Causality in Extensions of Petri Nets

Jetty Kleijn and Maciej Koutny

TECHNICAL REPORT SERIES

No. CS-TR-1294 **November 2011**

TECHNICAL REPORT SERIES

No. CS-TR-1293 November, 2011

Causality in Extensions of Petri Nets

J. Kleijn and M. Koutny

Abstract

The causal semantics of standard net classes like Elementary Net Systems and Place/Transition Nets, is typically expressed in terms of partially ordered sets of transition occurrences. In each such partial order, causally related occurrences are ordered while concurrent transition occurrences remain unordered. Partial order semantics can, in particular, support model checking by efficient verification techniques based on net unfoldings.

To enhance the modelling power of standard net classes, one can introduce different forms of 'testing' using, for example, inhibitor arcs. However, the causal semantics of such extended nets can often no longer be described solely in terms of partial orders. In this paper, we explain what modifications to the partial order semantics are needed in order to provide a satisfactory treatment for nets with activator, inhibitor and mutex arcs. On the technical side, the proposed solution is based on causal structures which enrich partial orders with additional order relations corresponding to other aspects of causality. With en-systems as our starting point, we discuss how their extensions can be treated using these richer notions of causality.

Bibliographical details

KLEIJN, J., KOUTNY, M.

Causality in Extensions of Petri Nets
[By] J. Kleijn, M. Koutny
Newcastle upon Tyne: Newcastle University: Computing Science, 2011.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1294)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1294

Abstract

The causal semantics of standard net classes like Elementary Net Systems and Place/Transition Nets, is typically expressed in terms of partially ordered sets of transition occurrences. In each such partial order, causally related occurrences are ordered while concurrent transition occurrences remain unordered. Partial order semantics can, in particular, support model checking by efficient verification techniques based on net unfoldings. To enhance the modelling power of standard net classes, one can introduce different forms of 'testing' using, for example, inhibitor arcs. However, the causal semantics of such extended nets can often no longer be described solely in terms of partial orders. In this paper, we explain what modifications to the partial order semantics are needed in order to provide a satisfactory treatment for nets with activator, inhibitor and mutex arcs. On the technical side, the proposed solution is based on causal structures which enrich partial orders with additional order relations corresponding to other aspects of causality. With en-systems as our starting point, we discuss how their extensions can be treated using these richer notions of causality.

About the authors

Jetty Kleijn is a visiting fellow within the School of Computing Science, Newcastle University.

Maciej Koutny obtained his MSc (1982) and PhD (1984) from the Warsaw University of Technology. In 1985 he joined the then Computing Laboratory of the University of Newcastle upon Tyne to work as a Research Associate. In 1986 he became a Lecturer in Computing Science at Newcastle, and in 1994 was promoted to an established Readership at Newcastle. In 2000 he became a Professor of Computing Science.

Suggested keywords

ELEMENTARY NET SYSTEMS
ACTIVATOR ARCS
INHIBITOR ARCS
MUTEX ARCS
SEMANTICAL FRAMEWORK
STEP SEQUENCES
PROCESSES
CAUSALITY SEMANTICS

Causality in Extensions of Petri Nets

Jetty Kleijn[1] and Maciej Koutny[2]

[1] LIACS, Leiden University
Leiden, 2300 RA The Netherlands
kleijn@liacs.nl
[2] School of Computing Science, Newcastle University
Newcastle upon Tyne NE1 7RU, United Kingdom
maciej.koutny@ncl.ac.uk

Abstract. The causal semantics of standard net classes like Elementary Net Systems and Place/Transition Nets, is typically expressed in terms of partially ordered sets of transition occurrences. In each such partial order, causally related occurrences are ordered while concurrent transition occurrences remain unordered. Partial order semantics can, in particular, support model checking by efficient verification techniques based on net unfoldings.
To enhance the modelling power of standard net classes, one can introduce different forms of 'testing' using, for example, inhibitor arcs. However, the causal semantics of such extended nets can often no longer be described solely in terms of partial orders. In this paper, we explain what modifications to the partial order semantics are needed in order to provide a satisfactory treatment for nets with activator, inhibitor and mutex arcs. On the technical side, the proposed solution is based on causal structures which enrich partial orders with additional order relations corresponding to other aspects of causality. With EN-systems as our starting point, we discuss how their extensions can be treated using these richer notions of causality.

Keywords: elementary net systems, activator arcs, inhibitor arcs, mutex arcs, semantical framework, step sequences, processes, causality semantics.

1 Introduction

In order to be able to verify complex, distributed systems, i.e., to guarantee correctness of their behaviour, one has to understand the relations between concurrently ongoing operations. This involves, in particular, providing appropriate mathematical abstractions to capture the operational properties of such systems.

Petri nets are a system model related to state machines and similar, sequential, behaviour defining devices. However, the states of Petri nets are distributed (over so-called places) and also their actions (transitions, in Petri net terms) occur purely locally. Whether or not a transition can occur, depends only those components (places) of the state to which it is directly related. Moreover, when it occurs, it affects only neighbouring places. Hence, each transition occurrence (an event) leads to a local change of state. All this induces local interactions between transition occurrences making it possible to extract from a run of a Petri net, the essential causal relationships between events. These local interactions can be derived from so-called processes, i.e., labelled acyclic nets representing the unfolding of a net corresponding to a single execution (with all choices and conflicts resolved). Abstracting from the places leads to a causal semantics expressed in terms of partially ordered sets of occurrences of transitions: causally related events are ordered, while concurrent events remain unordered. Each such partial order describes the causal structure of a single concurrent history or run of the system and as such represents several — closely related — (step) sequences of (simultaneously occurring) transitions, each of them being a possible observation of that run. The standard net classes of Elementary Net Systems (or EN-systems) and Place/Transition Nets (or PT-nets) are typical examples of this approach [1, 26].

As an example, consider Figure 1(a) depicting an EN-system with three step sequences involving the executions of transitions a, b and c, viz. $\sigma_1 = \{a,b\}\{c\}$, $\sigma_2 = \{a\}\{b\}\{c\}$ and $\sigma_3 = \{b\}\{a\}\{c\}$. They can be seen as observations of a single history underpinned by a causal partial order in which a and b are unordered and both a and b precede c.

Consistency between the different levels of abstraction at which one captures the concurrency in the behaviour can be established within a generic approach (the *semantical framework* of [19]) aimed at fitting together systems (i.e., nets from a certain class of Petri nets), abstract causal orders and individual observations.

Partial order semantics as just described can support efficient verification techniques. Rather than exploring the full state space of a system constructed from sequential observations, one uses unfoldings, see [4] for a general description of this idea. The idea behind the resulting more efficient algorithms is to exploit the concurrency (unorderedness) in the behaviour to alleviate the state space explosion problem. For Petri nets, unfoldings and nonsequential net processes provide a truly concurrent semantics with partial orders as a succinct representation of related observations. Unfoldings based on the branching processes from [3] in which also all choices are modelled, are the basis for efficient verification algorithms [5, 18, 23].

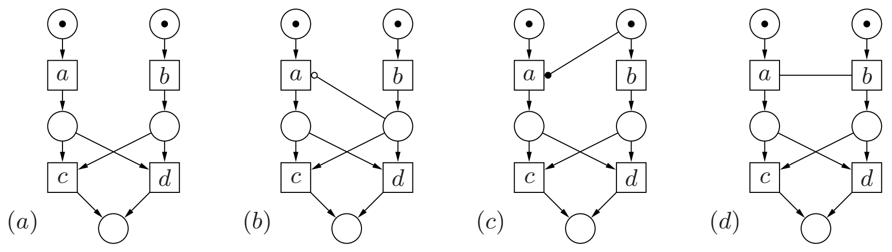


Fig. 1. An EN-system (a); an EN-system with an inhibitor arc joining the output place of transition b with transition a implying that a cannot be fired if the output place of b is not empty (b); an EN-system with an activator arc joining the input place of transition b with transition a implying that a can be fired provided that the input place of b is not empty (c); and an EN-system with a mutex arc between transitions a and b implying that the two transitions cannot be fired in the same step (d).

To enhance the modelling power of the standard net classes one can introduce different forms of 'testing', for example, testing for the absence of a token using inhibitor arcs. This may imply that the causal semantics of such extended Petri net models can no longer be described solely in terms of partial orders.

Figure 1(b) depicts an EN-system with an inhibitor arc. Such an arc between a place and a transition indicates that the place has to be empty for the transition to be able to fire. Hence this net has only two step sequences involving transitions a, b and c, namely $\sigma_1 = \{a, b\}\{c\}$ and $\sigma_2 = \{a\}\{b\}\{c\}$. This is because a can occur before b or simultaneously with b but 'not later than' b (weak causality). These two step sequences can be seen as belonging to the abstract causal history underpinned *not* by causal partial orders but rather by causality structures introduced in [14] — called *stratified order structures* — based on causal partial orders and, in addition, weak causal partial orders. Inhibitor arcs are closely related to activator arcs. Another form of testing is portrayed by the net in Figure 1(c) which depicts an EN-system with an activator arc. Such a 'testing' arc between a place and a transition means that the place has to be non-empty for the transition to be able to fire. As a result, both step sequences and abstract causal histories of this net are exactly the same as in the previous example.

Yet another example, in Figure 1(d), depicts an EN-system with a mutex arc. Such an arc means that the two adjacent transitions may occur in any order but not simultaneously (commutativity). Hence this net has two step sequences involving transitions a, b and c, namely $\sigma_2 = \{a\}\{b\}\{c\}$ and $\sigma_3 = \{b\}\{a\}\{c\}$. They belong to an abstract history underpinned by causality structures introduced in [7, 10] — called *generalised stratified order structures* — based on causal partial orders together with weak causal partial orders and, in addition, a commutativity relation which tells what pairs of events cannot belong to the same step.

In this paper, we explain what modifications to the partial order semantics are needed in order to provide a satisfactory treatment for nets with inhibitor, activator and mutex arcs. The model which we extend with these new types of arcs are Elementary Net systems [26]. This model is *the* basic class of Petri nets and is particularly suited for the study of fundamental properties of concurrent systems. In particular, Elementary Net Systems are the typical concurrency model in which event independence, simultaneity, and unorderedness amount to basically the same semantical phenomenon, making partial orders exactly the right abstract model for their behaviour. We will discuss how the extended classes of EN-systems can be treated with the richer notions of causal semantics using the generic approach provided by the semantical framework of [19]. Finally, we will include Place/Transition Nets into our discussion and reflect upon similarities and differences with the EN-systems approach. As a tutorial survey, this paper provides no proofs, but rather provides 'facts' with references for proofs and more background information, given per (sub)section.

2 Preliminaries

Composing two functions $f : X \to 2^Y$ and $g : Y \to 2^Z$ is defined by $g \circ f(x) = \bigcup_{y \in f(x)} g(y)$, for all $x \in X$. Restricting function f to a subset Z of X is denoted by $f|_Z$. Similarly, the restriction of a binary relation $R \subseteq X \times Y$ to a subset Z of $X \times Y$

is denoted by $R|_Z$. We may use the infix notation $x\,R\,y$ to denote that $(x, y) \in R$. The composition $R \circ Q$ of two relations $R \subseteq X \times Y$ and $Q \subseteq Y \times Z$ comprises all pairs (x, z) in $X \times Z$ for which there is y in Y such that $(x, y) \in R$ and $(y, z) \in Q$. We assume the following notions and notations:

- $R^{-1} = \{(y, x) \mid (x, y) \in R\}$. (reverse)
- $R^0 = id_X = \{(x, x) \mid x \in X\}$. (identity)
- $R^n = R^{n-1} \circ R$. (n-th power, $n \geq 1$).
- $R^+ = R^1 \cup R^2 \cup \ldots$. (transitive closure)
- $R^* = R^0 \cup R^+$. (reflexive transitive closure)
- $R^{\mathsf{sym}} = R^0 \cup R^{-1}$. (symmetric closure)
- R is symmetric, reflexive, irreflexive, transitive if, respectively, $R = R^{-1}$, $id_X \subseteq R$, $id_X \cap R = \varnothing$, $R \circ R \subseteq R$.
- R is acyclic if R^+ is irreflexive.

A *relational structure* is a tuple $rs = (X, Q_1, \ldots, Q_n)$ where X is a finite *domain*, and the Q_i's are binary relations on X (we can select components using the subscript rs, e.g., X_{rs}). For relational structures with the same domain and arity, rs and rs', we write $rs \subseteq rs'$ if the subset inclusion holds component-wise. The intersection $\bigcap \mathcal{R}$ of a set $\mathcal{R}$ of relational structures with the same arity and domain is defined component-wise.

A *sequence* over a finite set X is a finite string $x_1 \ldots x_n$ of symbols x_i from X. A *step* over X is a non-empty subset of X, and a *step sequence* over X is a finite string $X_1 \ldots X_n$ of steps. A step sequence is *singular* if the X_i's are mutually disjoint. The empty (step) sequence, corresponding to the case $n = 0$, is denoted by λ. As singleton sets can be identified with their only elements, sequences can be regarded as special step sequences. Moreover, we will drop the set brackets of singleton sets.

A *labelling* ℓ of a set X is a function from X to a set of labels $\ell(X)$, and a *labelled set* is a pair (X, ℓ) where X is a set and ℓ is a labelling of X. The labelling is extended to finite sequences of elements x_i of X by $\ell(x_1 \ldots x_n) = \ell(x_1) \ldots \ell(x_n)$, and to finite sequences of subsets X_i of X by $\ell(X_1 \ldots X_n) = \ell(X_1) \ldots \ell(X_n)$. To make the labelling explicit, we will sometimes denote a labelled step sequence by (σ, ℓ). We also will use $\phi(\sigma, \ell) = \ell(\sigma)$ to indicate that we 'forget' about the underlying elements but rather focus on the step sequence $\ell(\sigma)$ over $\ell(X)$.

We *assume* throughout that all sets in this paper are *labelled sets*, with the default labelling simply being the identity function. If the actual labelling is irrelevant for a particular definition or result, it may be omitted. Moreover, whenever it is stated that two domains are the same, we implicitly assume that their labellings are identical.

3 Causal partial orders and order structures

To capture the intrinsic causal relationships between events occurring in a concurrent system history, one normally resorts to using a suitable *ordering relation*. In its basic form, such a relation is a partial order (reflecting the generally accepted view that causality is transitive and acyclic). However, for systems with a complex structure, partial orders may need to be extended to more expressive *order structures* which support additional relations between events, such as *weak* causality. We will present two kinds of such extended order structures.

When using (causal) ordering relations in the treatment of concurrent histories, there are two crucial issues which need to be satisfactorily addressed. The first is the relationship with their associated executions or observations, typically captured by sequences or step sequences of events. To be meaningful, an ordering relation should be a faithful abstraction of a set of executions in the sense that each of these correspond to the given order (should be allowed as an execution). Moreover, there should be an unambiguous way of deriving an ordering relation from a set of observations by capturing all essential causal orderings between events while ignoring coincidental ordering in any concrete observation. We will refer to such a property as *Abstraction*. The second issue is related to the way ordering relations are derived. Intuitively, an overall causal ordering relation is built up from smaller, more direct local, causal ordering relations by applying some notion of transitivity. We will refer to such an operation as *Closure*.

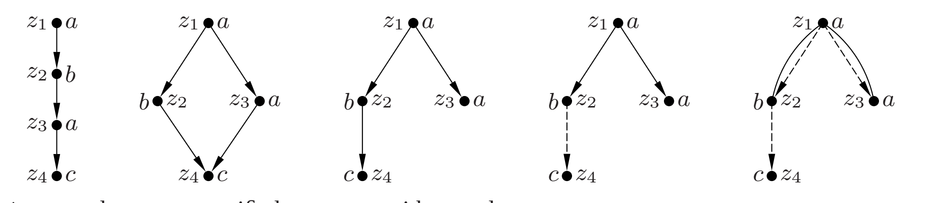
3.1 Partial orders

A *partially ordered set* (or poset) $po = (X, \prec)$ is a relational structure comprising a finite set X and an irreflexive and transitive binary relation $\prec$ on X. Two distinct elements x, y of X are *unordered*, $x \frown y$, if neither $x \prec y$ nor $y \prec x$. We denote $a \overset{\frown}{\prec} b$ if $a \prec b$ or $a \frown b$.

Intersecting posets to filter out their common ordering is a sound operation yielding a new poset.

Fact 1 (poset intersection) *If $\mathcal{PO}$ is a non-empty set of posets with a common domain, then $\bigcap \mathcal{PO}$ is a poset with the same domain.*

A poset po is *total* (or linear) if all pair of distinct elements of X are ordered, and *stratified* (or weak) if $\frown \cup id_X$ is an equivalence relation. Note that all total posets are also stratified. If a poset represents a history of a concurrent system, then $x \prec y$ means that x can only be observed before y, while $x \frown y$ means that x and y can be observed in any order, even simultaneously. In Figure 2, tpo_0 is a total poset and spo_0 is a stratified poset



tpo_0: total spo_0: stratified po_0: neither total nor stratified sos_0: SO-structure $gsos_0$: GSO-structure

Fig. 2. Hasse diagrams of posets and order structures showing also the labels (a, b and c) of their elements. Solid arcs represent $\prec$, dashed arcs represent $\sqsubset$, and solid edges represent $\rightleftharpoons$.

To formulate the *Abstraction* property for posets, we first need to make it clear which executions correspond to a given (causal) poset po. A total poset tpo is a *linearisation* of po if $tpo \subseteq po$, while a stratified poset spo is a *stratification* of po if $spo \subseteq po$. (That is, po is a faithful abstraction of tpo and spo.) We denote this respectively by $tpo \in \mathsf{lin}(po)$ and $spo \in \mathsf{strat}(po)$. In Figure 2, $tpo_0 \in \mathsf{lin}(po_0)$ and $spo_0 \in \mathsf{strat}(po_0)$. Conversely, po captures all essential orderings present in its linearisations or stratifications, respectively.

Fact 2 (poset abstraction [27]) *For every poset po, $\mathsf{lin}(po) \neq \varnothing$ and*

$$po = \bigcap \mathsf{lin}(po)\ .$$

The above fact, known as *Szpilrajn's Theorem*, implies that a poset is uniquely determined by the intersection of its linearisations. The same holds for its stratifications.

Fact 3 (poset abstraction [15]) *For every poset po, $\mathsf{strat}(po) \neq \varnothing$ and*

$$po = \bigcap \mathsf{strat}(po)\ .$$

The *Poset Closure* property described next is simple and indeed standard, but it is still a good idea to state it explicitly as we will soon generalise it to more complicated order structures.

A *pre-poset* is a relational structure $\varrho = (X, \prec)$ such that $\prec^+$ is irreflexive. In such a case, its *po-closure* is defined as $\varrho^{\mathsf{po}} = (X, \prec^+)$. Intuitively, $\prec$ indicates which of the executed actions are *directly* causally related and ϱ^{po} provides a full account of both direct and indirect (derived) causality between events. Therefore, we require that $\prec$ be acyclic, i.e., $\prec^+$ is irreflexive. Then its transitive closure yields the overall causality relationship.

Fact 4 (poset closure) *For every pre-poset ϱ, ϱ^{po} is a poset.*

As we already mentioned, individual executions of a concurrent systems are often represented by sequences of events or sequences of sets of simultaneously occurring events (step sequences). Both are language theoretic rather than order theoretic notions, but there is a straightforward way to move between these two representations. Given a stratified poset $spo = (X, \prec)$, there is a unique enumeration $X_1, \ldots, X_k$ of the equivalence classes of the relation $\frown \cup id_X$ such that $x \prec y$, for all $x \in X_i$ and $y \in X_j$ and $i < j$. We then associate with spo the singular step sequence $\mathsf{steps}(spo) = X_1 \ldots X_k$. Conversely, if $\sigma = X_1 \ldots X_k$ ($k \geq 0$) is a singular step sequence, then

$$\mathsf{spo}(\sigma) = \Big(\bigcup_i X_i, \bigcup_{i<j} X_i \times X_j\Big)$$

is the stratified poset associated with σ. In Figure 2,

$$\begin{array}{ll} \mathsf{steps}(tpo_0) = z_1 z_2 z_3 z_4 & \mathsf{spo}(z_1 z_2 z_3 z_4) \quad\ = tpo_0 \\ \mathsf{steps}(spo_0) = z_1 \{z_2, z_3\} z_4 & \mathsf{spo}(z_1\{z_2, z_3\} z_4) = spo_0\ . \end{array}$$

Fact 5 (posets and step sequences) *$spo = \mathsf{spo}(\mathsf{steps}(spo))$, for every stratified poset spo, and $\sigma = \mathsf{steps}(\mathsf{spo}(\sigma))$, for every singular step sequence σ.*

Hence we can *identify* each stratified poset spo with $\mathsf{steps}(spo)$ or, equivalently, *identify* each singular step sequence σ with $\mathsf{spo}(\sigma)$. This also applies to labelled stratified posets and labelled singular step sequences.

3.2 Stratified order structures

Posets capture an 'earlier than' relationship between the elements of their domains. Their first extension we consider, consists in introducing the concept of a weaker — 'not later than' — relationship.

A *stratified order structure* (or SO-structure) $sos = (X, \prec, \sqsubset)$ comprises two binary relations, $\prec$ (*causality*) and $\sqsubset$ (*weak causality*, in diagrams represented by dashed arcs, see Figure 2) on a finite set X such that, for all $x, y, z \in X$:

$$S1: \ x \not\sqsubset x \qquad\qquad S3: \ x \sqsubset y \sqsubset z \ \wedge \ x \neq z \implies x \sqsubset z$$
$$S2: \ x \prec y \implies x \sqsubset y \qquad\qquad S4: \ x \sqsubset y \prec z \ \vee \ x \prec y \sqsubset z \implies x \prec z \ .$$

Intuitively, $\prec$ represents the 'earlier than' relationship in X, and $\sqsubset$ the 'not later than' relationship. Accordingly, $\prec$ is a partial order, and $x \prec y$ implies $y \not\sqsubset x$. It is easily seen that if spo is a stratified poset, then the relational structure $\mathsf{sos}(spo) = (X_{spo}, \prec_{spo}, \precsim_{spo})$ is an SO-structure.

Again, intersecting SO-structures to filter out their common orderings is a sound operation yielding a new SO-structure.

Fact 6 (sos intersection) *If $\mathcal{SOS}$ is a non-empty set of SO-structures with a common domain, then $\bigcap \mathcal{SOS}$ is an SO-structure with the same domain.*

To formulate the *Abstraction* property for SO-structures, we first need to define which executions would correspond to a given SO-structure sos. A stratified poset spo is an *extension* of sos if $sos \subseteq \mathsf{sos}(spo)$. (Thus sos is faithful to spo.) We denote this by $spo \in \mathsf{ext}(sos)$. In Figure 2, $tpo_0, spo_0 \in \mathsf{ext}(sos_0)$.

Fact 7 (sos abstraction) *For every SO-structure sos, $\mathsf{ext}(sos) \neq \varnothing$ and*

$$sos = \bigcap \mathsf{sos}(\mathsf{ext}(sos)) \ .$$

The *Closure* property for SO-structures generalises the notion of po-closure introduced for posets. A *pre-SO-structure* is a relational structure $\varrho = (X, \prec, \sqsubset)$ such that the relation $\gamma \circ \prec \circ \gamma$ is irreflexive, where $\gamma = (\prec \cup \sqsubset)^*$. Then the *so-closure* is:

$$\varrho^{\mathsf{so}} = (X, \gamma \circ \prec \circ \gamma, \gamma \setminus id_X) \ .$$

Note that in a pre-SO-structure ϱ there are no $x_0, x_1, \ldots, x_n = x_0$ such that $x_0 \prec x_1$ and, for all $0 < i < n$, $x_i \prec x_{i+1}$ or $x_i \sqsubset x_{i+1}$. This can be regarded as a counterpart of the acyclicity required of pre-posets.

Fact 8 (sos closure) *For every pre-SO-structure ϱ, ϱ^{so} is an SO-structure.*

Stratified order structures were independently introduced in [6] and [12]. Their theory has been presented in [15], and they have been used, for example, to model inhibitor and priority systems, asynchronous races and synthesis problems (see, e.g., [17]).

3.3 Generalised stratified order structures

The second extension of causal posets introduces a representation of ‘non-simultaneity’.

A *generalised* SO-*structure* (or GSO-structure) $gsos = (X, \rightleftharpoons, \sqsubset)$ comprises two irreflexive relations, $\rightleftharpoons$ (*commutativity*, which is symmetric) and $\sqsubset$ (*weak causality*, as before) on X such that $(X, \rightleftharpoons \cap \sqsubset, \sqsubset)$ is an SO-structure. Note that commutativity represents the ‘earlier than or later than, but never simultaneous’ relationship. Accordingly, $\rightleftharpoons \cap \sqsubset$ represents the ‘earlier than’ relationship, and so it is required that together with $\sqsubset$ it forms an SO-structure. In fact, one could have defined GSO-structures as $gsos = (X, \prec, \sqsubset, \rightleftharpoons)$ making them a direct generalisation of GSO-structures. However, it is always the case that $\prec$ is the same as the intersection of $\sqsubset$ and $\rightleftharpoons$, and so it can be omitted. It is easily seen that if spo is a stratified poset, then the relational structure $\mathsf{gsos}(spo) = (X_{spo}, \prec_{spo}^{\mathsf{sym}}, \succcurlyeq_{spo})$ is a GSO-structure.

Also in this case, intersecting GSO-structures to filter out their common orderings is a sound operation yielding a new GSO-structure.

Fact 9 (gsos intersection) *If $\mathcal{GSOS}$ is a non-empty set of* GSO-*structures with a common domain, then $\bigcap \mathcal{GSOS}$ is an* GSO-*structure with the same domain.*

To formulate the *Abstraction* property for GSO-structures, we need to define which executions would correspond to a given GSO-structure $gsos$. A stratified poset spo is an *extension* of $gsos$ if $gsos \subseteq \mathsf{gsos}(spo)$. (Thus $gsos$ is faithful to spo.) We denote this by $spo \in \mathsf{ext}(gsos)$. In Figure 2, $spo_0 \in \mathsf{ext}(gsos_0)$. We then obtain that GSO-structures are fully determined by their extensions.

Fact 10 (gsos abstraction) *For every* GSO-*structure $gsos$, $\mathsf{ext}(gsos) \neq \varnothing$ and*

$$gsos = \bigcap \mathsf{gsos}(\mathsf{ext}(gsos)) \ .$$

The *Closure* property for GSO-structures generalises the notion of so-closure introduced for SO-structures. A *pre*-GSO-*structure* is a relational structure $\varrho = (X, \prec, \sqsubset, \rightleftharpoons)$ based on local relationships between events such that the relation $\alpha^{\mathsf{sym}} \cup \beta^{\mathsf{sym}} \cup \rightleftharpoons$ is irreflexive and symmetric, where

$$\alpha = \gamma \circ \prec \circ \gamma \quad \text{and} \quad \beta = \sqsubset^* \circ (\rightleftharpoons \cap \sqsubset^*) \circ \sqsubset^* \quad \text{and} \quad \gamma = (\prec \cup \sqsubset)^* \ .$$

In such a case, its *gso-closure* is defined as $\varrho^{\mathsf{gso}} = (X, \alpha^{\mathsf{sym}} \cup \beta^{\mathsf{sym}} \cup \rightleftharpoons, \gamma \setminus id_X)$. Note that $\rightleftharpoons$ relates events that cannot be executed simultaneously.

Fact 11 (gsos closure) *For every pre-*GSO-*structure ϱ, ϱ^{gso} is a* GSO-*structure.*

Generalised SO-structures were introduced in [7] to represent the most general concurrent histories in the approach of [13]. They were investigated in [10], and used to provide nets with mutex arcs with a semantics in [21].

4 Elementary net systems

All net models considered in this paper have a net as their underlying structure.

A *net* $N = (P, T, F)$ comprises disjoint finite sets of nodes, P and T, called respectively *places* and *transitions*, and the *flow* relation $F \subseteq (T \times P) \cup (P \times T)$. A *marking* of N is a set of places. In diagrams, places (local states) are represented by circles, transitions (actions) by rectangles, the flow relation by directed arcs, and a marking (global state) by *tokens* (small black dots) drawn inside places. The *inputs* and *outputs* of a node x are the sets ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$; moreover, ${}^\bullet x^\bullet = {}^\bullet x \cup x^\bullet$. It is assumed that ${}^\bullet a \neq \varnothing \neq a^\bullet$, for every transition a. The dot-notation extends to sets X of nodes in the usual way, e.g., ${}^\bullet X = \bigcup\{{}^\bullet x \mid x \in X\}$.

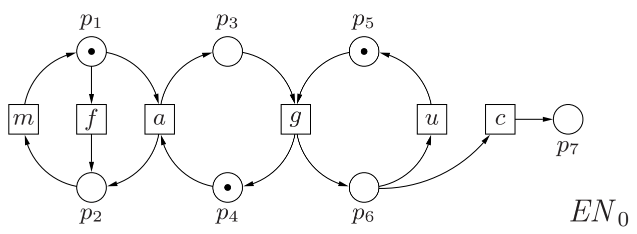


Fig. 3. EN-system model of a producer/consumer system.

Figure 3 shows a net model of a system consisting of a producer, a buffer of capacity one, and a consumer. The producer can execute: m (making an item), a (adding a new item to the buffer), and f (failing to add an item). The consumer can execute: g (getting an item), u (using the item), and c (completing the work). The buffer executes cyclically the a and g actions. The three components operate independently with shared actions being executed jointly. Figure 3 also shows an (initial) marking $M = \{p_1, p_4, p_5\}$.

Net executions can be captured by sequences of steps of transitions. A *step* of a net is a set U of transitions such that ${}^\bullet t^\bullet \cap {}^\bullet v^\bullet = \varnothing$, for all $t \neq v \in U$. It is *enabled* at a marking M if ${}^\bullet U \subseteq M$ and $U^\bullet \cap M = \varnothing$. In such a case, the *execution* of U leads to marking $M' = (M \setminus {}^\bullet U) \cup U^\bullet$. We denote this by $M[U\rangle M'$.

A *step sequence* from a marking M to a marking M' is a sequence $\sigma = U_1 \dots U_n$ $(n \geq 0)$ of non-empty steps U_i such that $M[U_1\rangle M_1, \dots, M_{n-1}[U_n\rangle M'$, for some $M_1, \dots, M_{n-1}$. We denote this by $M[\sigma\rangle M'$, and call M' *reachable* from M. When all steps U_i are singletons, σ is a *firing sequence*. For the net in Figure 3, we have:

$$\begin{array}{cc} \{p_2,p_3,p_6\}\ [m\rangle\ \{p_1,p_3,p_6\} & \{p_1,p_4,p_5\}\ [a\{m,g\}\{a,c\}m\rangle\ \{p_1,p_3,p_7\} \\ \{p_2,p_3,p_6\}\ [\{m,c\}\rangle\ \{p_1,p_3,p_7\} & \{p_1,p_4,p_5\}\ [amgacm\rangle\ \{p_1,p_3,p_7\}\ . \end{array}$$

An EN-*system* is a tuple $EN = (P, T, F, M_{init})$ such that (P, T, F) is its *underlying* net, and M_{init} is an *initial* marking. Moreover, $\mathsf{steps}(EN)$ and $\mathsf{fseq}(EN)$ comprise respectively all the step sequences and all firing sequences from the initial marking M_{init}. Figure 3 depicts an EN-system with $\mathsf{steps}(EN_0) = \{\lambda, a, ag, am, a\{g, m\}, \dots\}$ and $\mathsf{fseq}(EN_0) = \{\lambda, a, ag, am, agm, amg, \dots\}$.

The *reachability graph* $\mathsf{rg}(EN) = (V, A)$ of EN has V as its set of vertices and A as its set of labelled arcs. V consists of all markings reachable from the initial marking M_{init}, and A is given as $A = \{(M, U, M') \mid M \in V \wedge M[U\rangle M'\}$. Similarly, the *sequential reachability graph* $\mathsf{rg}_{seq}(EN) = (V', A')$ of EN has V' as its set of vertices and A' as its set of labelled arcs. V' consists of all markings reachable from the initial marking M_{init} through firing sequences, and A' is given as $A' = \{(M, t, M') \mid M \in V \wedge M[t\rangle M'\}$. It can be seen that although, in general, $\mathsf{rg}_{seq}(EN)$ is a proper subgraph of $\mathsf{rg}(EN)$, their vertices are the same.

The EN-system in Figure 3 is *contact-free* which means that, for all markings M reachable from M_{init} and transitions t, ${}^\bullet t \subseteq M$ implies $t^\bullet \cap M = \varnothing$. Contact-freeness can always be enforced without influencing the step sequence behaviour, by *complementing* (all or some) places p using fresh places $\widetilde{p}$ satisfying ${}^\bullet p = \widetilde{p}^\bullet$, $p^\bullet = {}^\bullet\widetilde{p}$, and declaring that $\widetilde{p} \in M_{init}$ *iff* $p \notin M_{init}$. For example, in Figure 3, $p_4 = \widetilde{p}_3$. In what follows, *all* EN-*systems* as well as their extensions are assumed to be contact-free.

Reachability graphs and structure

Strong connections between structure and behaviour have been for a long time a rich source of analytical techniques for Petri nets. These connections are particularly direct in the case of EN-systems. To start with, at a marking M, we say that two transitions, t and v, are:

– *independent*, if they are both enabled and the execution of one does not disable the other. In EN-systems this is equivalent to saying that $\{t, v\}$ is a step enabled at M. This is illustrated in Figure 4(a) for $t = f$ and $v = g$.
– *in conflict*, if they are both enabled and the execution of one disables the other. In EN-systems this is equivalent to saying that $\{t, v\}$ is not a step enabled at M. This is illustrated in Figure 4(b) for $t = a$ and $v = f$.
– *causally related*, if one is enabled and its execution makes the other enabled. This is illustrated in Figure 4(c) for $t = m$ and $v = f$.

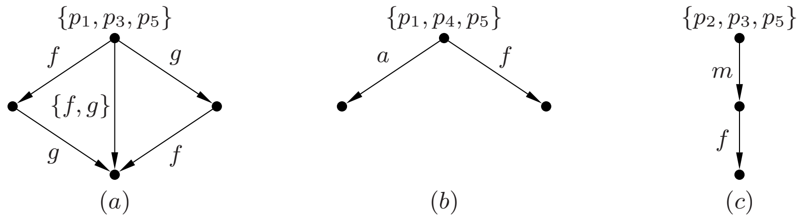


Fig. 4. Independence, conflict and causality in the reachability graph $\mathsf{rg}(EN_0)$.

The above relationships are *behavioural*, in the sense that they refer explicitly to executability at a markings. There is, however, an alternative characterisation, where we say that two transitions, t and v, are *structurally*:

– *independent*, if ${}^\bullet t^\bullet \cap {}^\bullet v^\bullet = \varnothing$; for example, f and g in EN_0.

- *in conflict*, if ${}^\bullet t \cap {}^\bullet v \neq \varnothing$ or $t^\bullet \cap v^\bullet \neq \varnothing$; for example, f and a in EN_0.
- *causally related*, if $t^\bullet \cap {}^\bullet v \neq \varnothing$ or $v^\bullet \cap {}^\bullet t \neq \varnothing$; for example, m and f in EN_0.

We then obtain a direct connection between the behavioural and structural characterisations of fundamental relationships between transitions in EN-systems.

Fact 12 (structure vs. behaviour) *In* EN*-systems, behavioural independence, conflict and causality respectively imply structural independence, conflict and causality.*

In other words, transitions which are structurally independent will never be in conflict or causally related whatever the current marking. Similar remarks hold for conflict and causality.

Another observation concerns the relationship between simultaneity and unorderedness in the behaviour of EN-systems. We can formulate the general property that

$$Simultaneity \iff Unorderedness$$

by which we mean that it is always the case that $M[t,v\rangle M'$ iff $M[tv\rangle M' \wedge M[vt\rangle M'$.

5 Fitting nets and order structures

Given the execution semantics of EN-systems, we could now turn to the development of a causality semantics in terms of occurrence nets and associated causal posets. However, since we aim at a systematic presentation of causality semantics for different net classes, it pays off to develop first a general scheme for doing this. As a result, one can then simplify the formal treatment and also appreciate common properties shared across a range of net classes.

The operational and causality semantics of a class of Petri nets $\mathbb{PN}$ can be presented within a common scheme introduced in [19] (see also [17]) and reproduced here as Figure 5 where N is a net from $\mathbb{PN}$ and:

- $\mathbb{EX}$ are executions (or observations) of nets in $\mathbb{PN}$.
- $\mathbb{LAN}$ are labelled acyclic nets, each representing a concurrent history.
- $\mathbb{LEX}$ are labelled executions of nets in $\mathbb{LAN}$.
- $\mathbb{LCS}$ are labelled causal structures (e.g., order structures) capturing causality relationships between executed actions.

Here, $\mathbb{EX}$ will be step sequences and $\mathbb{LEX}$ will be labelled singular step sequences. However, $\mathbb{LAN}$ and $\mathbb{LCS}$ will depend on the chosen class of nets $\mathbb{PN}$.

The maps in Figure 5 relate the semantical views in $\mathbb{EX}$, $\mathbb{LAN}$, $\mathbb{LEX}$, and $\mathbb{LCS}$:

- ω returns a set of executions, defining the *operational* semantics of N.
- α returns a set of labelled acyclic nets, defining an *axiomatic process* semantics of N.
- π_N returns, for each execution of N, a non-empty set of labelled acyclic nets, defining the *operational process* semantics of N.

$$\begin{tikzcd}
N \in \mathbb{PN} \arrow[r, "\alpha"] \arrow[dd, "\omega"'] & \mathbb{LAN} \arrow[rd, "\kappa"] \arrow[dd, "\lambda"] & \\
& & \mathbb{LCS} \arrow[ld, bend left, "\epsilon"] \\
\mathbb{EX} \arrow[uur, "\pi_N"] & \mathbb{LEX} \arrow[l, "\phi"] \arrow[ur, dashed, bend left, "\imath"] &
\end{tikzcd}$$

Fig. 5. Semantical framework for a class of Petri nets $\mathbb{PN}$. The bold arcs indicate mappings to powersets and the dashed arc indicates a partial function.

– λ returns a set of *labelled* executions for each process of N, and after applying ϕ to such labelled execution one should obtain an execution of N.
– κ associates a labelled *causal* structure with each process of N.
– ϵ and $\imath$ allow one to go back and forth between labelled causal structures and the sets of their labelled executions.

The semantical framework provided by the schema indicates how the different semantical views should agree. According to the rectangle on the left, the Petri net defines processes satisfying certain axioms and moreover all labelled acyclic nets satisfying these axioms can be derived from the executions of the Petri net. Also, the labelled executions of the processes correspond with the executions of the original Petri net. In the triangle on the right, the labelled acyclic nets from $\mathbb{LAN}$, the causal structures from $\mathbb{LCS}$ and the labelled executions from $\mathbb{LEX}$ are related. The order structure defined by a labelled acyclic net can be obtained by combining its executions and, conversely, the stratified extensions of the order structure defined by a labelled acyclic net are the (labelled) executions of that net. Thus the abstract relations between the actions in the labelled causal structures associated with the Petri net will be consistent with its chosen operational semantics.

To demonstrate that these different semantical views agree as captured through this semantical framework, it is sufficient to establish a series of results called *aims*. As there exist four simple requirements (called *properties*) guaranteeing these aims, one can concentrate on defining the semantical domains and maps appearing in Figure 5 and proving these properties.

Property 1 (soundness of mappings) *The maps ω, α, λ, ϕ, $\pi_N|_{\omega(N)}$, κ, ϵ and $\imath|_{\lambda(\mathbb{LAN})}$ are total. Moreover, ω, α, λ, $\pi_N|_{\omega(N)}$ and ϵ always return non-empty sets.*

Property 2 (consistency) *For all $\xi \in \mathbb{EX}$ and $LN \in \mathbb{LAN}$,*

$$\left.\begin{array}{r} \xi \in \omega(N) \\ LN \in \pi_N(\xi) \end{array}\right\} \quad \text{iff} \quad \left\{\begin{array}{l} LN \in \alpha(N) \\ \xi \in \phi(\lambda(LN)) \ . \end{array}\right.$$

Property 3 (representation) $\imath \circ \epsilon = id_{\mathbb{LCS}}$.

Property 4 (fitting) $\lambda = \epsilon \circ \kappa$.

The above four properties imply that the axiomatic (defined through α) and operational (defined through $\pi_N \circ \omega$) process semantics of nets in $\mathbb{PN}$ are in full agreement.

Also, the operational semantics of N (defined through ω) coincides with the operational semantics of the processes of N (defined through $\phi \circ \lambda \circ \alpha$). Finally, the causality in a process of N (defined through κ) coincides with the causality structure implied by its operational semantics (through $\imath \circ \lambda$). That is, we have the following.

Aim 1 $\alpha = \pi_N \circ \omega$.

Aim 2 $\omega = \phi \circ \lambda \circ \alpha$.

Aim 3 $\kappa = \imath \circ \lambda$.

As a consequence, the operational semantics of the Petri net N and the set of labelled causal structures associated with it are related by $\omega = \phi \circ \epsilon \circ \kappa \circ \alpha$.

6 Semantical framework for EN-systems

Some of the notions needed to specialise the general concepts of the semantical framework for EN-systems have already been introduced. We will now introduce the missing ones, starting with the definition of a class of labelled acyclic nets capturing the causality semantics of EN-systems.

An *occurrence net* is a tuple $ON = (P', T', F', \ell)$ such that (P', T', F') is its underlying net[3] and ℓ is a labelling for $P' \cup T'$. Moreover, it is assumed that $|^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$, for every place p, and $\varrho_{ON} = (T', (F' \circ F')|_{T' \times T'})$ is a pre-poset (in other words, F' is acyclic). The default *initial* M^{ON}_{init} and *final* M^{ON}_{fin} markings respectively consist of all places without inputs and outputs. Figure 6 shows an occurrence net labelled by places and transitions of the EN-system EN_0 of Figure 3, with the default initial and final markings $\{b_1, b_2, b_3\}$ and $\{b_6, b_9, b_{10}\}$.

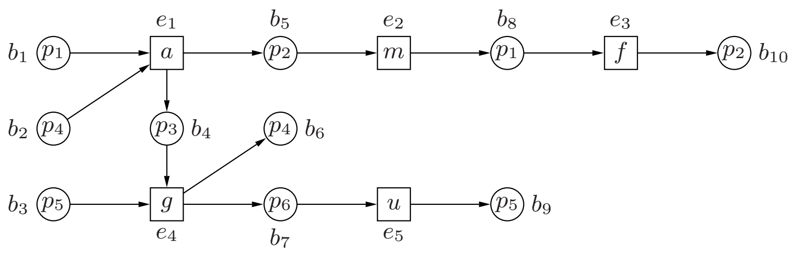


Fig. 6. An occurrence net ON_0 (labels are shown inside the nodes).

Note that, due to the acyclicity of the flow relation and the lack of multiple inputs (or outputs) of places, each transition in T' appears exactly *once* in any step sequence σ satisfying $M^{ON}_{init}[\sigma\rangle M^{ON}_{fin}$. In particular, such a step sequence is singular, and so $\mathsf{spo}(\sigma)$ is a well-defined stratified poset.

The behaviour of an occurrence net ON is captured by the set $\mathsf{steps}(ON)$ of *labelled step sequences*, comprising all pairs $(\sigma, \ell|_{T'})$ such that σ is a step sequence from

[3] The dot-notations, markings, etc, for ON are as those defined for the underlying net.

the default initial marking of ON to the default final marking. For each such labelled step sequence, $\phi(\sigma, \ell|_{T'}) = \ell(\sigma)$. Moreover, $\mathsf{fseq}(ON)$ are the *labelled firing sequences* of ON, i.e., all the labelled step sequences $(\sigma, \ell|_{T'})$ such that σ is a sequence of singleton steps. For the occurrence net of Figure 6, we have $a\{m, g\}\{f, u\} \in \phi(\mathsf{steps}(ON_0))$ as well as $amgfu \in \phi(\mathsf{fseq}(ON_0))$.

Fact 13 (labelled executions) $\mathsf{steps}(ON) \neq \varnothing$ *and* $\mathsf{fseq}(ON) \neq \varnothing$*, for every occurrence net* ON.

For an occurrence net ON, ϱ_{ON} is a pre-poset representing the direct causal relationships between its transitions. Hence, by Fact 4, $\mathsf{po}(ON) = \varrho_{ON}^{\mathsf{po}}$ is the *induced* poset representing all, direct and indirect, causal dependencies between the transitions in T'. For the occurrence net of Figure 6, we have that e_1 causes e_2 directly, but there is only an indirect causal link from e_1 to e_3. Also, there are no causal links between e_3 and e_5 which means that they are independent. This and other relationships can be read out from the diagram of the pre-poset ϱ_{ON_0} shown in Figure 7.

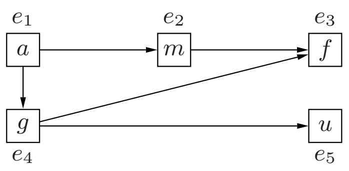


Fig. 7. Pre-poset ϱ_{ON_0} for the occurrence net ON_0.

To define processes of an EN-system, we need to provide an axiomatic characterisation of occurrence nets consistent with the structure of a given EN-system. A *process* of EN-system EN is an occurrence net ON with the labelling ℓ which:

– labels places of ON with places of EN.
– labels transitions of ON with transitions of EN.
– is injective on M_{init}^{ON} and $\ell(M_{init}^{ON}) = M_{init}$.
– is injective on $^\bullet t$ and $t^\bullet$ and, moreover, $\ell(^\bullet t) = {}^\bullet\ell(t)$ and $\ell(t^\bullet) = \ell(t)^\bullet$, for every transition t of ON.

We denote this by $ON \in \mathsf{proc}(EN)$. For example, $ON_0 \in \mathsf{proc}(EN_0)$, where EN_0 and ON_0 are the nets in Figures 3 and 6.

Fact 14 (injective labelling) *The labelling* ℓ *of* $ON \in \mathsf{proc}(EN)$ *is injective on any marking reachable from the default initial marking. It is also injective on any individual step appearing in the step sequences of* $\mathsf{steps}(ON) \neq \varnothing$.

The only missing component of the semantical framework for EN-systems is now the mapping returning processes derived from individual step sequences.

The occurrence net $\mathsf{proc}_{EN}(\sigma)$ *generated* by a step sequence $\sigma = U_1 \ldots U_n$ of EN is the last element in the sequence $ON_0, \ldots, ON_n$ where each ON_k is an occurrence net (P_k, T_k, F_k, ℓ_k) constructed in the following way.

Step 0: $P_0 = \{p^1 \mid p \in M_{init}\}$ and $T_0 = F_0 = \varnothing$.
Step k: Given ON_{k-1} we extend the sets of nodes and arcs as follows:

$$\begin{aligned}
P_k &= P_{k-1} \cup \{p^{1+\triangle p} \mid p \in U_k^\bullet\} \\
T_k &= T_{k-1} \cup \{t^{1+\triangle t} \mid t \in U_k\} \\
F_k &= F_{k-1} \cup \{(p^{\triangle p}, t^{1+\triangle t}) \mid t \in U_k \wedge p \in {}^\bullet t\} \\
&\quad\ \cup \{(t^{1+\triangle t}, p^{1+\triangle p}) \mid t \in U_k \wedge p \in t^\bullet\}\ .
\end{aligned}$$

In the above, the label of each node x^i is set to be x, and $\triangle x$ denotes the number of the nodes of ON_{k-1} labelled by x.

The above construction is illustrated in Figure 8 for the EN-system EN_0 of Figure 3. The resulting occurrence net is isomorphic to ON_0 of Figure 6 which, as we already noted, is a process of EN_0.

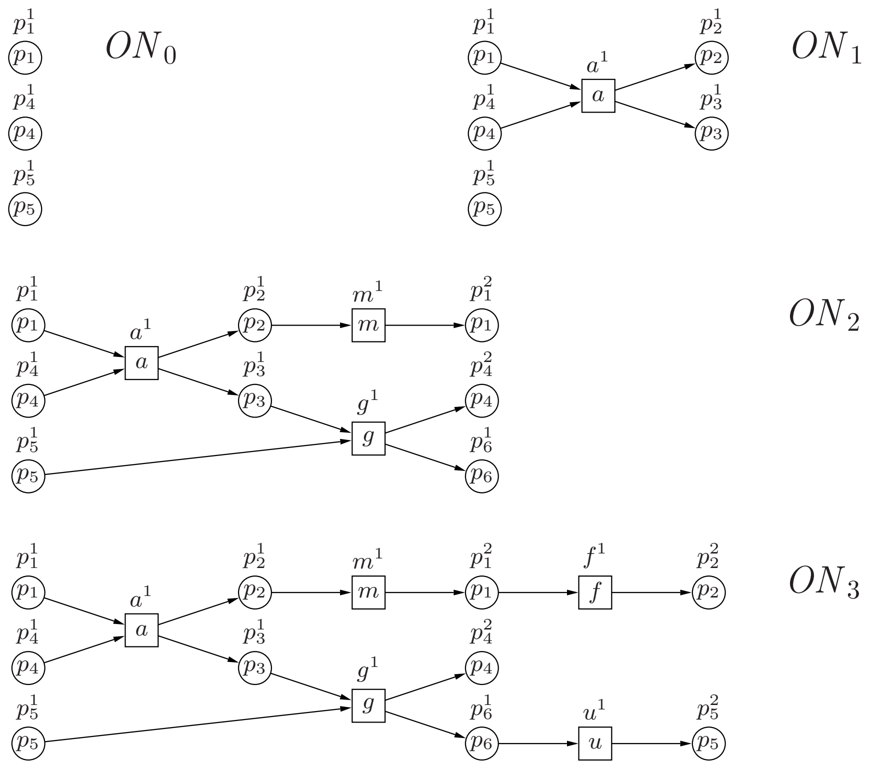


Fig. 8. Process $\mathsf{proc}_{EN_0}(\sigma) = ON_3$ generated for EN_0 and step sequence $\sigma = a\{m, g\}\{f, u\}$.

We will now explain how the four semantical properties can be established for EN-systems and their step sequence semantics (the treatment for firing sequences is almost the same). Referring to Figure 5, where: EN is an EN-system, ON an occurrence net,

(σ, ℓ) a labelled step sequence, po a poset, and Σ a set of labelled singular step sequences with the same domain, we have the following:

$\mathbb{PN}$	are	EN-systems	$\mathbb{EX}$	are	step sequences
$\mathbb{LAN}$	are	occurrence nets	$\mathbb{LEX}$	are	labelled singular step sequences
$\mathbb{LCS}$	are	labelled posets			

$\omega(EN)$	is	$\mathsf{steps}(EN)$	$\alpha(EN)$	is	$\mathsf{proc}(EN)$
$\lambda(ON)$	is	$\mathsf{steps}(ON)$	$\pi_{EN}(\sigma)$	is	$\mathsf{proc}_{EN}(\sigma)$
$\phi(\sigma, \ell)$	is	$\ell(\sigma)$	$\kappa(ON)$	is	$\mathsf{po}(ON)$
$\epsilon(po)$	is	$\mathsf{steps}(\mathsf{strat}(po))$	$\imath(\Sigma)$	is	$\bigcap \mathsf{spo}(\Sigma)$.

Properties 1–4 hold for EN-systems [19, 26]. Below EN is an EN-system and σ its firing sequence, ON is an occurrence net, po is a poset, and Σ is a set of singular step sequences with the same domain. (Note that Fact 17 follows from Facts 3 and 5.)

Fact 15 $\mathsf{steps}(EN)$, $\mathsf{proc}(EN)$, $\mathsf{steps}(ON)$ *and* $\mathsf{strat}(po)$ *are non-empty sets. Moreover,* $\mathsf{po}(ON)$ *and* $\bigcap \mathsf{spo}(\Sigma)$ *are posets, and* $\mathsf{proc}_{EN}(\sigma)$ *is an occurrence net.*

Fact 16 $\mathsf{proc}_{EN}(\sigma)$ *is a process of* EN*. Moreover, if* ON *is a process of* EN *and* $\sigma' \in \phi(\mathsf{steps}(ON))$*, then* $\sigma' \in \mathsf{steps}(EN)$ *and* $ON = \mathsf{proc}_{EN}(\sigma')$.

Fact 17 $po = \bigcap \mathsf{spo}(\mathsf{steps}(\mathsf{strat}(po)))$.

Fact 18 $\mathsf{steps}(ON) = \mathsf{steps}(\mathsf{strat}(\mathsf{po}(ON)))$.

Hence we can claim the semantical aims for EN-systems and step sequences.

Fact 19 *Let* EN *be an* EN*-system, and* ON *be an occurrence net.*

$$\begin{aligned} \mathsf{proc}(EN) &= \mathsf{proc}_{EN}(\mathsf{steps}(EN)) \\ \mathsf{steps}(EN) &= \phi(\mathsf{steps}(\mathsf{proc}(EN))) \\ \mathsf{po}(ON) &= \bigcap \mathsf{spo}(\mathsf{steps}(ON)) \ . \end{aligned}$$

7 EN-systems with activator arcs

This section extends the treatment of concurrency to nets with activator arcs. Consider again the EN-system of Figure 3 and add an activator arc from place p_4 to transition c with a small black circle as arrowhead. In the resulting net ENA_0 shown in Figure 9, c can only be enabled if there is a token in place p_4. However, the execution of transition c does not consume the token in place p_4.

An *elementary net system with activator arcs* (or ENA-system) is a tuple $ENA = (P, T, F, Act, M_{init})$ such that $\mathsf{und}(ENA) = (P, T, F, M_{init})$ is its underlying EN-system, and $Act \subseteq P \times T$ is a set of *activator* arcs. Notions and notations relating to ENA are inherited from $\mathsf{und}(ENA)$. The only new notation is ${}^{\blacklozenge}t$ denoting the set of all the places p where the presence of a token is necessary to enable a transition t, i.e., $(p, t) \in Act$. The behaviour of ENA is also derived from that of $\mathsf{und}(ENA)$ after assuming that a step of transitions U is enabled at a marking M in ENA if it is enabled

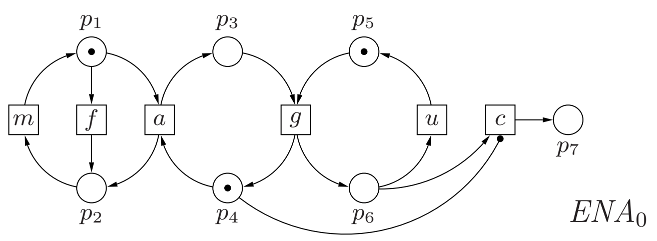


Fig. 9. An ENA-system modelling a second version of the producer/consumer system.

at M in $\mathsf{und}(ENA)$ and ${}^{\blacklozenge}U \subseteq M$, where ${}^{\blacklozenge}U = \bigcup_{t \in U} {}^{\blacklozenge}t$. The marking resulting from the execution of such a U is exactly the same as it would be in $\mathsf{und}(ENA)$. For the ENA-system of Figure 9, we have that $M[\{a,c\}\rangle M'$ and $M[ca\rangle M'$, where $M = \{p_1, p_4, p_6\}$ and $M' = \{p_2, p_3, p_7\}$. However, $M[ac\rangle M'$ does not hold as after executing transition a, a token is removed from the activator place p_4 of transition c.

Reachability graphs of ENA-systems

Reachability in ENA-systems depends on the chosen execution semantics: sequences or step sequences. Taking, as an example the ENA-system in Figure 10(a), we may observe that $M_{init}[\{t,v\}\rangle\{p_3,p_4\}$, but there is no firing sequence σ such that $M_{init}[\sigma\rangle\{p_3,p_4\}$.

Another observation concerns the relationship between simultaneity and unorderedness in the behaviour of ENA-systems. Whereas in the case of EN-systems, we have the general property that *Simultaneity* $\Longleftrightarrow$ *Unorderedness* we now have

$$\mathit{Simultaneity} \quad \Longleftarrow \quad \mathit{Unorderedness}$$

by which we mean that it is always the case that $M[t,v\rangle M'$ if $M[tv\rangle M' \wedge M[vt\rangle M'$. Figure 10($b, c$) shows that the reverse implication does not hold.

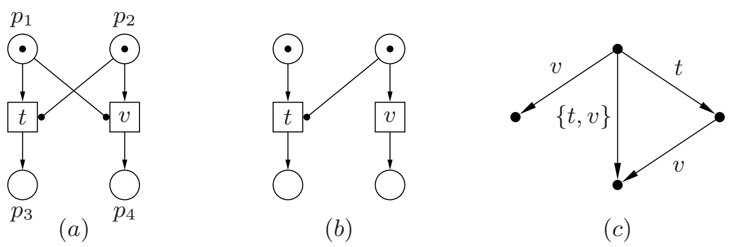


Fig. 10. Two ENA-systems and the reachability graph of the second one.

Semantical framework for ENA-systems

Causality semantics for ENA-systems will be developed by instantiating semantical framework, similarly as in the case of EN-systems. The labelled causal structures employed are SO-structures, while executions remain to be (labelled singular) step sequences. To define processes we extend occurrence nets to include activator arcs.

An *activator occurrence net* (or AO-net) $AON = (P', T', F', Act, \ell)$ is a tuple such that $\mathsf{und}(AON) = (P', T', F', \ell)$ is its underlying occurrence net, and $Act \subseteq P' \times T'$ is a set of *activator* arcs. Moreover, it is assumed that $\varrho_{AON} = (T', \prec_{loc}, \sqsubset_{loc}, \ell|_{T'})$, with

$$\prec_{loc} = (F' \circ F')|_{T' \times T'} \cup (F' \circ Act) \quad \text{and} \quad \sqsubset_{loc} = Act^{-1} \circ F'$$

is a pre-SO-structure (see Figure 11). We then define $\mathsf{sos}(AON) = \varrho^{\mathsf{so}}_{AON}$ to be the SO-structure *induced* by AON.

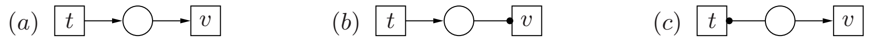


Fig. 11. Two cases (a, b) defining $t \prec_{loc} v$, and one case (c) defining $t \sqsubset_{loc} v$.

The step sequences $\mathsf{steps}(AON)$ of an AO-net AON are defined as for $\mathsf{und}(AON)$, except that the enabling condition takes into account activator arcs.

Fact 20 (labelled executions) $\mathsf{steps}(AON) \neq \varnothing$, *for every* AO-*net* ON.

Note that it may happen that $\mathsf{fseq}(AON) = \varnothing$ even though $\mathsf{steps}(AON) \neq \varnothing$. Take, for example, the AO-net AON_1 in Figure 12(a) for which $\mathsf{steps}(AON_1) = \{\{t, v, w\}z\}$ and $\mathsf{fseq}(AON_1) = \varnothing$ as executing at the default initial marking any transition in $\{t, v, w\}$ means that just one of the remaining transitions will never be enabled, and so the default final marking cannot be reached.

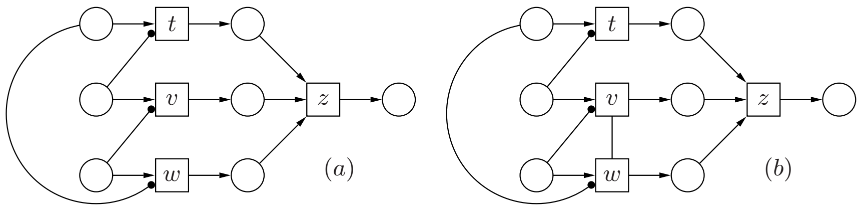


Fig. 12. An AO-net AON_1 (a), and a failed attempt to extend it to an AMO-net (b).

An AO-net represents a concurrent run of a system and has to avoid circularity. Intuitively, $\prec_{loc}$ stands for causal precedence (the first transition has to produce a token for consumption or testing by the second transition) and $\sqsubset_{loc}$ for weak causal precedence (the first transition cannot happen after the second one, since the latter consumes a token which activates the former). Figure 13 shows an AO-net AON_0 labelled by places and transitions of the ENA-system ENA_0 of Figure 9. Its default initial marking is $\{b_1, b_2, b_3\}$, and its default final marking is $\{b_9, b_{10}, b_{11}\}$. Note that transition e_5 weakly precedes transition e_4, i.e., $e_5 \sqsubset_{loc} e_4$. Moreover, we have that $\{m, g\}\{a, c\}$ and $a\{m, g\}ca$ belong to $\phi(\mathsf{steps}(AON_0))$, but $a\{m, g\}ac$ does not.

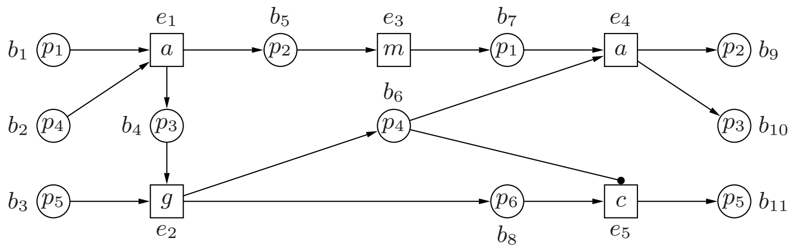


Fig. 13. An activator occurrence net AON_0.

Processes of an ENA-system are similar to those of the underlying EN-system extended with an appropriate treatment of activator arcs. A *process* of ENA is an AO-net AON such that $\mathsf{und}(AON)$ is a process of $\mathsf{und}(ENA)$ and, in addition, ℓ is injective on ${}^{\blacklozenge}t$ and $\ell({}^{\blacklozenge}t) = {}^{\blacklozenge}\ell(t)$ for every transition t of AON. We denote this by $AON \in \mathsf{proc}(ENA)$.

Process generation from a given step sequence is also based on that introduced for EN-systems. The AO-net $\mathsf{proc}_{ENA}(\sigma)$ *generated* by a step sequence $\sigma = U_1 \dots U_n$ of ENA is the last element in the sequence $AON_0, \dots, AON_n$ where each $AON_k = (P_k, T_k, F_k, \ell_k, A_k)$ is an AO-net with the components constructed as in the definition for $\mathsf{proc}_{\mathsf{und}(ENA)}(\sigma)$, and the following additions (see Figure 14):
Step 0: $A_0 = \varnothing$.
Step k: $A_k = A_{k-1} \cup \{(p^{\triangle p}, t^{1+\triangle t}) \mid t \in U \wedge p \in {}^{\blacklozenge}t\}$.

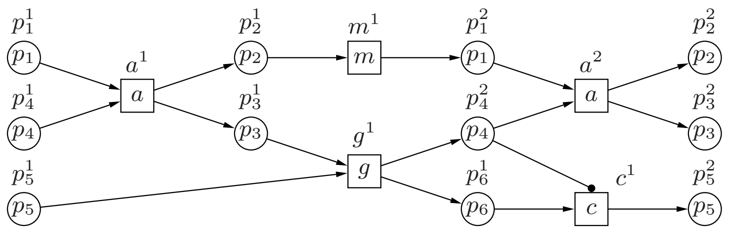


Fig. 14. Process $\mathsf{proc}_{ENA_0}(\sigma)$ generated for ENA_0 and step sequence $\sigma = a\{g, m\}\{a, c\}$.

We will now show the semantical properties formulated above can be established for ENA-systems and their firing sequences. Referring to the notation used in Figure 5, we have the following, where ENA is an ENA-system, AON an AO-net, (σ, ℓ) a labelled step sequence, sos an SO-structure, and Σ a set of labelled singular step sequences with the same domain:

$\mathbb{PN}$	are	ENA-systems	$\mathbb{EX}$	are step sequences	
$\mathbb{LAN}$	are	AO-nets	$\mathbb{LEX}$	are labelled singular step sequences	
$\mathbb{LCS}$	are	labelled SO-structures			

$\omega(ENA)$	is	$\mathsf{steps}(ENA)$	$\alpha(ENA)$	is	$\mathsf{proc}(ENA)$
$\lambda(AON)$	is	$\mathsf{steps}(AON)$	$\pi_{ENA}(\sigma)$	is	$\mathsf{proc}_{ENA}(\sigma)$
$\phi(\sigma, \ell)$	is	$\ell(\sigma)$	$\kappa(AON)$	is	$\mathsf{sos}(AON)$
$\epsilon(sos)$	is	$\mathsf{steps}(\mathsf{ext}(sos))$	$\imath(\Sigma)$	is	$\bigcap \mathsf{sos}(\mathsf{spo}(\Sigma))$.

It can be shown that Properties 1–4 hold. Below ENA is an ENA-system and σ its step sequence, AON is an AO-net, sos is an SO-structure, and Σ is a set of singular step sequences with the same domain. (Note that Fact 23 follows from Facts 5 and 7.)

Fact 21 $\mathsf{steps}(ENA)$, $\mathsf{proc}(ENA)$, $\mathsf{steps}(AON)$ *and* $spo(sos)$ *are non-empty sets. Moreover,* $\mathsf{sos}(AON)$ *and* $\bigcap \mathsf{sos}(\mathsf{spo}(\Sigma))$ *are* SO*-structures, and* $\mathsf{proc}_{ENA}(\sigma)$ *is an* AO*-net.*

Fact 22 $\mathsf{proc}_{ENA}(\sigma)$ *is a process of* ENA*. Moreover, if* AON *be a process of* ENA *and* $\sigma' \in \phi(\mathsf{steps}(AON))$*, then* $\sigma' \in \mathsf{steps}(ENA)$ *and* $AON = \mathsf{proc}_{ENA}(\sigma')$*.*

Fact 23 $sos = \bigcap \mathsf{sos}(\mathsf{steps}(\mathsf{ext}(sos)))$.

Fact 24 $\mathsf{steps}(AON) = \mathsf{ext}(\mathsf{sos}(AON))$.

Hence we can claim the semantical aims for ENA-systems.

Fact 25 *Let* ENA *be an* ENA*-system, and* AON *be an* AO*-net.*

$$\begin{array}{rcl} \mathsf{proc}(ENA) & = & \mathsf{proc}_{ENA}(\mathsf{steps}(ENA)) \\ \mathsf{steps}(ENA) & = & \phi(\mathsf{steps}(\mathsf{proc}(ENA))) \\ \mathsf{sos}(AON) & = & \bigcap \mathsf{sos}(\mathsf{steps}(AON)) \ . \end{array}$$

EN-systems with inhibitor arcs

It is easy to extend the treatment presented above for ENA-systems to EN-systems with inhibitor arcs. Consider again the EN-system of Figure 3 and add to it an inhibitor arc linking the place p_3 and transition c. This yields the net system ENI_0 shown in Figure 15. (Inhibitor arcs are drawn with small open circles as arrowheads.) Adding such an arc means that c cannot be enabled when the buffer is non-empty (a token in place p_3 signifies that the buffer contains an item).

An *elementary net system with inhibitor arcs* (or ENI-system) is a tuple $ENI = (P, T, F, Inh, M_{init})$ such that $\mathsf{und}(ENI) = (P, T, F, M_{init})$ is its underlying EN-system, and $Inh \subseteq P \times T$ is a set of *inhibitor* arcs. Notions and notations relating to ENI are inherited from $\mathsf{und}(ENI)$. The behaviour of ENI is also derived from that of $\mathsf{und}(ENI)$: a step of transitions U is enabled at a marking M of ENI if it is enabled at M in $\mathsf{und}(ENI)$ and $\{p \mid \exists t \in U : (p, t) \in Inh\} \cap M = \varnothing$. The marking resulting from the execution of such a U is exactly the same as in $\mathsf{und}(ENI)$.

Intuitively, the testing for the presence of tokens with activator arcs in ENA-systems has been replaced by testing for their absence with inhibitor arcs in ENI-systems. In

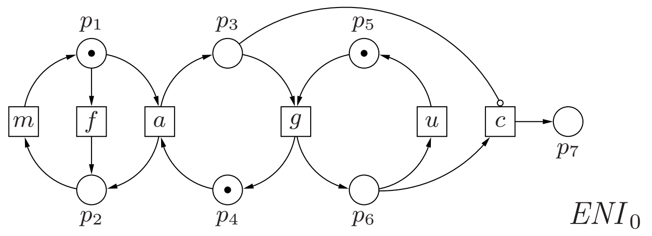


Fig. 15. An ENI-system modelling a third version of the producer/consumer system.

fact, the latter can be faithfully simulated by the former in the case of EN-systems (i.e., they have isomorphic reachability graphs). All we need to assume is that every inhibitor place p has a *complement* place $\widetilde{p}$ satisfying ${}^{\bullet}p = \widetilde{p}^{\bullet}$ and ${}^{\bullet}\widetilde{p} = p^{\bullet}$. Processes of ENI-systems are similar to those of EN-systems with the inhibitor arcs of the system represented by activator arcs which rather than testing for the absence of tokens are used to test for the presence of tokens in complement places. Hence, we assume that each place p of ENI adjacent to an inhibitor arc has a complement place $\widetilde{p}$ in the underlying EN-system. Then, each inhibitor arc (p, t) can be replaced by an equivalent activator arc $(\widetilde{p}, t)$. Since adding complement places is harmless, we can consider the causality treatment of ENI-systems as being obtained through the corresponding ENA-systems. Note that ENI_0 in Figure 15 corresponds in this way to ENA_0 in Figure 9.

8 ENA-systems with mutex arcs

We now extend ENA-systems with mutex arcs prohibiting certain pairs of transitions from occurring simultaneously (i.e., in the same step). Mutex arcs were introduced in [11], and their causality semantics was developed in [21].

Consider Figure 16 which shows another variant of the producer/consumer scheme. In this case, the consumer is allowed to complete (transition c), but never at the same time as the producer makes an item (transition m). Other than that, there are no restrictions on the executions of transitions c and m. To model such a scenario we use a mutex arc between c and m (depicted as an undirected edge). Note that mutex arcs are relating transitions in a direct way. This should however not be regarded as an unusual feature as, for example, Petri nets with priorities also impose direct relations between transitions.

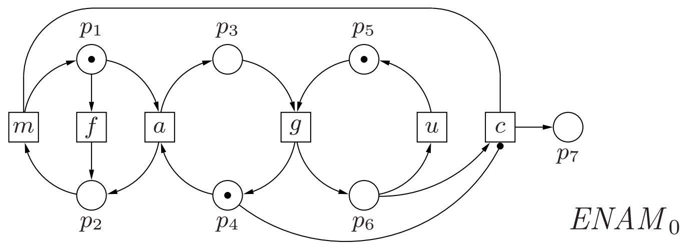


Fig. 16. An ENAM-system modelling a fourth version of the producer/consumer system.

An *elementary net system with activator and mutex arcs* (or ENAM-system) is a tuple $ENAM = (P, T, F, Act, Mtx, M_{init})$ such that $\mathsf{und}(ENAM) = (P, T, F, Act, M_{init})$ is the ENA-system underlying $ENAM$ and $Mtx \subseteq T \times T$ is a symmetric irreflexive relation specifying the *mutex* arcs of $ENAM$. Where possible, we retain the definitions introduced for ENAM-systems. The notion of a step now changes however. A *step of* $ENAM$ is a non-empty set U of transitions such that U is a step of $\mathsf{und}(ENAM)$ and $Mtx \cap (U \times U) = \varnothing$. With this modified notion of a step, the remaining definitions pertaining to the dynamic aspects of an ENAM-system are the same as for the underlying ENA-system $\mathsf{und}(ENAM)$.

For the ENAM-system of Figure 16, we have $M[cm\rangle M'$ as well as $M[mc\rangle M'$, where $M = \{p_2, p_4, p_6\}$ and $M' = \{p_1, p_4, p_7\}$. However, $M[\{c, m\}\rangle M'$ which holds now for the underlying ENA-system does not hold as c and m cannot be executed in the same step.

Reachability graphs of ENAM-systems

Reachability in ENAM-systems, like in ENA-systems, is affected by the choice of the execution semantics. This is, however, entirely due to the presence of activator arcs, rather than mutex arcs. For an ENAM-system without any activator arcs, the same sets of markings are reachable under the step sequence and firing sequence semantics.

Another observation concerns the relationship between simultaneity and unorderedness in the behaviour of ENAM-systems. Whereas ENA-systems satisfy the relationship *Simultaneity* $\Longleftarrow$ *Unorderedness*, this no longer holds for ENAM-systems, as illustrated in Figure 17.

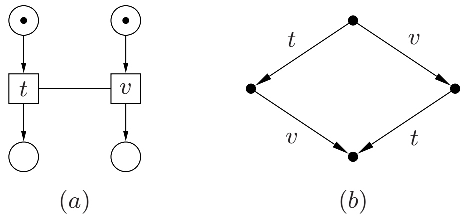


Fig. 17. An ENAM-system (without activator arcs) and its reachability graph.

Semantical framework for ENAM-systems

Causality semantics for ENAM-systems will be developed similarly as for EN-systems and ENA-systems. The labelled causal structures employed are GSO-structures, while executions remain to be step sequences. To define processes we extend AO-nets to include mutex arcs. An *activator mutex occurrence net* (or AMO-net) is a tuple $AMON = (P', T', F', Act, Mtx', \ell)$ such that $\mathsf{und}(AMON) = (P', T', F', Act, \ell)$ is the AO-net *underlying* $AMON$ and $Mtx' \subseteq T' \times T'$ is a symmetric irreflexive relation specifying a set of *mutex* arcs. Moreover, it is assumed that

$$\varrho_{AMON} = (T', \prec_{loc}, \sqsubset_{loc}, Mtx') \ ,$$

where $\prec_{loc}$ and $\sqsubset_{loc}$ are defined as for $\mathsf{und}(AMON)$, is a pre-GSO-structure. We then define $\mathsf{gsos}(AMON) = \varrho^{\mathsf{gso}}_{AMON}$ to be the GSO-structure induced by $AMON$. The step sequences $\mathsf{steps}(AMON)$ of $AMON$ are defined as for $\mathsf{und}(AMON)$, except that the enabling condition takes into account mutex arcs. The default initial and final markings of $AMON$, as well as its step sequence executions are defined as for $\mathsf{und}(AMON)$ under the proviso that steps do not contain transitions joined by mutex arcs.

The way ϱ_{AMON} deals with the mutex arcs is illustrated in Figure 12(b). We have there three transitions satisfying $t \sqsubset_{loc} v \sqsubset_{loc} w \sqsubset_{loc} t$. Hence, in any execution involving all these transitions, they have to belong to the same step. This, however, is inconsistent with a mutex arc between v and w, and ϱ_{AMON} fails to be a pre-GSO-structure as (t,t) belongs to $\sqsubset^*_{loc} \circ (Mtx' \cap \sqsubset^*_{loc}) \circ \sqsubset^*_{loc}$.

Processes of an ENAM-system are similar to those of the underlying ENA-system extended with appropriate treatment of mutex arcs. A *process* of $ENAM$ is an AMO-net $AMON$ such that $\mathsf{und}(AMON)$ is a process of $\mathsf{und}(ENAM)$ and, in addition, $Mtx' = \{(t,v) \mid (\ell(t),\ell(v)) \in Mtx\}$. We denote this by $AMON \in \mathsf{proc}(ENA)$.

Process generation from a given step sequence is also based on that introduced for EN-systems. The AO-net $\mathsf{proc}_{ENA}(\sigma)$ *generated* by a step sequence $\sigma = U_1 \dots U_n$ of $ENAM$ is the last element in the sequence $AMON_0, \dots, AMON_n$ where each $AMON_k = (P_k, T_k, F_k, A_k, M_k, \ell_k)$ is an AMO-net with the components constructed as in the definition for $\mathsf{proc}_{\mathsf{und}(ENAM)}(\sigma)$, except that

$$M_k = \{(e,f) \in T_k \times T_k \mid (\ell_k(e), \ell_k(f)) \in Mtx\} \ .$$

We denote this by $AMON_n \in \mathsf{proc}_{ENAM}(\sigma)$.

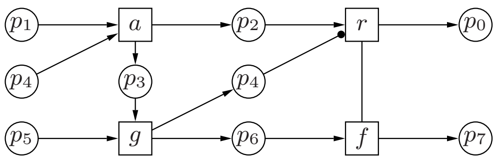


Fig. 18. An AMO-net $AMON_0$ with labels shown inside places and transitions.

Figure 18 depicts an AMO-net labelled with places and transitions of the ENAM-system of Figure 16. We have that both $agrf$ and $agfr$ belong to $\phi(\mathsf{steps}(AMON_0))$, however, $ag\{f,r\}$ does not. The AMON-net shown in Figure 18 is a process of the ENAM-system of Figure 16 with $\phi(\mathsf{steps}(AMON_0)) = \{agfr, agrf\}$. Figure 19 shows the result of applying the construction to the ENAM-system of Figure 19 and one of its step sequences. Note that the resulting AMO-net is isomorphic to that shown in Figure 18.

The way in which mutex arcs are added in the process construction entails means that some may be superfluous. For instance, the transitions they join may be causally related. Analysing paths in the AMO-net would make it possible to eliminate such redundant mutex arcs. This, however, would be against the locality principle which is fundamental to the process approach as it would compromise the local causes and effects in the definition and construction of process nets.

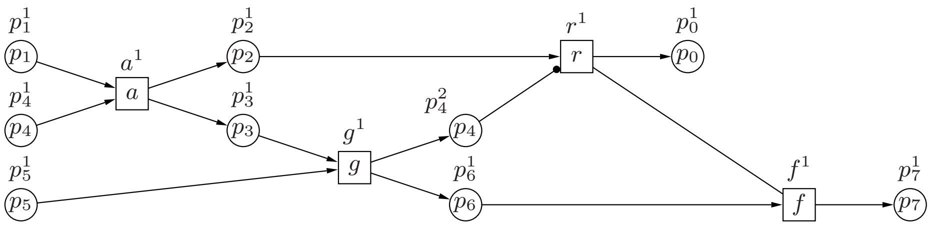


Fig. 19. Process $\mathsf{proc}_{ENAM_0}(\sigma)$ generated for $ENAM_0$ and step sequence $\sigma = \{a\}\{g\}\{r\}\{f\}$.

The semantical properties formulated above can be established also for ENAM-systems. Referring to the notation used in Figure 5, we have the following, where $ENAM$ is an ENAM-system, $AMON$ an AMO-net, (σ, ℓ) a labelled step sequence, $gsos$ a GSO-structure, and Σ a set of labelled singular step sequences with the same domain:

$\mathbb{PN}$	are	ENAM-systems	$\mathbb{EX}$	are	step sequences
$\mathbb{LAN}$	are	AMO-nets	$\mathbb{LEX}$	are	labelled singular step sequences
$\mathbb{LCS}$	are	labelled GSO-structures			
$\omega(ENAM)$	is	$\mathsf{steps}(ENAM)$	$\alpha(ENAM)$	is	$\mathsf{proc}(ENAM)$
$\lambda(AMON)$	is	$\mathsf{steps}(AMON)$	$\pi_{ENAM}(\sigma)$	is	$\mathsf{proc}_{ENAM}(\sigma)$
$\phi(\sigma, \ell)$	is	$\ell(\sigma)$	$\kappa(AMON)$	is	$\mathsf{gsos}(AMON)$
$\epsilon(gsos)$	is	$\mathsf{steps}(\mathsf{ext}(gsos))$	$\imath(\Sigma)$	is	$\bigcap \mathsf{gsos}(\mathsf{spo}(\Sigma))$.

It can be shown that Properties 1–4 hold. Below $ENAM$ is an ENAM-system and σ its step sequence, $AMON$ is an AMO-net, $gsos$ is an SO-structure, and Σ is a set of singular step sequences with the same domain. (Note that Fact 28 follows from Facts 5 and 10.)

Fact 26 $\mathsf{steps}(ENAM)$, $\mathsf{proc}(ENAM)$, $\mathsf{steps}(AMON)$ *and* $spo(gsos)$ *are non-empty sets. Moreover,* $\mathsf{gsos}(AMON)$ *and* $\bigcap \mathsf{gsos}(\mathsf{spo}(\Sigma))$ *are* GSO*-structures, and* $\mathsf{proc}_{ENAM}(\sigma)$ *is an* AMO*-net.*

Fact 27 $\mathsf{proc}_{ENAM}(\sigma)$ *is a process of* $ENAM$*. Moreover, if* $AMON$ *be a process of* $ENAM$ *and* $\sigma' \in \phi(\mathsf{steps}(AMON))$*, then* $\sigma' \in \mathsf{steps}(ENAM)$ *and* $AMON = \mathsf{proc}_{ENAM}(\sigma')$.

Fact 28 $gsos = \bigcap \mathsf{gsos}(\mathsf{steps}(\mathsf{ext}(gsos)))$.

Fact 29 $\mathsf{steps}(AMON) = \mathsf{ext}(\mathsf{gsos}(AMON))$.

Hence we can claim the semantical aims for ENAM-systems.

Fact 30 *Let $ENAM$ be an* ENAM*-system, and $AMON$ be an* AMO*-net.*

$$\begin{aligned} \mathsf{proc}(ENAM) &= \mathsf{proc}_{ENAM}(\mathsf{steps}(ENAM)) \\ \mathsf{steps}(ENAM) &= \phi(\mathsf{steps}(\mathsf{proc}(ENAM))) \\ \mathsf{gsos}(AMON) &= \bigcap \mathsf{gsos}(\mathsf{steps}(AMON)) \ . \end{aligned}$$

9 Place/Transition nets

Place/Transition nets [25] (or PT-nets) are the basic class of Petri nets suited for the study of systems in which multiplicity of resources matters.

A PT-*net* is a tuple $PT = (P, T, F, M_{init})$ such that (P, T, F) is its *underlying* net, and M_{init} is the *initial* marking of PT, where a marking in this case is any *multiset* of places, i.e., a mapping $M : P \to \mathbb{N}$. Most notions concerning the structure and graphical representation of PT-nets are the same as for EN-systems except that a marking M is represented by displaying $M(p)$ tokens in each place p. More important changes concern the execution semantics which extends that defined for EN-systems.

A *step* U of PT is any *multiset* of transitions, i.e., $U : T \to \mathbb{N}$. Such a step is *enabled* at a marking M if, for every place p, the current marking M provides enough input tokens for each occurrence of a transition in U, thus $M(p) \geq \sum_{t \in p^\bullet} U(t)$. Executing an enabled step leads to the marking M' such that, for every place p,

$$M'(p) = M(p) - \sum_{t \in p^\bullet} U(t) + \sum_{t \in {}^\bullet p} U(t) \ .$$

We denote this, as before, by $M[U\rangle M'$ or $M[U\rangle_{PT} M'$. The notions of firing sequence, step sequence, marking reachability and reachability graph, are then defined similarly as in the case of EN-systems. Figure 20 depicts three PT-nets such that:

$$\begin{aligned} &\mathsf{fseq}(PT_0) = \{\ldots, amamamam, \ldots\} \\ &M_{init}[gu\{g,a\}_{PT_1}\{u,m\}am\rangle M_{init} \\ &\mathsf{steps}(PT_2) = \{\ldots, a\{g,g\}mama\{u,u\}\{g,g\}, \ldots\} \ . \end{aligned}$$

As in the case of EN-systems, marking reachability in PT-nets does not depend on whether one uses firing sequences or step sequences. This follows from the fact that if U and U' are two steps satisfying $M[U + U'\rangle M'$ then $M[UU'\rangle M'$, where $U + U'$ is the multiset sum of U and U'. As a consequence, every step of transitions occurring at a marking can be split into any sequence of subsets forming a partition of this set, and each such step sequence leads to the same marking as the original step. However, the reverse implication does not, in general, hold. For example, if one takes the PT-net in Figure 23(a), then we have $M_{init}[ab\rangle\{p_2, p_4\}$ and $M_{init}[ba\rangle\{p_2, p_4\}$ but $M_{init}[\{a, b\}\rangle\{p_2, p_4\}$ is not a valid execution. Moreover, the relation between transition occurrences is not structural, but depends on the current marking: with two tokens in p_5 in Figure 23(a), the transitions a and b would be concurrently, i.e., as a step, enabled.

As before, processes formalise the idea of a concurrent run. Interestingly, occurrence nets provide the basis for the process definition of PT-nets in the same way as

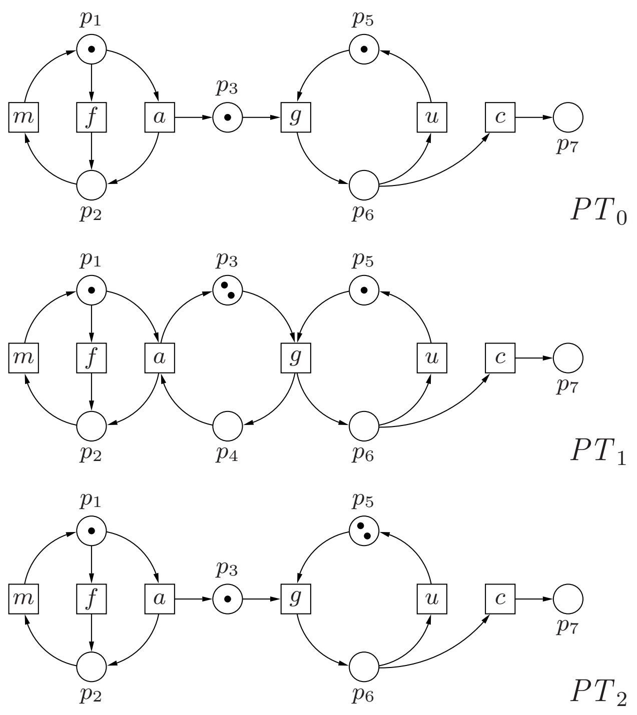


Fig. 20. PT-nets modelling three final versions of the producer/consumer system: PT_0 with an unbounded buffer (the number of tokens in place p_3 can grow unboundedly); PT_1 with a two-place buffer (the number of tokens in place p_3 can be at most two); and PT_2 with an unbounded buffer and two consumers (represented by the two tokens in place p_5).

they did for EN-systems. We only need to take into account the potential multiplicity of tokens in PT-nets. This is done by giving each occurrence of a token its own place in the occurrence nets. A *process* of a PT-net PT is an occurrence net ON with labelling ℓ which:

– labels places of ON with places of PT.
– labels transitions of ON with transitions of PT.
– labels exactly $M_{init}(p)$ places of M^{ON}_{init} with p, for every place p of PT.
– is injective on ${}^\bullet t$ and $t^\bullet$ and, moreover, $\ell({}^\bullet t) = {}^\bullet\ell(t)$ and $\ell(t^\bullet) = \ell(t)^\bullet$, for every transition t of ON.

We denote this by $ON \in \mathsf{proc}(PT)$. The occurrence net ON in Figure 21 is a process of PT-net PT_2 in Figure 20.

The main difference with definition of processes of EN-systems is that now the labelling of a process is not required to be injective on the default initial marking which is meant to represent the initial marking. In general, Fact 14 does not hold for processes of PT-nets. For example, the process in Figure 21 allows the following sequence of executions:

$$\{q_1, q_2, q_3, q_4\}[t_1\rangle\{q_2, q_3, q_4, q_5, q_6\}[\{t_2, t_3\}\rangle\{q_5, q_7, q_8\}\ ,$$

with $\ell(q_7) = \ell(q_8) = p_6$ and $\ell(t_2) = \ell(t_3) = g$.

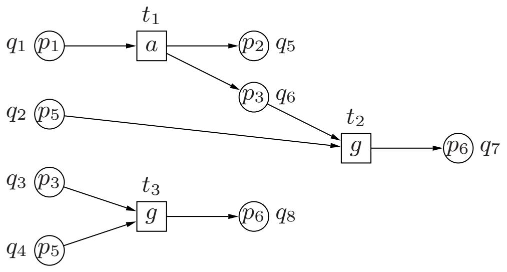


Fig. 21. A process ON of the PT-net PT_2 in Figure 20.

Defining a process for a given step sequence σ of a PT-net PT is also a straightforward extension of the construction for EN-systems. An occurrence net *generated* by a step sequence $\sigma = U_1 \dots U_n$ of PT is the last element in the sequence $ON_0, \dots, ON_n$ where each ON_k is an occurrence net (P_k, T_k, F_k, ℓ_k) constructed in the following way.
Step 0: $P_0 = \{p^i \mid p \in P \wedge 1 \leq i \leq M_{init}(p)\}$ and $T_0 = F_0 = \varnothing$.
Step k: Given ON_{k-1} we extend the sets of nodes as follows:

$$\begin{aligned} P_k &= P_{k-1} \cup \{p^{i+\triangle p} \mid p \in P \wedge 1 \leq i \leq \textstyle\sum_{t \in {}^\bullet p} U_k(t)\} \\ T_k &= T_{k-1} \cup \{t^{i+\triangle t} \mid t \in T \wedge 1 \leq i \leq U_k(t)\} \ . \end{aligned}$$

Again, the label of each node x^i is set to be x, and $\triangle x$ denotes the number of the nodes of ON_{k-1} labelled by x.
To define the arcs, we proceed as follows. For every $e = t^i \in T_k \setminus T_{k-1}$, we *choose* two sets of conditions, $In_e \subseteq M_{fin}^{ON_{k-1}}$ and $Out_e \subseteq P_k \setminus P_{k-1}$, such that In_e comprises a distinct condition p^m for each place $p \in {}^\bullet t$ while Out_e comprises a distinct condition q^l for each place $q \in t^\bullet$. Moreover, for any $e \neq f \in T_k \setminus T_{k-1}$, $In_e \cap In_f = \varnothing$ and $Out_e \cap Out_f = \varnothing$. Then:

$$F_k = F_{k-1} \cup \bigcup_{e \in T_k \setminus T_{k-1}} (In_e \times \{e\}) \cup (\{e\} \times Out_e) \ .$$

We denote this by $ON_n \in \mathsf{proc}_{PT}(\sigma)$.

Note that since there may be more than one choice of suitable In_e's, in general, more than one process can be constructed for a given step sequence σ. The above construction is illustrated in Figure 22 for PT-net PT_2 of Figure 20. The resulting occurrence net is isomorphic to ON of Figure 6 which, as we already noted, is a process of PT_2.

The detailed development of the process semantics of PT-nets can be carried out along the same lines as was done for EN-systems earlier in this paper, with some straightforward modification resulting from the multiset — rather than set — nature of markings and executed steps. It is also possible to extend the treatment of PT-nets to include weighted arcs and (weighted) activator and inhibitor nets, using AO-nets as a process model, following what was done for ENA-systems and ENI-systems in, e.g., in [19, 20].

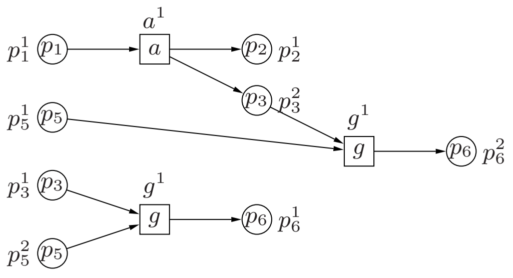


Fig. 22. Deriving a process for PT_2 and its step sequence $\sigma = \{a, g\}g$.

Mutex arcs and self-loops

In PT-nets, in contrast to EN-systems, mutex arcs can be represented by self-loops connected to a place marked with a single token, as shown in Figure 23(a, b). From a modelling perspective, there appears to be no real difference. Semantically, however, the differences can be significant as mutex arcs represent concurrent histories in a more compact way. This could have an impact when net unfoldings are used for model checking. For example, the single process in Figure 23(c) derived for the representation of Figure 23(b) has to be replaced by two processes derived for the representation of Figure 23(a) depicted in Figure 23(d).

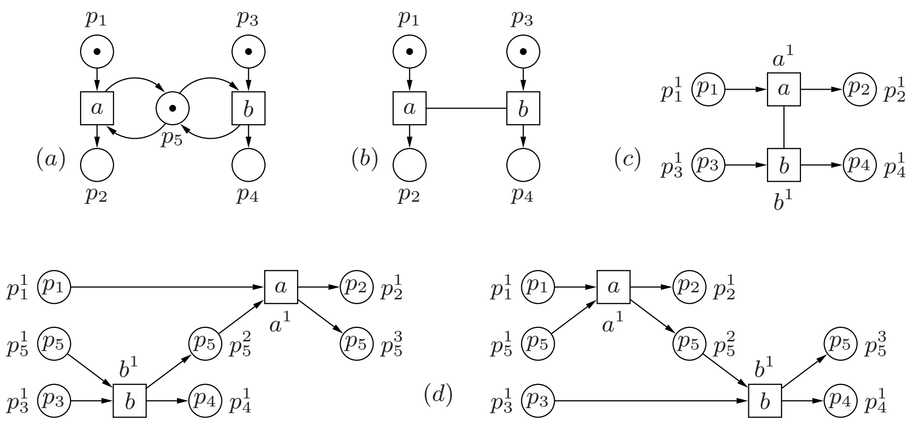


Fig. 23. Mutex arcs can lead to more condensed process semantics than self-loops.

10 Concluding remarks

This paper is an introduction to the many issues fundamental to understanding concurrent behaviour. Here we have concerned ourselves with different forms of causality induced by extensions to the basic structure of Petri nets and leading to relational structures extending the classical partial order approach. There are several strands of related research which have not been described here. For instance, we have not considered the modelling of conflicts between enabled transitions. Our processes and their abstractions (partial orders) model concurrent runs in which conflicts have already been resolved. Branching processes of Petri nets [3] model all possible choices and lead to a single unfolding representing all runs of the net model. They are actually the basis for efficient verification techniques [5, 18, 23]. If, in addition, one abstracts from state information and only considers relations between events the result is the more abstract model of event structures [9, 24, 28], that can be used to study fundamental concepts of concurrency in a model-independent way. As far as we are aware, event structures have not yet been enriched with weak causality and commutativity relationships, and we consider such extensions a relevant, and indeed exciting, topic of future research in this area.

Finally, an abstraction not considered here at all, usually referred to as trace theory [2] initiated in [22], allows one to group together sequential observations on the basis of reordering of concurrent (independent) events. The resulting model of *trace monoid* captures precisely the semantical treatment of EN-systems outlined in this paper. For the extended models of ENI/ENA-systems, one needs to use the extended model of *comtraces* introduced in [14]. The last extension of EN-systems considered here, i.e., ENAM-systems, calls for the even more elaborate model of *generalised comtraces* [16]. It should then not come as a surprise that PT-nets require a different kind of extensions of the basic trace monoid, initiated through the work on local traces of [8]. An extensive account of the intrinsic relationships between various concurrency monoids and different net classes can be found in [11].

References

1. Best, E., Devillers, R.: Sequential and Concurrent Behaviour in Petri Net Theory. Theoretical Computer Science **55** (1987) 87–136
2. Diekert, V., and Rozenberg, G., (eds.): The Book of Traces. World Scientific, Singapore (1995)
3. Engelfriet, J.: Branching Processes of Petri Nets. Acta Informatica **28** (1991) 575–591
4. Esparza, J., Heljanko, K.: Unfoldings: a Partial Order Approach to Model Checking. Springer, Monographs in Theoretical Computer Science. (2008)
5. Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. Lecture Notes in Computer Science **1055** (1996) 87–106
6. Gaifman, H., Pratt, V.R.: Partial Order Models of Concurrency and the Computation of Functions. In: LICS, IEEE Computer Society (1987) 72–85
7. Guo, G., Janicki, R.: Modelling Concurrent Behaviours by Commutativity and Weak Causality Relations. Lecture Notes in Computer Science **2422** (2002) 178–191
8. Hoogers, P.W., Kleijn, H.C.M., Thiagarajan, P.S.: A Trace Semantics for Petri Nets. Information and Computation **117** (1995) 98–114
9. Hoogers, P.W., Kleijn, H.C.M., Thiagarajan, P.S.: An Event Structure Semantics for General Petri Nets. Theoretical Computer Science **153** (1996) 129–170
10. Janicki, R.: Relational Structures Model of Concurrency. Acta Informatica **45** (2008) 279–320

11. Janicki, R., Koutny, M., Kleijn, J.: Quotient Monoids and Concurrent Behaviours. In: Scientific Applications of Language Methods, Carlos Martín-Vide (ed.). Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory **2**, World Scientific (2010)
12. Janicki, R., Koutny, M.: Invariants and Paradigms of Concurrency Theory. Lecture Notes in Computer Science **506** (1991) 59–74.
13. Janicki, R., Koutny, M.: Structure of Concurrency. Theoretical Computer Science **112** (1993) 5–52
14. Janicki, R., Koutny, M.: Semantics of Inhibitor Nets. Information and Computation **123** (1995) 1–16
15. Janicki, R., Koutny, M.: Order Structures and Generalisations of Szpilrajn's Theorem. Acta Informatica **34** (1997) 367–388
16. Janicki, R., Le, D.T.M.: Modelling Concurrency with Comtraces and Generalized Comtraces. Information and Computation (to appear)
17. Juhás, G., Lorenz, R., Mauser, S.: Complete Process Semantics of Petri Nets. Fundamenta Informaticae **87** (2008) 331–365
18. Khomenko, V., Koutny, M., Vogler, W.: Canonical Prefixes of Petri Net Unfoldings. Acta Informatica **40** (2003) 95–118
19. Kleijn, H.C.M., Koutny, M. Process Semantics of General Inhibitor Nets. Information and Computation **190** (2004) 18–69
20. Kleijn, J., Koutny, M.: Processes of Petri Nets with Range Testing. Fundamenta Informaticae **80** (2007) 199–219
21. Kleijn, J., Koutny, M.: The Mutex Paradigm of Concurrency. To appear in Petri Nets 2011, Lecture Notes in Computer Science (2011)
22. Mazurkiewicz, A.: Concurrent Program Schemes and Their Interpretations. DAIMI Report PB 78, Aarhus University, Aarhus (1977)
23. McMillan, K.L.: Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits. Lecture Notes in Computer Science **663** 164–177
24. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri Nets, Event Structures and Domains, Part I. Theoretical Computer Science **13** (1981) 85–108
25. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets. Lecture Notes in Computer Science **1491,1492** (1998)
26. Rozenberg, G., Engelfriet, J.: Elementary Net Systems. In: Part I of [25] (1998) 12–121
27. Szpilrajn, E.: Sur l'extension de l'ordre partiel. Fundamenta Mathematicae **16** (1930) 386–389
28. Winskel, G.: Event Structures. Lecture Notes in Computer Science **255** (1986) 325–392