimap l(h)$ or $b^\bullet = \{g\}$, $(b,h) \in Act$ and additionally ${}^\bullet l(h) \cap l(g)^\bullet \cap {}^- z \neq \emptyset$ for some $z \in T$.

(Cond6') For all $e, f \in E$, if $f \multimap e$ then there is exactly one $c \in \widetilde{B}$ such that $f \multimap e$ through c.

(Cond7') For all $e \in E$ and $S \in \mathrm{WSL(AON)}$, 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 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).

Example 3.2. Figure 10 illustrates the complete process semantics given by ca-processes of the marked p/t-net shown in Figure 8. The ao-net O_5 is a ca-process (given by α') but no a-process (given by α) as explained in the context of Figure 8.

[4]Note that in [22] the term *ca-process* is used for processes of a restricted class of PTI-nets, namely p/t-nets with *complemented* inhibitor places.

The $\curlywedge$-condition establishing $e_{15} \multimap e_{14}$ can be introduced according to (Cond5'), since ${}^\bullet b \cap c^\bullet \cap {}^- a \neq \emptyset$.

Observe that the slice MIN_{AON} enables the step e_{15}. Since $\{e_{15}\}$ is a strong but not weak configuration, the occurrence of e_{15} leads to the strong but not weak slice $S_{\{e_{15}\}}$. $S_{\{e_{15}\}}$ enables e_{13}, but $l(S_{\{e_{15}\}} \cap B)$ does not obey the inhibitor restrictions for a (i.e. does not enable a). That means, O_5 generates a labeled step sequence (namely $e_{15}e_{13}$) which defines a step sequence via ϕ (namely ac) which is not enabled, but this sequence cannot be completed by e_{14}. Such situation is possible, because an inhibitor restriction of a transition occurrence (the event e_{13} in the example) need not longer be directly reflected by relating $\curlywedge$-conditions to such a transition occurrence.

Nevertheless, property (Cond7') is satisfied, since there are considered only weak slices (in (Cond7) strong slices are considered). Starting in the respective marking of a weak slice, all events of AON not occurred yet can still be executed. With (Cond7'), we only model such behavior by a process net AON, in which every event of AON actually occurs, i.e. the labeled step sequence $e_{15}e_{13}$ is not regarded.

The LSO K_5 is the run associated to O_5. Note that K_5 equals $\mathcal{S}$ from Figure 9. This illustrates that the process semantics given by Definition 3.11 is complete for the considered example PTI-net NI.

Note that the requirements (Cond1), (Cond3), (Cond4) of Definition 3.10 are preserved in Definition 3.11, 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 $\mathrm{Und}(NI)$.

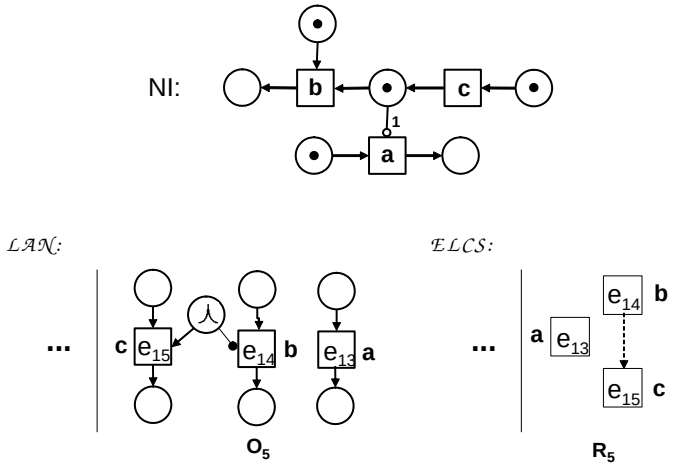

Figure 10. Completeness of the ca-process definition.

The main ideas of the modifications (in Definition 3.11 in contrast to Definition 3.10) can be deduced from the proof of Theorem 3.1. The first observation in this proof is that, according to (Cond6), each $\multimap$-relation must be reflected by a $\curlywedge$-condition. Since through this requirement possibly too much causality is added, it is weakened in (Cond6') to the possibility of introducing a (unique) $\curlywedge$-condition. The second observation was that there are situations additional to (Cond5), in which the introduction of $\curlywedge$-conditions should be possible. Considering the example shown in Figure 9, this is the case, if a transition, testing some place via an inhibitor arc, occurs concurrently to transitions consuming and producing tokens in this place. Then these transition occurrences must eventually be ordered via a $\curlywedge$-condition. Such

λ-conditions are intended to ensure that tokens in the considered place are consumed not later than produced in order to restrict the maximal number of tokens in the place according to the inhibitor weight. This is formally reflected in (Cond5').

These modifications can lead to situations, where a strong slice enables an event, but the marking corresponding to this slice does not obey the corresponding inhibitor restrictions (violating (Cond7), see Example 3.2). This problem is resolved by considering only weak slices in (Cond7') instead of strong slices. Since each weak slice is also a strong slice, (Cond7') weakens (Cond7). As already mentioned, from weak slices always the slice MAX_{AON} can be reached. That means, starting in the respective marking of a weak slice, all events of AON not occurred yet can still be executed. This is not the case for strong slices. With (Cond7'), we only model such behavior by a process net AON, in which every event of AON actually occurs. Altogether, a ca-process may generate a labeled step sequence which does not define an enabled step sequence via ϕ, but in such sequences never all events of the process can occur (Figure 10). According to the definition of λ such sequences do not belong to $\mathcal{LEX}$.

Altogether (Cond6') does not any more require certain λ-conditions, (Cond5') allows λ-conditions additional to (Cond5), and (Cond7') is only formulated for weak slices (instead of strong slices). Thus (Cond5'), (Cond6') and (Cond7') are weaker requirements than (Cond5), (Cond6) and (Cond7) Consequently every a-process is a ca-process, i.e. $\alpha(NI) \subseteq \alpha'(NI)$.

We show the main result of the paper, stating that actually the ca-process definition in general fulfills the aim of completeness. As a preparation we need the notion of prefixes of LSOs and a specific relation between prefixes of an LSO $\mathcal{S}$ and prefixes of labeled step sequences in $\epsilon(\mathcal{S})$. Prefixes are defined by subsets of nodes which are downward closed w.r.t. the $\sqsubset$-relation:

Definition 3.12. (Prefix)
Let $\mathcal{S} = (V, \prec, \sqsubset, l)$ be an LSO and let $V' \subseteq V$ be a set of events such that $u' \in V'$, $u \sqsubset u' \Longrightarrow u \in V'$. Then V' defines a *prefix* $\mathcal{S}'$ w.r.t. $\mathcal{S}$ by $\mathcal{S}' = (V', \prec\ |_{V' \times V'}, \sqsubset\ |_{V' \times V'}, l|_{V'})$. A *prefix* $\mathcal{S}'$ *enabling* $u \in V \setminus V'$ is a prefix w.r.t. $\mathcal{S}$ satisfying $(v \prec u \Longrightarrow v \in V')$.

Lemma 3.1. Let V' define a prefix $\mathcal{S}'$ (enabling $u \in V$) w.r.t. $\mathcal{S}$. Then there exists $(\tau_1 \ldots \tau_n, l) \in \epsilon(\mathcal{S})$ such that $V' = \bigcup_{i=1}^{k} \tau_i$ (and $u \in \tau_{k+1}$) for some k.

Proof:
$\tau_1 \ldots \tau_n$ can be constructed by $\tau_1 = \{v \in V' \mid \forall v' \in V' : v' \not\prec v\}$, $\tau_2 = \{v \in V' \setminus \tau_1 \mid \forall v' \in V' \setminus \tau_1 : v' \not\prec v\}$ and so on. In general, 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 $V' \setminus (\bigcup_{j=1}^{i-1} \tau_j)$, until $V' \setminus (\bigcup_{j=1}^{k} \tau_j) = \emptyset$ for some k.

Then we continue with the same procedure on $V \setminus V' = V \setminus (\bigcup_{j=1}^{k} \tau_j)$, i.e. $\tau_{k+1} = \{v \in V \setminus (\bigcup_{j=1}^{k} \tau_j) \mid \forall v' \in V \setminus (\bigcup_{j=1}^{k} \tau_j) : v' \not\prec v\}$ and so on. By construction $V' = \bigcup_{i=1}^{k} \tau_i$ and $u \in \tau_{k+1}$. □

Theorem 3.2. For every enabled LSO $\mathcal{S} = (E, \prec, \sqsubset, l)$ of a PTI-net NI there exists a ca-process $\mathrm{AON} \in \alpha'(NI)$ such that $\mathcal{S}$ is an extension of the run $\kappa(\mathrm{AON})$.

Proof:
We construct a ca-process AON, which satisfies the statement. First we define AON. Then we prove that AON fulfills all formulated requirements. For illustration of the proof, we provide an example in Figure 11 and Figure 12.

"Construction of AON**":** Since the inhibitor relation I of NI restricts the behaviour of the underlying p/t-net $\mathrm{Und}(NI)$, it a fortiori holds that $\mathcal{S}$ is enabled w.r.t. $\mathrm{Und}(NI)$. Note here that in a p/t-net transitions that can be executed as one step can also be executed in arbitrary order. Furthermore, every $\sqsubset \setminus \prec$-relation between two events in the so-structure $\mathcal{S}$ allows the occurrence of these events in one step. Therefore, the enabledness w.r.t. the p/t-net $\mathrm{Und}(NI)$ is preserved omitting the $\sqsubset$-relation. That means, the LPO $\mathrm{lpo}_{\mathcal{S}} = (E, \prec, l)$ underlying $\mathcal{S}$ is enabled w.r.t. $\mathrm{Und}(NI)$. Since the enabledness notion for so-structures applied to LPOs coincides with the usual enabledness notion for LPOs, we can use the usual notion here.

Now we can apply the LPO-analogon to this theorem proved in [20]: Since $\mathrm{lpo}_{\mathcal{S}}$ is enabled w.r.t. $\mathrm{Und}(NI)$, there exists a process $\mathrm{UAON} = (B, E, R', l')$ of $\mathrm{Und}(NI)$ fulfilling that $\mathrm{lpo}_{\mathcal{S}}$ sequentializes the run $\kappa(\mathrm{UAON})$ (for the definition of the p/t-net process UAON and the mapping κ we can use Definition 3.10 as well as the usual process definition for p/t-nets in [20], because they coincide for p/t-nets). Note that UAON is not unique here, but this causes no troubles.

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 the properties (Cond 5)' and (Cond 6)' and do not produce causal dependencies contradicting $\mathcal{S}$. We claim that this ao-net $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ is the desired ca-process.

Formally, first for each pair of events $f, e \in E$ with $f \prec e$ we insert a $\curlywedge$-condition into UAON generating this causality according to Figure 7, if this is allowed according to (Cond 5)' and (Cond 6)' in Definition 3.11. Analogously, for each pair of events $f, e \in E$ with $f \sqsubset e$ we insert a $\curlywedge$-condition to UAON generating this causality according to Figure 7, if this is allowed according to (Cond 5)' and (Cond 6)' in Definition 3.11. Note that the order of adding $\curlywedge$-conditions is irrelevant, since adding a possible $\curlywedge$-condition does not prohibit another possible $\curlywedge$-condition in this construction. To verify this, (Cond 6)' has to be regarded (all other requirements for adding $\curlywedge$-conditions are independent from other $\curlywedge$-conditions): The only possibility to get two $\curlywedge$-conditions generating $f \multimap e$ is $f \prec e$ and $e \sqsubset f$, but this is not possible in an so-structure. Next we prove that AON fulfills the desired properties.

"Obvious properties of AON**":** Most requirements formulated in the statement of the theorem can easily be observed. By construction it is clear that

- AON is an ao-net: We have to verify that $\mathcal{S}(\mathrm{AON})$ is $\diamondsuit$-acyclic, which is obvious, since $\mathcal{S}$ is an so-structure and consequently $\prec_{\mathcal{S}(\mathrm{AON})^{\diamondsuit}} \subseteq \prec$ is irreflexive.
- $\mathcal{S}$ extends $\kappa(\mathrm{AON})$.
- AON fulfills the conditions (Cond 1) - (Cond 4), (Cond 5)' and (Cond 6)' of Definition 3.11.

"(Cond 7)' for AON**":** It only remains to show that AON meets condition (Cond 7)' of Definition 3.11: Given $e \in E$ and $S \in \mathrm{WSL(AON)}$ with ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b, e) \in Act\} \subseteq S$ we have to show $l(S \cap B) \leq {}^- l(e)$. In the following we denote $MAR(C) = l(S_C \cap B)$ for a configuration C of AON. According to [22] (see section 3) there exists a weak configuration C of AON with $S = S_C$. Therefore, if we show that $l(e)$ is enabled (w.r.t. the inhibitor relation) in the PTI-net NI after the occurrence of the transitions corresponding to events in C, the theorem is proven. The proof for this proceeds in steps and in each step we consider new sets of events C_i, $i \in \{1, 2, 3\}$, and prove intermediate properties. We start by showing that a set of events denoted by C_{pre} fulfills $\mathrm{MAR}(C_{pre}) \leq {}^- l(e)$. Then this set will be stepwise modified using the sets C_i and we will show that the above property is preserved for the

modified sets. The last modified set of events will equal C proving (Cond 7)', i.e. $\mathrm{MAR}(C) \leq {}^{-}l(e)$. More precisely, we will proceed as follows:

- We define C_{pre} and show that $l(e)$ is enabled in the PTI-net NI after the occurrence of the transitions corresponding to events in C_{pre}, i.e. $\mathrm{MAR}(C_{pre}) \leq {}^{-}l(e)$.
- We define C_1, C_2 and C_3 and show $C = ((C_{pre} \setminus C_1) \setminus C_2) \cup C_3$.
- We show that $C_{pre} \setminus C_1$ fulfills $\mathrm{MAR}(C_{pre} \setminus C_1) \leq {}^{-}l(e)$.
- We show that $(C_{pre} \setminus C_1) \setminus C_2$ fulfills $\mathrm{MAR}((C_{pre} \setminus C_1) \setminus C_2) \leq {}^{-}l(e)$.
- We show that $C = ((C_{pre} \setminus C_1) \setminus C_2) \cup C_3$ fulfills $\mathrm{MAR}(((C_{pre} \setminus C_1) \setminus C_2) \cup C_3) \leq {}^{-}l(e)$.

"Inequality $\mathrm{MAR}(C_{pre}) \leq {}^{-}l(e)$": We denote $C_3 = \{c \in C \mid e \sqsubset c\}$ (these events cannot belong to an enabling prefix of e in $\mathcal{S}$), $C_0 = C \setminus C_3$ (these events will belong to C_{pre}) and $C_{pre} = C_0 \cup \{c \in E \mid \exists c' \in C_0, c \sqsubset c'\} \cup \{e' \in E \mid e' \prec e\}$. C_{pre} defines an enabling prefix of e in $\mathcal{S}$ which contains a maximal number of events from the set C and is minimal with this property. By Lemma 3.1 there is $(\tau_1 \ldots \tau_n, l) \in \epsilon(\mathcal{S})$ such that $C_{pre} = \bigcup_{i=1}^{k} \tau_i$ and $e \in \tau_{k+1}$ for some k. Because $\mathcal{S}$ is enabled, $l(\tau_1) \ldots l(\tau_n)$ represents an enabled synchronous step sequence of NI. This implies that $l(e)$ is enabled in the marking given by the slice $S_{C_{pre}}$, i.e. $\mathrm{MAR}(C_{pre}) \leq {}^{-}l(e)$.

"Equality $C = ((C_{pre} \setminus C_1) \setminus C_2) \cup C_3$": There are the following events in $C_{pre} \setminus C$: $c \in C_1 = \{c \in C_{pre} \setminus C_0 \mid c \not\prec e\}$ and $c \in C_2 = \{c \in C_{pre} \setminus C_0 \mid c \prec e\}$, i.e. we consider the partition $C_{pre} = C_0 \uplus C_1 \uplus C_2$. Let C_3 equal the set of events $C \setminus C_{pre}$, consequently $C = ((C_{pre} \setminus C_1) \setminus C_2) \cup C_3$. First we consider the most complicated case of the three announced modifications of C_{pre}. This first inequality is proven by means of several preliminary results.

"Inequality $\mathrm{MAR}(C_{pre} \setminus C_1) \leq {}^{-}l(e)$": Recalling the definition of $C_{pre} = C_0 \cup \{c \in E \mid \exists c' \in C_0, c \sqsubset c'\} \cup \{e' \in E \mid e' \prec e\}$, an event $c \in C_1 \cap C_{pre}$ is obviously neither in the first set C_0 nor in the third set $\{e' \in E \mid e' \prec e\}$ of the respective union. Thus it is in the second one $\{c \in E \mid \exists c' \in C_0, c \sqsubset c'\}$ meaning that there exists $c' \in C_0$ with $c \sqsubset c'$ (and $c' \not\prec e$ because of (C4) in Definition 3.3). We first show that if we omit all events $c' \in \{c' \in C_0 \mid \exists c'' \in C_1 : c'' \sqsubset c'\}$ additionally to C_1 from C_{pre} we get again a prefix enabling e. We define $C_{cancel} = C_1 \cup \{c' \in C_0 \mid \exists c'' \in C_1, c'' \sqsubset c'\}$ and prove:

"$C_{pre} \setminus C_{cancel}$ defines a prefix of $\mathcal{S}$ enabling e": We show first that $C_{pre} \setminus C_{cancel}$ defines a prefix of $\mathcal{S}$, i.e. for $c' \in E, c \in C_{pre} \setminus C_{cancel}, c' \sqsubset c$ there holds $c' \in C_{pre} \setminus C_{cancel}$. Since C_{pre} defines a prefix, we know that $c' \in C_{pre}$. Assume now that $c' \in C_{cancel}$, then according to the definition of C_{cancel} there are two possibilities:

(i) $c' \in C_1$: In this case either $c \in C_0$, which implies that $c \in \{c' \in C_0 \mid \exists c'' \in C_1, c'' \subseteq c'\} \subseteq C_{cancel}$, or $c \in C_{pre} \setminus C_0$, which implies that $c \in C_1 \subseteq C_{cancel}$, since $c \not\prec e$. This contradicts $c \notin C_{cancel}$.

(ii) $c' \in C_0$ such that $\exists c'' \in C_1 : c'' \sqsubset c'$: In this case $c'' \sqsubset c' \sqsubset c$, i.e. $c'' \sqsubset c$ or $c'' = c$ (by (C3)). The latter implies $c \in C_1 \subseteq C_{cancel}$ contradicting $c \notin C_{cancel}$. The case $c'' \sqsubset c$ leads to (i) (considering c'' instead of c'), i.e. a contradiction.

This gives $c' \notin C_{cancel}$ and thus $C_{pre} \setminus C_{cancel}$ defines a prefix of $\mathcal{S}$.

Furthermore $C_{pre} \setminus C_{cancel}$ even defines a prefix enabling e, i.e. $\{e' \in E \mid e' \prec e\} \subseteq C_{pre} \setminus C_{cancel}$. This can be shown as follows: First, by definition C_{pre} includes $\{e' \in E \mid e' \prec e\}$. Second, inn order to show $C_{cancel} \cap \{e' \in E \mid e' \prec e\} = \emptyset$, we have to verify $C_1 \cap \{e' \in E \mid e' \prec e\} = \emptyset$ and $\{c' \in C_0 \mid \exists c'' \in C_1, c'' \sqsubset c'\} \cap \{e' \in E \mid e' \prec e\} = \emptyset$. The statement $C_1 \cap \{e' \in E \mid e' \prec e\} = \emptyset$ directly follows from the definition of $C_1 = \{c \in C_{pre} \setminus C_0 \mid c \not\prec e\}$. Assuming that $\{c' \in C_0 \mid \exists c'' \in C_1, c'' \sqsubset c'\} \cap \{e' \in E \mid e' \prec e\} \neq \emptyset$, there is an event $c' \prec e$ and an event $c'' \in C_1, c'' \sqsubset c'$. Then by (C4) we have $c'' \prec e$, which contradicts the definition of $C_1 = \{c \in C_{pre} \setminus C_0 \mid c \not\prec e\}$.

"Preliminary inequality $\mathrm{MAR}(C_{pre} \setminus C_{cancel}) \leq {}^{-}l(e)$": Of course $C_{pre} \setminus C_{cancel}$ also defines a prefix of the prefix defined by C_{pre}. Consequently, without loss of generality, by Lemma 3.1 we can assume that $C_{pre} \setminus C_{cancel} = \bigcup_{j=1}^{i} \tau_j$ for some $i \leq k$ (the above requirements for $(\tau_1 \dots \tau_n, l)$ allow choosing such $(\tau_1 \dots \tau_n, l)$) and $l(e)$ is enabled after the occurrence of $l(\tau_1) \dots l(\tau_i)$ (as well as after $l(\tau_1) \dots l(\tau_k)$ as explained above). As for C_{pre}, this implies that $l(e)$ is enabled in the marking given by the slice $S_{C_{pre} \setminus C_{cancel}}$, i.e. $\mathrm{MAR}(C_{pre} \setminus C_{cancel}) \leq {}^{-}l(e)$.

"$C_{pre} \setminus C_1$ is a weak configuration": Since we are interested in $C_{pre} \setminus C_1$, we first verify that $C_{pre} \setminus C_1$ is a weak configuration of AON (it need not define a prefix of $\mathcal{S}$). Since C_{pre} defines a prefix of $\mathcal{S}$ and $\mathcal{S}$ extends $\kappa(\mathrm{AON})$, it is a weak configuration of AON. Assuming that $C_{pre} \setminus C_1$ is no weak configuration, there is $c' \in C_1$, $c \in C_{pre} \setminus C_1$, such that the relation $c' \prec_{loc} c$ or $c' \sqsubset_{loc} c$ is generated by some condition in AON (according to Figure 7). We distinguish two possibilities: Either $c \in C$, then by definition of weak configurations, we have $c' \in C$ (C is a weak configuration). This is a contradiction, because $C \cap C_1 = \emptyset$. Or $c \notin C$, then $c \in C_2$, i.e. $c \prec e$. By (C4) we have $c' \prec e$ contradicting the definition of C_1.

"Inhibitor constraint $\mathrm{MAR}(C_{pre} \setminus C_1) \leq {}^{-}l(e)$": Now we check whether the inhibitor constraints of $l(e)$ are respected in the marking resulting from the execution of the events in $C_{pre} \setminus C_1$. Assume the opposite: Let $p \in P$ with $\mathrm{MAR}(C_{pre} \setminus C_1)(p) > {}^{-}l(e)(p)$. We know that $\mathrm{MAR}(C_{pre})(p) \leq {}^{-}l(e)(p)$ and $\mathrm{MAR}(C_{pre} \setminus C_{cancel})(p) \leq {}^{-}l(e)(p)$. There must be a transition $l(c)$ corresponding to an event $c \in C_1$, that consumes tokens from p, since $\mathrm{MAR}(C_{pre} \setminus C_1)(p) > {}^{-}l(e)(p) \geqslant \mathrm{MAR}(C_{pre})(p)$. Similarly, there must be a transition $l(c)$ corresponding to an event $c \in C_{cancel} \setminus C_1$, that produces tokens in p, since $\mathrm{MAR}(C_{pre} \setminus C_{cancel})(p) \leqslant {}^{-}l(e)(p) < \mathrm{MAR}(C_{pre} \setminus C_1)(p)$. Thus the sets $C_1^p = \{c \in C_1 \mid W(l(c), p) < W(p, l(c))\}$ and $C_c^p = \{c \in C_{cancel} \setminus C_1 \mid W(l(c), p) > W(p, l(c))\}$ are not empty. Now we distinguish two cases:

(i) $\exists c' \in C_1^p, \exists c \in C_c^p : c' \sqsubset c$: In this case, by construction there exist a $\curlywedge$-condition in AON with a read arc to c' and a flow arc to c, because this $\curlywedge$-condition matches the requirements of (Cond 5)' in Definition 3.11 and reflects the $c' \sqsubset c$ relation. Since $c \in C_{cancel} \setminus C_1 \subseteq C_0 \subseteq C$ and $c' \notin C$ ($c' \in C_1$ and $C_1 \cap C = \emptyset$), this is a contradiction to the definition of weak configurations.

(ii) $\forall c' \in C_1^p, \forall c \in C_c^p : c' \not\sqsubset c$: In this case we claim that $X = (C_{pre} \setminus C_{cancel}) \cup C_c^p \cup \{c' \in C_{cancel} \mid \exists c \in C_c^p, c' \sqsubset c\}$ is an enabling prefix of e w.r.t. $\mathcal{S}$ with $\mathrm{MAR}(X)(p) > {}^{-}l(e)(p)$ – what is a contradiction to the enabledness of $\mathcal{S}$.

Clearly $\{e' \in E \mid e' \prec e\} \subseteq X$ because $C_{pre} \setminus C_{cancel} \subseteq X$. Moreover, X is $\sqsubset$-downward closed: For an event $c' \sqsubset c$, $c \in C_c^p$ (which is in $\sqsubset$-relation to an event $c \in C_c^p$) there are two cases.

Either $c' \in C_{cancel}$, then $c' \in \{c' \in C_{cancel} \mid \exists c \in C_c^p, c' \sqsubset c\} \subseteq X$;

Or $c' \notin C_{cancel}$, then $c' \in C_{pre} \setminus C_{cancel} \subseteq X$, since by definition of C_c^p and C_{cancel}, we have $c \in C_0 \subseteq C_{pre}$ and C_{pre} is a prefix of $\mathcal{S}$ (which implies $c' \in C_{pre}$).

Thus, X is an enabling prefix of e in $\mathcal{S}$. Since $X \cap C_1^p = \emptyset$ we finally compute $\mathrm{MAR}(X)(p) \geq \mathrm{MAR}((X \setminus C_1) \cup (C_{cancel} \setminus C_1))(p) = \mathrm{MAR}(C_{pre} \setminus C_1)(p) > {}^{-}l(e)(p)$. Here, the first $\geq$-relation in this inequation follows, since X does not contain any events of C_1^p, and therefore by erasing C_1 from X no events consuming tokens in p are erased; similarly, since X already contains all C_c^p-events, by adding $C_{cancel} \setminus C_1$ no events producing tokens in p are added.

Altogether the assumption has lead to a contradiction and thus it holds $\mathrm{MAR}(C_{pre} \setminus C_1) \leq {}^{-}l(e)$. Now we prove the remaining two inequalities:

"Inequality $\mathrm{MAR}((C_{pre} \setminus C_1) \setminus C_2) \leq {}^{-}l(e)$": We again first show that $(C_{pre} \setminus C_1) \setminus C_2 = C_0$ is a weak configuration. Since $C_{pre} \setminus C_1$ is a weak configuration, if we assume that C_0 is not, then there is $c' \in C_2, c \in C_0$ such that the relation $c' \prec_{loc} c$ or $c' \sqsubset_{loc} c$ is generated by some condition in AON (according to Figure 7). Since $c \in C$, by definition of weak configurations, we have $c' \in C$ (C is a weak configuration). This is a contradiction, because $C \cap C_2 = \emptyset$.

Let $c \in C_2$. Since $c \notin C$ there cannot exist a $\curlywedge$-condition with a read arc to e and an ingoing flow arc from c (otherwise this $\curlywedge$-condition is in the weak slice S according to the preliminaries of (Cond 7)' of Definition 3.11 and therefore c is in the weak configuration C). This implies $l(c) \not\multimap l(e)$ (otherwise a $\curlywedge$-condition as described above is present by construction). Consequently, the transitions $l(c)$, $c \in C_2$ do not produce or consume tokens in places with ${}^{-}l(e) < \infty$ and consequently can be omitted from the inequation $\mathrm{MAR}(C_{pre} \setminus C_1) \leq {}^{-}l(e)$ preserving the $\leq$-relation: $\mathrm{MAR}((C_{pre} \setminus C_1) \setminus C_2) \leq {}^{-}l(e)$.

"Inequality $\mathrm{MAR}(((C_{pre} \setminus C_1) \setminus C_2) \cup C_3) \leq {}^{-}l(e)$": By construction $((C_{pre} \setminus C_1) \setminus C_2) \cup C_3 = C$ is a weak configuration.

Let $c \in C_3$. Then $e \sqsubset c$ but since $c \in C$ and $e \notin C$ there is no $\curlywedge$-condition having a read arc to e and a flow arc to c. Thus $l(c) \not\multimap l(e)$, otherwise such a $\curlywedge$-condition exists by construction of AON. Consequently, as in the case of C_2, C_3 has no relevance for the marking of places with ${}^{-}l(e) < \infty$ and can therefore be added in the inequation as follows: $\mathrm{MAR}(((C_{pre} \setminus C_1) \setminus C_2) \cup C_3) \leq {}^{-}l(e)$. □

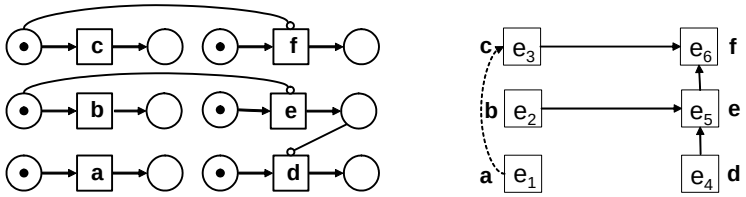

Figure 11. Left: Example PTI-net NI (place names are neglected). Right: Example LSO $\mathcal{S}$ enabled w.r.t. NI.

Example 3.3. The following example illustrates the proof of Theorem 3.2. Considering the net NI on the left of Figure 11 and the enabled LSO $\mathcal{S}$ shown on the right of Figure 11, the ca-process AON constructed according to this proof is depicted in Figure 12.

In the unique maximal process net UAON of the underlying p/t-net $\mathrm{Und}(NI)$, all transition occurrences are concurrent. For each of the "earlier than"- and "not later than"-relations between events in $\mathcal{S}$, a $\curlywedge$-condition is introduced, if this is possible according to (Cond 5)' and (Cond 6)'. For example, a $\curlywedge$-condition establishes e_3 "earlier than" e_6. This $\curlywedge$-condition can be added since $c \multimap f$. Note that it is not possible to add a $\curlywedge$-condition producing e_1 "not later than" e_3 since the transitions a and c are not related via inhibitor arcs.

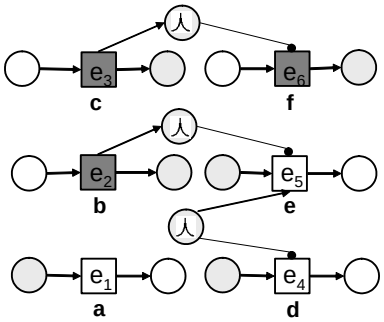

Figure 12. Ca-process AON constructed according to the proof of Theorem 3.2 from the LSO $\mathcal{S}$ shown in Figure 11 (labels of the conditions are neglected).

To illustrate the main part of the proof, the verification of condition (Cond 7)', we consider the weak slice S given by the grey conditions and the event e_5 labeled by e. The respective weak configuration $C = \{e_2, e_3, e_6\}$ comprises the grey events. For this example, the sets of events considered in the proof are $C_0 = \{e_2, e_3\}$, $C_1 = \{e_1\}$, $C_2 = \{e_4\}$, $C_{pre} = C_0 \uplus C_1 \uplus C_2 = \{e_1, e_2, e_3, e_4\}$, $C_3 = \{e_6\}$ and $C_{cancel} = \{e_1, e_3\}$.

3.5. The Whole Framework

The last theorem showed that the newly developed process semantics for PTI-nets defined in Definition 3.11 is complete. In the following we briefly explain that the other properties of the presented semantical framework are still fulfilled by the new process definition (see Figure 4). Therefore, this definition is an adequate generalization of Definition 3.10.

3.5.1. 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 Szpilrajn's theorem to so-structures as described in Subsection 3.2 (note that in this context nothing is changed with respect to [22]).

3.5.2. Weak Completeness

Any execution $\sigma \in \mathcal{EX} = \omega(NI)$ of a PTI-net NI is generated from a ca-process, i.e. there exists a 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 holds for ca-processes, because the aim of completeness is a generalization of the weak completeness property as already mentioned.

3.5.3. Consistency of Runs and Processes

Consistency of runs and processes requires that processes and runs generate the same labeled 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 7 are correct). This relation holds

for ca-processes, because in Proposition 5.19 of [22] $\epsilon(\kappa(\mathrm{AON})) = \lambda(\mathrm{AON})$ was shown for arbitrary ao-nets AON. Note here that the construction rules of the involved mappings λ, κ and ϵ have not changed with respect to [22], only the process definition constituting the starting point of this relation is changed.

3.5.4. Soundness

According to Proposition 5.19 of [22] (Subsection 3.5.1), to show $\phi(\epsilon(\kappa(\mathrm{AON}))) \subseteq \omega(NI)$ for all $\mathrm{AON} \in \alpha'(NI)$, it suffices to prove that $\phi(\lambda(\mathrm{AON})) \subseteq \omega(NI)$ for $\mathrm{AON} \in \alpha'(NI)$. Recalling the definition of $\lambda(\mathrm{AON})$ in Definition 3.9, this relation is clear by (Cond3), (Cond4) and (Cond7') of Definition 3.11: (Cond3) and (Cond4) ensure that $\sigma \in \phi(\lambda(\mathrm{AON}))$ is enabled in $Und(NI)$ and (Cond7') guarantees that σ respects the inhibitor constraints of NI. This shows that every run of NI is enabled w.r.t. NI.

3.5.5. Construction of Processes from Step Sequences

There is no obvious way to generalize the constructive definition of π from [22] (a-processes) to ca-processes, because especially the new requirement (Cond6') of Definition 3.11 is problematic: Now it is no more mandatory but optional to introduce $\curlywedge$-conditions between certain transitions (the transition candidates can be identified with (Cond5')) and one has to check whether (Cond7') holds ((Cond7) holds by construction). There is the following constructive process definition that is based directly on the axiomatic definition. Given an enabled step sequence $\sigma = U_1 \dots U_n$ of NI, a ca-processes $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ of NI can be generated from σ, such that $\sigma \in \phi(\lambda(\mathrm{AON}))$, as follows:

(I) Construct a usual p/t-net process $\mathrm{UAON} = (B, E, R, l)$ of $Und(NI)$ (based on an occurrence net) starting from $\sigma = U_1 \dots U_n$ (e.g. as shown in Definition 6.2 in [22]), such that $E = E_1 \biguplus \dots \biguplus E_n$, $\Sigma = (E_1 \dots E_n, l)$ is a labeled step sequence generated by UAON and $l(E_i) = U_i$ for $i \in \{1, \dots, n\}$, i.e. $\sigma \in \phi(\lambda(\mathrm{UAON}))$. Initialize $\widetilde{B} = \emptyset$ and $Act = \emptyset$.

(II) Initialize two relations $\prec^{\curlywedge}_{\mathrm{UAON}} = \{(e,f) \in E \times E \mid e \in E_i,\ f \in E_j,\ i < j \wedge l(e) \multimap l(f)\}$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}} = \{(e,f) \in E \times E \mid e \in E_i, f \in E_j, i \leq j \wedge (l(f) \multimap l(e) \vee (\exists z \in T : {}^\bullet l(e) \cap l(f)^\bullet \cap {}^- z \neq \emptyset))\}$ (depending on UAON) specifying possible $\curlywedge$-labeled conditions in accordance with (Cond5') and (Cond6') of Definition 3.11 and Σ (not contradicting the causal relations given by Σ).

(III) Choose arbitrary subsets $\prec^{\curlywedge} \subseteq \prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge} \subseteq \sqsubset^{\curlywedge}_{\mathrm{UAON}}$.

(IV) Introduce $\curlywedge$-labeled conditions corresponding to $\prec^{\curlywedge}$ and $\sqsubset^{\curlywedge}$ to AON as follows: For $(e,f) \in \prec^{\curlywedge}$ create exactly one condition $b \in \widetilde{B}$, set $l(b) = \curlywedge$ and add two arcs $(e,b) \in R$ and $(b,f) \in Act$. For $(e,f) \in \sqsubset^{\curlywedge}$ create exactly one condition $b \in \widetilde{B}$, set $l(b) = \curlywedge$ and add two arcs $(b,e) \in Act$ and $(b,f) \in R$. Delete the considered subsets $\prec^{\curlywedge}$ respectively $\sqsubset^{\curlywedge}$ from the relations $\prec^{\curlywedge}_{\mathrm{UAON}}$ respectively $\sqsubset^{\curlywedge}_{\mathrm{UAON}}$: $\prec^{\curlywedge}_{\mathrm{UAON}}$ becomes $\prec^{\curlywedge}_{\mathrm{UAON}} \setminus \prec^{\curlywedge}$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}}$ becomes $\sqsubset^{\curlywedge}_{\mathrm{UAON}} \setminus \sqsubset^{\curlywedge}$.

(V) If $\prec^{\curlywedge}_{\mathrm{UAON}} = \sqsubset^{\curlywedge}_{\mathrm{UAON}} = \emptyset$ the construction is finished and AON is returned, else (Cond7') of Definition 3.11 is checked for AON. If AON fulfils (Cond7') the construction is finished and AON is returned, else the next step is performed.

(VI) Choose arbitrary subsets $\prec^{\curlywedge} \subseteq \prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge} \subseteq \sqsubset^{\curlywedge}_{\mathrm{UAON}}$, such that $\prec^{\curlywedge} \neq \emptyset$ or $\sqsubset^{\curlywedge} \neq \emptyset$. Then proceed with step (IV).

Note that this algorithm is non-deterministic. First the construction of UAON in step (I) (UAON is not unique) as well as the choice of $\Sigma = (E_1 \dots E_n, l)$ are non-deterministic. Second choosing subsets of possible $\curlywedge$-labeled conditions in step (III) and step (VI) is non-deterministic. Nevertheless the algorithm always terminates in finite time, since step (I) terminates [22], and the sets $\prec^{\curlywedge}_{\mathrm{UAON}} \subseteq E \times E$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}} \subseteq E \times E$ specifying possible $\curlywedge$-labeled conditions are finite. It remains to be shown that the computed net AON is actually a ca-process, independently from the choices in the non-deterministic algorithm. There are two termination criteria in step (V):

(1) $\prec^{\curlywedge}_{\mathrm{UAON}} = \sqsubset^{\curlywedge}_{\mathrm{UAON}} = \emptyset$.

(2) AON fulfils (Cond7').

For both, we have to prove that the resulting net $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ is a ca-process of NI.

Lemma 3.2. $\mathrm{AON} \in \alpha'(NI) = \mathcal{L}AN$ for the net AON constructed with the above algorithm.

Proof:
First we have to show that AON is an ao-net. The only defining property of ao-nets, which is not obvious, is that $\mathcal{S}(\mathrm{AON}) = (E, \prec_{loc}, \sqsubset_{loc}, l|_E)$ is $\diamondsuit$-acyclic. This follows by construction from $(e \prec_{loc} f \Longrightarrow e \in E_i, f \in E_j, i < j)$ and $(e \sqsubset_{loc} f \Longrightarrow e \in E_i, f \in E_j, i \leq j)$.

Since UAON is a process net of $Und(NI)$, by construction (Cond1), (Cond2), (Cond3) and (Cond4) in Definition 3.11 are satisfied by AON (compare Definition 6.2 in [22]). The definition of $\prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}}$ guarantees (Cond5') in Definition 3.11.

To prove the uniqueness in (Cond6') ("exactly one"), assume that there are two $\curlywedge$-labeled conditions $c, c' \in \widetilde{B}$ such that $f \multimap e$ through c and c'. The only possibility for this is that c is introduced corresponding to $(f, e) \in \prec^{\curlywedge}_{\mathrm{UAON}}$ and c' is introduced corresponding to $(e, f) \in \sqsubset^{\curlywedge}_{\mathrm{UAON}}$. In this case the definition of $\prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}}$ implies that $e \in E_i$ and $f \in E_j$ for $j < i \wedge i \leq j$. A contradiction.

Concerning the termination criterion (2), it is obvious that AON fulfills (Cond7') in Definition 3.11. For $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l)$ resulting from the algorithm terminating with criterion (1), property (Cond7') can be proven as shown in the proof of Proposition 9.4 in [22]: Given $e \in E$, $S \in \mathrm{WSL}(\mathrm{AON})$, such that ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b, e) \in Act\} \subseteq S$, we have to show $l(S \cap B) \leq {}^- l(e)$. Let $e \in E_i$. By construction $C = E_1 \cup \ldots \cup E_{i-1}$ is a weak configuration defining a weak slice S_C. Since σ is enabled in NI, we get $l(S_C \cap B) \leq {}^- l(e)$ (use (Cond4) in Definition 3.11). Thus it is enough to show $l(S \cap B)(p) \leq l(S_C \cap B)(p)$ for all $p \in P$ fulfilling ${}^- l(e)(p) < \infty$. If we assume the opposite, then there is $b \in S \setminus S_C$ and $p \in P$ fulfilling ${}^- l(e)(p) < \infty$ and $l(b) = p$. We distinguish the following cases each leading to a contradiction:

(i) ${}^\bullet b = b^\bullet = \emptyset$: This implies $b \in S_C$.

(ii) $\exists f \in E_j,\ j < i :\ b^\bullet = \{f\}$: This implies $l(f) \multimap l(e)$. That means $(f, e) \in \prec^{\curlywedge}_{\mathrm{UAON}}$. Therefore, there exists c with ${}^\bullet c = \{f\}$ such that $f \multimap e$ through c. It follows $c \in S$, i.e. $b \notin S$.

(iii) $\exists f \in E_j,\ j \geq i :\ {}^\bullet b = \{f\}$: This implies $l(f) \multimap l(e)$, i.e. $(e, f) \in \sqsubset^{\curlywedge}_{\mathrm{UAON}}$. Therefore, there exists c with $c^\bullet = \{f\}$ such that $f \multimap e$ through c. It follows $c \in S$, i.e. $b \notin S$.

(iv) $({}^\bullet b = \emptyset) \wedge (\exists f \in E_j,\, j \geq i :\ b^\bullet = \{f\})$: This implies $b \in S_C$.

(v) $(\exists f \in E_j,\, j < i :\ {}^\bullet b = \{f\}) \wedge (b^\bullet = \emptyset)$: This implies $b \in S_C$.

(vi) $\exists f \in E_j,\, g \in E_k,\, j < i \leq k :\ {}^\bullet b = \{f\} \wedge b^\bullet = \{g\}$: This implies $b \in S_C$.

□

Denote the set of ca-processes constructible with the presented (non-deterministic) algorithm from an enabled step sequence $\sigma = U_1 \dots U_n$ of NI by $\pi'(\sigma)$. The processes in $\pi'(\sigma)$ are exactly the ca-processes of NI having σ (provided with respective labels) as an execution. This finally shows the aim of the construction of processes from step sequences.

Lemma 3.3. $\pi'(\sigma) = \{\mathrm{AON} \in \mathcal{L}AN \mid \sigma \in \phi(\lambda(\mathrm{AON}))\}$.

Proof:
Let $\sigma = U_1 \dots U_n$. We first prove $\pi'(\sigma) \subseteq \{\mathrm{AON} \in \mathcal{L}AN \mid \sigma \in \phi(\lambda(\mathrm{AON}))\}$. We already showed $\pi'(\sigma) \subseteq \mathcal{L}AN = \alpha'(NI)$. It remains to show $\sigma \in \phi(\lambda(\mathrm{AON}))$ for $\mathrm{AON} \in \pi'(\sigma)$. For this it is enough to show that (in the notation given in step (I)) the labeled step sequence $\Sigma = (E_1 \dots E_n, l)$ is generated by AON, i.e. $\Sigma \in \lambda(\mathrm{AON})$. This is guaranteed by $\Sigma \in \lambda(\mathrm{UAON})$ and the definition of $\prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}}$.

Second we prove $\pi'(\sigma) \supseteq \{\mathrm{AON} \in \mathcal{L}AN \mid \sigma \in \phi(\lambda(\mathrm{AON}))\}$. Let $\mathrm{AON} = (B \uplus \widetilde{B}, E, R, Act, l) \in \mathcal{L}AN$ fulfill $\sigma \in \phi(\lambda(\mathrm{AON}))$ and let $\Sigma = (E_1 \dots E_n, l)$ be a labeled step sequence generated by AON and $l(E_i) = U_i$ for $i \in \{1, \dots, n\}$, $E = E_1 \biguplus \dots \biguplus E_n$. Since Σ is also generated by the process $Und(\mathrm{AON})$ underlying AON, there is a non-deterministic choice in step (I) leading to $\mathrm{UAON} = Und(\mathrm{AON})$ and the labeled step sequence Σ. All $\curlywedge$-labeled conditions $\widetilde{B}$ in AON are in accordance with (Cond5') and (Cond6') of Definition 3.11 and Σ. Thus $\widetilde{B}$ corresponds to two subsets $\prec^{\curlywedge} \subseteq \prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge} \subseteq \sqsubset^{\curlywedge}_{\mathrm{UAON}}$ in the sense that choosing non-deterministically $\prec^{\curlywedge}$ and $\sqsubset^{\curlywedge}$ in step (III), the set of $\curlywedge$-labeled conditions introduced in step (IV) coincides with $\widetilde{B}$. Therefore, there is a non-deterministic choice in step (III) leading to AON in step (IV). Since AON fulfills (Cond7'), AON is returned as the constructed ca-process in step (V), i.e. $\mathrm{AON} \in \pi'(\sigma)$. □

With the presented construction algorithm the requirements interrelated with the mapping π in the semantical framework of Figure 4 are fulfilled for ca-processes in a similar manner as for a-processes or processes of p/t-nets. In these two cases π is also defined by a non-deterministic algorithm constructing process nets [22]. Although it is not explicitly mentioned in the semantical framework, the performance of such a construction algorithm is important for the practical applicability of a process definition. In our case, the number of possible $\curlywedge$-conditions specified by $\prec^{\curlywedge}_{\mathrm{UAON}}$ and $\sqsubset^{\curlywedge}_{\mathrm{UAON}}$ is at most quadratic in the number of events E, which means that the number of repetitions of the steps (IV) – (VI) of the algorithm is polynomial. As shown in [22], step (I) of the algorithm runs in linear time. Thus, only checking (Cond7') in step (V) may be not efficient, since there exists an exponential number of (weak) slices in the number of events. But current research results on a similar topic summarized in [24] show that there exists an algorithm polynomial in time solving this problem. In [24] we present a polynomial algorithm (based on flow theory) which tests whether an LSO is enabled w.r.t. a given PTI-net. If n denotes the number of nodes of the considered LSO, this algorithm runs in $O(n^3)$ time. According to the following lemma, we can apply this algorithm in step (V).

Lemma 3.4. AON fulfills (Cond7') if and only if the LSO $\kappa(\mathrm{AON})$ is enabled.

Proof:
Let $\kappa(\mathrm{AON})$ be enabled, let S be a weak slice and let e be a node with ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b,e) \in Act\} \subseteq S$. Let C be the weak configuration with $S = S_C$, i.e let S represent the marking reached after the occurrence of all transitions corresponding to C. There is a labeled step sequence $\Sigma = (E_1 \dots E_n, l) \in \epsilon(\kappa(\mathrm{AON}))$ such that $C = E_1 \cup \dots \cup E_k$ and $e \in E_{k+1}$ for some k. Since $\kappa(\mathrm{AON})$ is enabled, the step sequence $l(E_1) \dots l(E_n)$ is an enabled step occurrence sequence. Since the marking reached after the occurrence of the prefix $l(E_1) \dots l(E_k)$ equals $l(S \cap B)$ and enables $l(E_{k+1})$, we get $l(S \cap B) \leq {}^- l(e)$.

Let $l(S \cap B) \leq {}^- l(e)$ for all weak slices S and all events e with ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b,e) \in Act\} \subseteq S$ and let $\Sigma = (E_1 \dots E_n, l) \in \epsilon(\kappa(\mathrm{AON}))$. Assume that, for some $k \geqslant 0$, the prefix $l(E_1) \dots l(E_k)$ is an enabled step occurrence sequence (for $k = 0$ the prefix is empty). The set $C_k = E_1 \cup \dots \cup E_k$ is a weak configuration of AON. For any $e \in E_{k+1}$ there holds ${}^\bullet e \cup \{b \in \widetilde{B} \mid (b,e) \in Act\} \subseteq S_{C_k}$. Since $l(S_{C_k} \cap B) \leq {}^- l(e)$ for all such e, also $l(E_1) \dots l(E_{k+1})$ is an enabled step occurrence sequence (this argumentation also holds for the empty prefix and the initial weak configuration $MIN(\mathrm{AON})$). We deduce that $\kappa(\mathrm{AON})$ is enabled. □

Altogether, we can check (Cond7') for AON through checking enabledness of $\kappa(\mathrm{AON})$ in polynomial time. Therefore, the whole presented algorithm has a polynomial time consumption. More precisely, each deterministic execution of the non-deterministic algorithm returns in polynomial time a ca-process of NI having a given step sequences σ as an execution. Moreover, each such ca-process of NI is constructed by some deterministic execution of the algorithm. There may be exponentially many possible deterministic executions of the non-deterministic algorithm. This is also the case for the non-deterministic construction algorithms for a-processes and processes of p/t-nets. But one deterministic execution in these cases constructs a process in linear time [22]. The linear time bound is reached through certain structural properties in the process definitions. For example in the non-deterministic algorithm constructing a-processes, (Cond7) of Definition 3.10 is fulfilled by construction, since all in an appropriate sense possible 人-labeled conditions are required by (Cond6) of Definition 3.10. Since the ca-process definition is more complex than the a-process definition, it seems very difficult to exploit similar structural properties to get an algorithm constructing ca-processes in linear time. This is a topic of further research. Nevertheless it is possible to apply the construction algorithm for a-processes presented in [22] to construct certain ca-processes, since every a-process is a ca-process.

4. Other Net Classes

In the last part of this paper we shortly discuss several other existing process definitions for Petri nets in the context of the semantical framework of Figure 4, where the considered list of Petri net classes and process definitions is not exhaustive. In particular we focus on the new completeness requirement, which turns out to be fulfilled by many of the existing acknowledged process semantics. This is a further justification that the developed framework including the completeness aim is reasonable.

Note that there are two different step semantics of Petri nets, one based on concurrent steps and one based on synchronous steps. For example, step semantics of p/t-nets is defined through concurrent steps and step semantics of PTI-nets is defined through synchronous steps in this paper. As already mentioned, the term of enabled LPOs can be equivalently defined through concurrent steps and through synchronous steps. As discussed in [17], enabled LSOs can be equivalently defined through synchronous steps or concurrent steps of synchronous steps. Altogether, the semantical framework does not depend on the considered step semantics.

4.1. P/t-nets

As already mentioned in the introduction, it is well known that the classical Goltz-Reisig processes [9, 10] for p/t-nets fulfill all requirements of the semantical framework in Figure 4. All aims of the framework except for the completeness aim can easily be shown (see e.g. [20, 34, 35]). Whether the Goltz-Reisig process definition is complete or not was unknown for several years. In [20], completeness was finally shown for Goltz-Reisig processes. As mentioned in [34], the proof in [20] is quite complicated. In [34, 35] an alternative proof based on a version of the marriage theorem from graph theory is presented. In [15] a third variant of the proof is shown. This self contained variant is based on the so called token flow property.

4.2. Elementary Nets with Extensions

Concerning process definitions of elementary nets, as defined e.g. in [30], and extensions of elementary nets by context arcs [26, 12], the completeness aim is the most complicated aim of the framework. The other aims of the semantical framework are either straightforward observations or proven in the respective papers [30, 26, 12]. Our considerations about algebraic nets in [5, 6, 15, 16] prove the completeness of processes of elementary nets [5, 15], of elementary nets with inhibitor arcs, read arcs and mixed context each equipped with the so called a-posteriori semantics [6, 15], as well as of elementary nets with inhibitor arcs equipped with the a-priori semantics [15, 16]. Namely, in these papers we deduce causal semantics from algebraic semantics. By construction, this causal semantics is exactly the set of enabled causal structures of a given net. As a main result, we show that the set of minimally enabled causal structures equals the set of minimal runs as defined in [30, 26, 12].

4.3. P/t-nets with Capacities

An extension of p/t-nets are capacities restricting the maximal number of tokens in places. Basically there are two different interpretations of capacities, the so called weak and the so called strong capacities as discussed in [7, 15]. In [7] it is shown, that given a p/t net with capacities with an initial marking, both for the strong and weak enabling rule there exists a transformation into a marked p/t-net with the

same number of transitions, such that the step sequences of the net with capacities and the transformed net without capacities are equal. For strong capacities, the transformation is analogous to the mentioned concept of complementation, while for weak capacities the transformation is more complicated. The processes and runs of the transformed net provide then the causal semantics of the p/t-net with capacities. Since the transformed net has equal step semantics as the original net, the relationships between enabled LPOs and runs of a p/t-net without capacities hold also for p/t-nets with capacities. That means p/t-nets with capacities fulfill the presented semantical framework including the completeness aim. Note here that the weak capacity semantics can be generalized by considering LSOs instead of LPOs. For this semantics no process definition exists.

4.4. P/t-nets with Inhibitor Arcs

For an inhibitor net class, which is a special case of the class of PTI-nets, the ca-process definition of this paper provides a sound and complete process semantics. But often there are more simple process definitions for such sub-classes: In [21, 22] the technique of complementation, used to define processes of elementary nets with inhibitor arcs in [12], is extended to define processes of bounded p/t-nets with unweighted inhibitor arcs (a-priori semantics) [21] as well as processes of complemented PTI-nets (a-priori semantics), where every place already has a unique complement place [22]. All aims of the semantical framework of [22] are shown in the respective papers. Since the general idea of generating causalities is the same as in the elementary net case, we assume that these process definitions also fulfill the completeness aim.

While the technique of complementation is very useful for elementary nets and bounded nets, it cannot be applied to unbounded p/t-nets with inhibitor arcs. Firstly we consider unweighted inhibitor arcs. Here we have a process definition in [2, 3, 4] for the a-posteriori case and in [21] for the a-priori case. The a-priori process definition in [21] is based on so called z-conditions, which lead to non-standard occurrence nets with branching conditions. For this definition all aims of the semantical framework of [22] are proven. Completeness, postulating that the processes can model minimal causalities, seems to be also valid, since the z-conditions directly model the dependencies arising from inhibitor arcs. More precisely, a z-conditions is an explicit ”record” that a place is empty. An inhibitor test is then modeled by a read arc to the respective event from the most recent record of the respective inhibitor place being empty. This technique benefits from the fact that, if an inhibitor place is empty, modeled by a z-condition, then some transition producing tokens in the inhibitor place has to occur earlier than some transition consuming tokens in the place. Since this cannot directly be transferred from zero-testing to weighted inhibitor arcs, the technique of z-conditions is not used for PTI-nets. Therefore in [22] the a-process semantics (of Definition 3.10) was introduced for this case. We have already copiously discussed that these processes fulfill the presented semantical framework except for the completeness aim. It is shown in [22] that for a restricted net class of the so called PTDI-nets, including standard unweighted inhibitor nets, the a-processes definition meets the completeness aim. In [2, 3, 4] process and causal semantics are introduced for so called contextual p/t-nets, extending p/t-nets by read arcs and unweighted inhibitor arcs, w.r.t. the a-posteriori case. The definition of processes is based on so called enriched occurrence nets, which contain read arcs and two types of inhibitor arcs producing different causal relations between events. The different inhibitor arcs distinguish the case, when an event, testing a condition via an inhibitor arc, occurs after an event consuming this condition, from the case, when an event, testing a condition via an inhibitor arc, occurs before an event producing this condition. The relation of process semantics,

causal semantics and step semantics is studied in [3, 4]. There it is shown that the presented process semantics is sound and weakly complete and how process nets can be constructed from step sequences. Also completeness seems to be valid: From the nature of unweighted inhibitor arcs it is necessary to introduce either an after- or an before-relation between events as described above, i.e. the causalities introduced in process nets cannot be omitted.

4.5. P/t-nets with Read Arcs

A concept, which is very similar to inhibitor arcs are read arcs. While inhibitor arcs test for the absence of tokens, read arcs test for the presence of tokens in places. Most of the approaches sketched in the last paragraph for inhibitor nets are also developed for or can be carried over to read arcs. In particular one also distinguishes between a-priori and a-posteriori process semantics. For safe nets with read arcs, in [36] an additional intermediate process semantics is considered. This semantics regards time intervals. The causal relationships are based on special causal structures, so called spc-structures. For these structures there is not yet introduced a notion of enabledness and therefore completeness is of course not examined yet (but some requirements of the presented semantical framework are checked in [36]).

4.6. High-level Petri Nets

Lastly, it is interesting to discuss the presented framework for high level Petri nets. There are several process semantics for various variants of high-level nets such as basic high-level nets [31], coloured Petri nets [23, 13] or M-nets [19]. Mostly the high-level process semantics is analogous to the process semantics given by the expansion of the high-level net to a low-level p/t-net. In this case, the results for processes of p/t-nets carry over to high-level nets, i.e. the high-level process definition fulfills the presented semantical framework. But as mentioned in [19, 23] one has to pay attention here, that in the case the colour (or element) set of the high-level net is infinite, the p/t-net expansion is infinite. In [33] a process definition of the special high-level net class of object nets is given.

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 [22] 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 [22] 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 [22]).

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. Concerning existing process semantics of Petri net classes, the considerations in the last section indicate that most aims of [22] are checked for the bigger part of the process definitions. Moreover, a lot of existing process semantics seem to satisfy the aim of completeness and at least for some process definitions there are formal proofs. 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. as a first step to model checking algorithms based on unfoldings of PTI-nets.

References

[1] Billington, J.: Protocol Specification Using P-Graphs, a Technique Based on Coloured Petri Nets, *Petri Nets (2)* (W. Reisig; G. Rozenberg, Ed.), 1492, Springer, 1996.

[2] Busi, N., Pinna, G. M.: Non Sequential Semantics for Contextual P/T Nets, *Application and Theory of Petri Nets* (J. Billington, W. Reisig, Eds.), 1091, Springer, 1996.

[3] Busi, N., Pinna, G. M.: Process Semantics for Place/Transition Nets with Inhibitor and Read Arcs, *Fundam. Inform.*, **40**(2-3), 1999, 165–197.

[4] Busi, N., Pinna, G. M.: Comparing Truly Concurrent Semantics for Contextual Place/Transition Nets with Inhibitor and Read Arcs, *Fundam. Inform.*, **44**(3), 2000, 209–244.

[5] Desel, J., Juhás, G., Lorenz, R.: Process Semantics of Petri Nets over Partial Algebra, in: M. Nielsen; D. Simpson [25], 146–165.

[6] Desel, J., Juhás, G., Lorenz, R.: Petri Nets over Partial Algebra, *Unifying Petri Nets* (H. Ehrig; G. Juhás; J. Padberg; G. Rozenberg, Ed.), 2128, Springer, 2001.

[7] Devillers, R.: The Semantics of Capacities in P/T Nets, *Advances in Petri Nets 1989*, 424, 1989.

[8] Donatelli, S., Franceschinis, G.: Modelling and Analysis of Distributed Software Using GSPNs, *Petri Nets (2)*, 1492, 1998.

[9] Goltz, U., Reisig, W.: The Non-Sequential Behaviour of Petri Nets., *Information and Control*, **57**(2/3), 1983, 125–147.

[10] Goltz, U., Reisig, W.: Processes of Place/Transition-Nets., *ICALP* (J. Díaz, Ed.), 154, Springer, 1983.

[11] Grabowski, J.: On Partial Languages., *Fundamenta Informaticae*, **4**(2), 1981, 428–498.

[12] Janicki, R., Koutny, M.: Semantics of Inhibitor Nets., *Inf. Comput.*, **123**(1), 1995, 1–16.

[13] Januzaj, V.: CPNunf: A tool for McMillan's Unfolding of Coloured Petri Nets, *CPN'07*, 2007.

[14] Jensen, K.: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.*, vol. 1-3 of *Monographs in Theoretical Computer Science*, Springer, 1992, 1994, 1997.

[15] Juhas, G.: Are these Events Independend? It Depends!, Habilitation, 2005.

[16] Juhas, G., Lorenz, R., Mauser, S.: Synchronous + Concurrent + Sequential = Earlier than + Not later than, *ACSD*, IEEE Computer Society, 2006.

[17] Juhas, G., Lorenz, R., Mauser, S.: Causal Semantics of Algebraic Petri Nets distinguishing Concurrency and Synchronicity, *Fundam. Inform.*, 2007, to appear.

[18] Juhás, G., Lorenz, R., Mauser, S.: Complete Process Semantics for Inhibitor Nets, *ICATPN* (J. Kleijn, A. Yakovlev, Eds.), 4546, Springer, 2007.

[19] Khomenko, V., Koutny, M.: Branching Processes of High-Level Petri Nets, *TACAS* (H. Garavel, J. Hatcliff, Eds.), 2619, Springer, 2003.

[20] Kiehn, A.: On the Interrelation Between Synchronized and Non-Synchronized Behaviour of Petri Nets., *Elektronische Informationsverarbeitung und Kybernetik*, **24**(1/2), 1988, 3–18.

[21] Kleijn, H. C. M., Koutny, M.: Process Semantics of P/T-Nets with Inhibitor Arcs, in: M. Nielsen; D. Simpson [25], 261–281.

[22] Kleijn, H. C. M., Koutny, M.: Process Semantics of General Inhibitor Nets, *Inf. Comput.*, **190**(1), 2004, 18–69.

[23] Kozura, V. E.: Unfoldings of Coloured Petri Nets, *Ershov Memorial Conference* (D. Björner, M. Broy, A. V. Zamulin, Eds.), 2244, Springer, 2001.

[24] Lorenz, R., Mauser, S., Bergenthum, R.: Testing the Executability of Scenarios in General Inhibitor Nets, *ACSD*, IEEE Computer Society, 2007.

[25] M. Nielsen; D. Simpson, Ed.: *Application and Theory of Petri Nets 2000, 21st International Conference, ICATPN 2000, Aarhus, Denmark, June 26-30, 2000, Proceeding*, vol. 1825 of *Lecture Notes in Computer Science*, Springer, 2000.

[26] Montanari, U., Rossi, F.: Contextual Nets, *Acta Inf.*, **32**(6), 1995, 545–596.

[27] Peterson, J.: *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981.

[28] Pratt, V.: Modelling Concurrency with Partial Orders., *Int. Journal of Parallel Programming*, **15**, 1986, 33–71.

[29] R.Janicki, Koutny, M.: Order Structures and Generalisations of Szpilrajn's Theorem, *FSTTCS* (R. K. Shyamasundar, Ed.), 761, Springer, 1993.

[30] Rozenberg, G., Engelfriet, J.: Elementary Net Systems, in: W. Reisig; G. Rozenberg [37], 12–121.

[31] Smith, E.: Principles of High-Level Net Theory, in: W. Reisig; G. Rozenberg [37], 174–210.

[32] Szpilrajn, E.: Sur l'extension de l'ordre partiel., *Fundamenta Mathematicae*, **16**, 1930, 386–389.

[33] Valk, R.: *On Processes of Object Petri Nets*, Technical report, Hamburg, Germany, 1996.

[34] Vogler, W.: *Modular Construction and Partial Order Semantics of Petri Nets.*, vol. 625 of *Lecture Notes in Computer Science*, Springer, 1992.

[35] Vogler, W.: Partial Words Versus Processes: a Short Comparison., *Advances in Petri Nets: The DEMON Project* (G. Rozenberg, Ed.), 609, Springer, 1992.

[36] Vogler, W.: Partial Order Semantics and Read Arcs., *MFCS* (I. Prívara, P. Ruzicka, Eds.), 1295, Springer, 1997.

[37] W. Reisig; G. Rozenberg, Ed.: *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, vol. 1491 of *Lecture Notes in Computer Science*, Springer, 1998.