Mutex Causality in Processes and Traces of General Elementary Nets

Jetty Kleijn and Maciej Koutny
 Mutex Causality in Processes and Traces of General Elementary Nets

J. Kleijn, M. Koutny

Abstract

A concurrent history represented by a causality structure that captures the intrinsic, invariant dependencies between its actions, can be interpreted as defining a set of closely related observations (e.g. step sequences). Depending on the relationships observed in the histories of a system, the concurrency paradigm to which it adheres may be identified, with different concurrency paradigms underpinned by different kinds of causality structures. Adding mutex arcs to elementary net systems with inhibitor arcs yields a system model enim-systems) that through its process semantics and associated causality structures fits the least restrictive concurrency paradigm. Here we complete the picture by giving an abstract description of the behaviour of an enim-system by grouping together step sequences in equivalence classes (generalised comtraces) using the structural relations between its transitions. The thus defined concurrent histories of the enim-system correspond exactly to the generalised stratified order structures underlying its processes. The results presented establish a link between enim-systems and trace theory and allow one to identify different observations of concurrent behaviour in a way that is consistent with the causality semantics defined by the operationally defined processes.
Abstract

A concurrent history represented by a causality structure that captures the intrinsic, invariant dependencies between its actions, can be interpreted as defining a set of closely related observations (e.g. step sequences). Depending on the relationships observed in the histories of a system, the concurrency paradigm to which it adheres may be identified, with different concurrency paradigms underpinned by different kinds of causality structures. Adding mutex arcs to elementary net systems with inhibitor arcs yields a system model enim-systems) that through its process semantics and associated causality structures fits the least restrictive concurrency paradigm. Here we complete the picture by giving an abstract description of the behaviour of an enim-system by grouping together step sequences in equivalence classes (generalised comtraces) using the structural relations between its transitions. The thus defined concurrent histories of the enim-system correspond exactly to the generalised stratified order structures underlying its processes. The results presented establish a link between enim-systems and trace theory and allow one to identify different observations of concurrent behaviour in a way that is consistent with the causality semantics defined by the operationally defined processes.

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

CONCURRENCY PARADIGM
ELEMENTARY NET SYSTEM
INHIBITOR ARC
MUTEX ARC
STEP SEQUENCE
CAUSALITY
GENERALISED STRATIFIED ORDER STRUCTURE
PROCESS SEMANTICS
GENERALISED COMTRACE
Mutex Causality in Processes and Traces of General Elementary Nets

Jetty Kleijn¹ and Maciej Koutny²

¹ LIACS, Leiden University
P.O.Box 9512, NL-2300 RA Leiden, The Netherlands
kleijn@liacs.nl

² School of Computing Science, Newcastle University
Newcastle upon Tyne NE1 7RU, U.K.
maciej.koutny@ncl.ac.uk

Abstract. A concurrent history represented by a causality structure that captures the intrinsic, invariant dependencies between its actions, can be interpreted as defining a set of closely related observations (e.g., step sequences). Depending on the relationships observed in the histories of a system, the concurrency paradigm to which it adheres may be identified, with different concurrency paradigms underpinned by different kinds of causality structures. Adding mutex arcs to elementary net systems with inhibitor arcs yields a system model (ENIM-systems) that through its process semantics and associated causality structures fits the least restrictive concurrency paradigm. Here we complete the picture by giving an abstract description of the behaviour of an ENIM-system by grouping together step sequences in equivalence classes (generalised comtraces) using the structural relations between its transitions. The thus defined concurrent histories of the ENIM-system correspond exactly to the generalised stratified order structures underlying its processes. The results presented establish a link between ENIM-systems and trace theory and allow one to identify different observations of concurrent behaviour in a way that is consistent with the causality semantics defined by the operationally defined processes.

Keywords: concurrency paradigm, elementary net system, inhibitor arc, mutex arc, step sequence, causality, generalised stratified order structure, process semantics, generalised comtrace.

1 Introduction

A run of a concurrent system may be observed and recorded in various ways. Sequential descriptions like sequences (or even step sequences) are not always expressive enough when it comes to giving faithful information on causality and independence of actions executed in a concurrent run. Traces as introduced by Mazurkiewicz [24, 4] are an example of how explicit information can be provided on the essential causal dependencies between executed actions. The order in which the executions of independent actions are observed is accidental and sequences which only differ in the order of occurrences of independent actions are observations of the same concurrent run. Hence these sequences may be identified yielding an equivalence class (a trace) of sequential observations of this run. In particular, given an alphabet of actions and an independence relation
between them, traces form a quotient monoid. Moreover, by explicitly specifying the order of dependencies, a unique partial order can be associated to each trace and each trace comprises all sequences consistent with some causal partial order on the executed actions [7]. A system model fitting well with the trace approach are elementary net systems (EN-systems), as demonstrated by the fact that the causal partial orders derived from the process semantics of an EN-system coincide with the partial orders defined by traces based on an independence relation between transitions obtained from the underlying graph structure of the system (see, e.g., [26, 19]).

So, concurrency can be studied at different levels of abstraction, from the lowest level dealing with individual behavioural runs (observations), to the intermediate level of more abstract concurrent histories (combining closely related observations) which can be represented by causality (order) structures capturing the intrinsic, invariant dependencies between executed actions, to the highest system level dealing with devices such as Petri nets or process algebra expressions. Clearly, different descriptions of concurrent systems and their behaviours at these distinct levels of abstractions must be consistent and their mutual relationships well understood.

In this paper, observations will be step sequences, i.e., sequences of finite sets (steps) of simultaneously executed actions. As an example consider the EN-system depicted in Figure 1(a). This system generates three step sequences involving the executions of transitions \( a, c \) and \( d \), viz. \( \sigma_1 = \{a\} \{c, d\} \), \( \sigma_2 = \{a\} \{c\} \{d\} \) and \( \sigma_3 = \{a\} \{d\} \{c\} \). They can be seen as belonging to a single abstract history \( \Delta_1 = \{\sigma_1, \sigma_2, \sigma_3\} \) underpinned by a causal partial order in which \( c \) and \( d \) are unordered and both depend on (are preceded by) \( a \). This \( \Delta_1 \) adheres to the true concurrency paradigm (here expressed for step sequences) captured by the following general statement:

Given two executed actions (e.g., \( c \) and \( d \) in \( \Delta_1 \)), they can be observed as simultaneous (e.g., in \( \sigma_1 \) \( \iff \) they can be observed in both orders (e.g., \( c \) before \( d \) in \( \sigma_2 \), and \( d \) before \( c \) in \( \sigma_3 \)).

\( \text{(TRUECON)} \)
Concurrent histories adhering to this paradigm are underpinned by causal partial orders, in the sense that each such history comprises all step sequences consistent with some causal partial order on executed actions. Elementary net systems with their step semantics provide a natural system level model for the true concurrency paradigm. A suitable link between an EN-system and histories like $\Delta_1$ can be formalised using the notion of a process or occurrence net [1, 26]. Full consistency between the three levels of abstraction can then be established within the generic approach of the semantical framework of [18] aimed at fitting together systems (nets from a certain class of Petri nets), abstract histories and individual observations. On the other hand, as shown e.g., in [14], the equivalence classes of step sequences of an EN-system defined using structural information about the dependencies between its transitions yield a (trace based) partial order description of its behaviour that coincides with the partial order semantics of nets represented by the non-sequential observations captured by its operationally defined processes.

Depending on the exact nature of relationships holding for actions executed in a single concurrent history, similar to (TRUE) recalled above, [11] identified eight general concurrency paradigms, $\pi_1$–$\pi_8$, with ‘true concurrency’ being another name for $\pi_8$. Another paradigm is $\pi_3$ characterised by (TRUE) with $\iff$ replaced by $\leftarrow$. This paradigm has a natural system level counterpart provided by elementary net systems with inhibitor arcs (ENI-systems). Note that inhibitor arcs (as well as activator arcs used later in this paper) are well suited to model situations involving testing for a specific condition, rather than producing and consuming resources, and proved to be useful in areas such as communication protocols [2], performance analysis [5] and concurrent programming [6].

For example, Figure 1(b) depicts an ENI-system generating two step sequences involving transitions $a$, $c$ and $d$, viz. $\sigma_1 = \{a\} \{c, d\}$ and $\sigma_2 = \{a\} \{c\} \{d\}$. The two step sequences can be seen as belonging to the abstract history $\Delta_2 = \{\sigma_1, \sigma_2\}$ adhering to paradigm $\pi_3$, but not adhering to paradigm $\pi_8$ as there is no step sequence in $\Delta_2$ in which $c$ is observed before $d$ (even though $c$ and $d$ are observed in $\sigma_1$ as simultaneous). Another consequence of the latter fact is that paradigm $\pi_3$ histories are underpinned not by causal partial orders but rather by causality structures introduced in [12] — called stratified order structures — based on causal partial orders and, in addition, weak causal partial orders. Again, full consistency between the three levels of abstraction can then be established within the semantical framework of [18]. In [12], comtraces (combined traces) have been introduced as an extension of traces to take into account the ‘not later than’ relationship between executed transitions of ENI-systems. Moreover, similar to the relation between traces and partial orders, comtraces correspond to stratified order structures and provide an additional trace based approach to the causality semantics of ENI-systems [19, 22]. Again, information about the dependencies between transitions can be obtained from the graph structure of the net.

In this paper, we focus on $\pi_1$ which simply admits all concurrent histories and is the least restrictive of the eight general paradigms of concurrency investigated in [11]. Concurrent histories conforming to paradigm $\pi_1$ are underpinned by yet another kind of causality structures introduced in [11] — called generalised stratified order structures — based on weak causal partial orders and commutativity. Intuitively, two executed
actions commute if they may be observed in any order in step sequences belonging to a
history, but they are never observed as simultaneous.

Until recently, a system level net model matching paradigm $\pi_1$ was missing. In [14] it
is proposed to add mutex arcs to ENI-systems where mutex arcs relate pairs of transi-
tions which cannot be executed simultaneously, even when they can be executed in any
order. Mutex arcs are therefore a system level device implementing commutativity (for
an early attempt aimed at capturing such a feature see [23]). In [20] it is shown that the
resulting ENIM-systems provide a natural match for histories conforming to paradigm
$\pi_1$, in the same way as EN-systems and ENI-systems provided a natural match for his-
tories conforming to paradigms $\pi_8$ and $\pi_3$, respectively. Thus, one could argue that
ENIM-systems are the most general elementary net systems.

Figure 1(c) depicts an ENIM-system generating two step sequences involving transi-
tions $a$, $c$ and $d$, viz. $\sigma_2 = \{a\} \{c\} \{d\}$ and $\sigma_3 = \{a\} \{d\} \{c\}$. They belong to an abstract
history $\Delta_3 = \{\sigma_2, \sigma_3\}$ adhering to paradigm $\pi_1$, in which the executions of $c$ and $d$
commute. Clearly, $\Delta_3$ does not conform to paradigms $\pi_8$ and $\pi_3$ as there is no step
sequence in $\Delta_3$ in which $c$ and $d$ are observed as simultaneous. To prove full consis-
tency between the three levels of abstraction for paradigm $\pi_1$ the semantical framework
of [18] can be used once more. In doing so, processes of ENIM-systems are defined
and it is demonstrated that these new processes provide the desired link with the gen-
eralised stratified order structures of paradigm $\pi_1$. To achieve this, a notion of gso-closure
allows one to construct generalised stratified order structures from more basic relation-
ships between executed actions involved in processes of ENIM-systems.

Our aim in this paper is to provide the full picture of the causality semantics of
ENIM-systems with all necessary technical details. In [15,16], generalised comtraces
have been introduced as extensions of comtraces that can additionally model the non-
simultaneous relationship. Moreover, it has been shown that generalised comtraces can
be represented by generalised stratified order structures. Here we show how generalised
comtraces can be used to identify step sequences of ENIM-systems based on struc-
tural relations between transitions. We demonstrate that the thus defined trace based
behaviour agrees with the process semantics defined in [20], and we prove that the gen-
eralised stratified order structure underlying a process coincides with the generalised
stratified order structure underlying its associated generalised comtrace.

The paper is organised in the following way. In the next section, we recall key no-
tions and notations used throughout the paper. In Section 3, we outline the way in which
paradigms of concurrency and the corresponding order structures can be defined. Sec-
tion 4 describes generalised comtrace and their relationship to GSO-structures. In Sec-
tion 5, we recall the semantical framework of [18]. Section 6 deals with ENIM-systems
and presents their process semantics. Section 7 shows how to express the behaviour of
an ENIM-system in terms of G-comtraces.

This paper is a companion paper to the conference publication [20] which it extends
and completes by providing a treatment of general concurrent histories of ENIM-systems
in the framework of trace theory. We recall observations and results (without proofs)
from related work to sketch the overall picture and include results from [20]; the latter
sometimes with proofs if they might contribute to a better understanding of the overall
approach.
2 Preliminaries

In this paper we use mostly standard mathematical notation.

**Functions and relations** The composition of two functions \( f : X \rightarrow 2^Y \) and \( g : Y \rightarrow 2^Z \) is defined by \( g \circ f(x) = \bigcup_{y \in f(x)} g(y) \), for all \( x \in X \). Restricting a function \( f \) to a sub-domain \( Z \) is denoted by \( f|_Z \). The composition of two binary relations \( Q \subseteq X \times Y \) and \( R \subseteq Y \times Z \) is a binary relation \( Q \circ R \subseteq X \times Z \) defined by \( Q \circ R = \{(x, z) \mid \exists y \in Y \text{ such that } (x, y) \in Q \text{ and } (y, z) \in R\} \). A relation \( P \subseteq X \times X \) is irreflexive if \( (x, x) \notin P \) for all \( x \in X \); transitive if \( P \circ P \subseteq P \); its transitive and reflexive closure is denoted by \( P^* \); and its symmetric closure by \( P_{sym} = P \cup P^{-1} \). If \( \sim \) is an equivalence relation on \( X \), then \( X/\sim \) is the set of equivalence classes of \( \sim \) and, for each \( x \in X \), \([x]_\sim \) (or simply \([x]_\sim \)) is the equivalence class of \( \sim \) containing \( x \).

**Relational structures** A relational structure is a tuple \( R = (X, Q_1, \ldots, Q_n) \) where \( X \) is a finite domain, and the \( Q_i \)'s are binary relations on \( X \) (we can select its components using the subscript \( R \), e.g., \( X_R \)). Relational structures, \( R \) and \( R' \), are isomorphic if there is a bijection \( \xi \) from the domain of \( R \) to the domain of \( R' \) such that if we replace throughout \( R \) each element \( a \) by \( \xi(a) \) then the result is \( R' \). For relational structures with the same domain and arity, \( R \) and \( R' \), we write \( R \subseteq R' \) if the subset inclusion holds component-wise. The intersection \( \bigcap \mathcal{R} \) of a non-empty set \( \mathcal{R} \) of relational structures with the same arity and domain is defined component-wise. In this paper, we assume that all relational structures, i.e., their domains, are labelled, with the identity function as default labelling. If the labelling is irrelevant for a definition or result, it may be omitted. If two domains are said to be the same, their labellings are identical.

**Posets** A partially ordered set (or poset) is a relational structure \( po = (X, \prec) \) consisting of a finite set \( X \) and a transitive irreflexive relation \( \prec \) on \( X \). Two distinct elements \( a, b \) of \( X \) are unordered, \( a \sim b \), if neither \( a \prec b \) nor \( b \prec a \) holds. Moreover, \( a \preceq b \) if \( a \sim b \) or \( a \prec b \). Poset \( po \) is total if the relation \( \sim \) is empty, and stratified if \( \simeq \) is an equivalence relation, where \( a \simeq b \) if \( a \sim b \) or \( a = b \). Note that if a poset is interpreted as an observation of concurrent system behaviour, then \( a \prec b \) means that \( a \) was observed before \( b \), while \( a \simeq b \) means that \( a \) and \( b \) were observed as simultaneous.

**Posets and step sequences** In general, a step sequence \( \sigma = X_1 \ldots X_k \) is a finite sequence of finite non-empty sets. It is called singular, if the steps \( X_i \) are mutually disjoint. If \( \sigma \) is singular, then \( spo(\sigma) = \bigcup_i X_i \) is a stratified poset. Conversely, each stratified poset \( spo \) induces a unique singular step sequence \( steps(spo) = X_1 \ldots X_k \) with each \( X_i \) being an equivalence class of \( \sim \) and \( (X_i \times X_j) \subseteq \preceq \) for all \( i < j \), satisfying \( spo = spo(steps(spo)) \). We will identify each stratified poset \( spo \) with \( steps(spo) \) or, equivalently, each singular step sequence \( \sigma \) with \( spo(\sigma) \).

Given a (not necessarily singular) step sequence \( \sigma = X_1 \ldots X_k \), we denote by \( occ(\sigma) \) the set of all indexed symbols \( x^i \) such that \( x \) occurs in \( \sigma \) and \( i \geq 1 \) does not exceed the total number of occurrences of \( x \) within \( \sigma \) (we also denote by \( pos_\sigma(x^i) \) the position of \( x^i \) defined as the smallest \( j \) such that the number of occurrences of \( x \) in \( X_1 \ldots X_j \) is precisely \( i \)). Such a step sequence also induces a stratified poset but over
the domain $occ(\sigma)$. More precisely, we use $spo^{occ}(\sigma)$ to denote the poset $(occ(\sigma), \prec_{\sigma})$ such that $x^i \prec_{\sigma} y^j$ whenever $pos_{\sigma}(x^i) < pos_{\sigma}(y^j)$. Moreover, for any stratified poset $spo$ with the domain $occ(\sigma)$, we use $steps^{occ}(spo)$ to denote the step sequence obtained from $steps(spo)$ by replacing each $x^i$ with $x$. Note that $\sigma = steps^{occ}(spo^{occ}(\sigma))$.

Monoids The basis of the algebraic approach to the description of systems’ behaviours is the concept of a monoid. A monoid is a triple $\mathcal{M} = (X, \circ, 1)$, where $X$ is a (possibly infinite) set, $\circ$ is an associative binary operation on $X$, and $1$ is an element of $X$ such that $x \circ 1 = 1 \circ x = x$, for each $x \in X$.

Let $E$ be a non-empty, finite set of action names. Then $(E^*, \circ, 1)$ is a monoid, where $E^*$ consists of all finite sequences of action names, $\circ$ is the sequence concatenation operation, and $1$ is the empty sequence.

A step over $E$ is a non-empty subset of $E$, and a step sequence over $E$ is a finite sequence of steps. The empty step sequence is again denoted by $1$.

Let $\mathcal{S}$ be a step alphabet over $E$, i.e., $\mathcal{S}$ is a non-empty set of steps over $E$. Then $\mathcal{S}^*$ consists of all sequences of steps over $E$, and $(\mathcal{S}^*, \circ, 1)$, with $\circ$ step sequence concatenation, is a monoid of step sequences (over $\mathcal{S}$). Since the elements of $\mathcal{S}$ are sets, one can — in addition to concatenation — use the standard set theoretic relationships and operators to manipulate them.

Quotient monoids To introduce structure to the otherwise plain sets of sequences or step sequences that constitute all observations of a system, (step) sequences are grouped into clusters of equivalent observations. A congruence in the monoid $\mathcal{M} = (X, \circ, 1)$ is an equivalence relation $\sim$ on $X$ such that $a \sim b$ and $c \sim d$ implies $a \circ c \sim b \circ d$, for all $a, b, c, d \in X$. In such a case, $\mathcal{M}_\sim = (X/\sim, \hat{\circ}, \{1\})$ with $\hat{[a]} \hat{\circ} \hat{[b]} = \hat{[a \circ b]}$, for all $a, b \in X$, is the quotient monoid of $\mathcal{M}$ w.r.t. congruence $\sim$. Note that because $\sim$ is a congruence rather than a mere equivalence, $\hat{\circ}$ is a well-defined operation, and $\mathcal{M}_\sim$ is indeed a monoid. Quotient monoids defined by congruence relations provide a convenient way of introducing algebraic structure to observations of concurrent systems, in particular, when the congruences are induced by equations.

Let $EQ = \{ x_1 = y_1 \ldots x_n = y_n \}$, where $x_i, y_i \in X$ for $1 \leq i \leq n$, be a finite set of equations over $\mathcal{M}$. The congruence defined by $EQ$, denoted by $\equiv_{EQ}$ (or simply $\equiv$), is the least congruence $\equiv$ in $\mathcal{M}$ such that $x_i \equiv y_i$, for every $i \leq n$. Actually, $\equiv_{EQ}$ can be defined in a more constructive way using the binary relation $\approx_{EQ}$, comprising all pairs $(u \circ x, w, u \circ y, w) \in X \times X$, where $u, w \in X$ and $1 \leq i \leq n$. One can then show (see, e.g., [14]) that $\equiv_{EQ}$ is the reflexive, symmetric, and transitive closure of $\approx_{EQ}$.

Example 1 (Mazurkiewicz traces). Consider $(E^*, \circ, 1)$ for some non-empty, finite set of action names $E$, and equations:

$$EQ = \{ a_1 \circ b_1 = b_1 \circ a_1 \ldots a_n \circ b_n = b_n \circ a_n \}$$

where $a_1, \ldots, a_n, b_1, \ldots, b_n \in E$ and $a_i \neq b_i$ for all $i$. The elements of $\mathcal{M}_{\equiv_{EQ}} = (E^*/\equiv_{EQ}, \hat{\circ}, \{1\})$ are called Mazurkiewicz traces [24] or simply traces. (For the rest of the example we omit the subscript $EQ$.) Intuitively, each equation $a_i \circ b_i = b_i \circ a_i$ indicates that events $a_i$ and $b_i$ are independent or concurrent. In practice, the set of
equations is induced by the symmetric and irreflexive independence relation \( \text{ind} \subseteq E \times E \) such that \( EQ = \{ ab = ba \mid (a, b) \in \text{ind} \} \). With this in mind, a concrete example motivated by the EN-system in Figure 1(a), is given by:

\[
E_0 = \{ a, b, c, d \} \quad \text{and} \quad \text{ind} = \{ (c, d), (d, c) \} \quad \text{and} \quad EQ_0 = \{ cd = dc \}.
\]

Then \( acd \equiv adc \) and \( bcd \equiv bdc \), while \( abcd \equiv abdec \) because \( abcd \approx abdec \approx abdec \). Moreover, if we take three Mazurkiewicz traces:

\[
x = [[acd]] = \{ acd, adc \} \quad y = [[ac]] = \{ ac \} \quad z = [[d]] = \{ d \},
\]

then \( x = y \circ z \) as \( acd = ac \circ d \).

**Example 2 (Comtraces).** Consider a monoid of step sequences \( M = (S^*, \circ, 1) \) with a step alphabet \( \mathcal{S} \) over \( E \), which is subset-closed, i.e., whenever \( A \in \mathcal{S} \) then all its non-empty subsets also belong to \( \mathcal{S} \), and with equations:

\[
EQ = \{ C_1 = A_1 \circ B_1 \ldots C_n = A_n \circ B_n \},
\]

where each \( C_i \) belongs to \( \mathcal{S} \) and \( A_i, B_i \) form a partition of \( C_i \). The elements of \( \mathcal{M}_\mathcal{S} = (S^*/\circ, \{1\}) \) are called comtraces [12]. (Again, we omit the subscript \( EQ \).) Intuitively, the steps in \( \mathcal{S} \) prescribe which events may occur simultaneously, while each equation \( C_i = A_i \circ B_i \) indicates that the events in \( C_i \) can be partitioned into two steps, \( A_i \) and \( B_i \), with step \( A_i \) occurring before \( B_i \). Similarly as before, one often specifies comtraces using, this time, two relations \( \text{ser} \subseteq \text{sim} \subseteq E \times E \), respectively called serialisability and simultaneity, such that \( \text{sim} \) is irreflexive and symmetric. The interpretation of the relations \( \text{sim} \) and \( \text{ser} \) is that the former states which events can be executed simultaneously, whereas the latter specifies which pairs of events can be executed also in order whenever they can be executed simultaneously. Then the set \( \mathcal{S} \) of all (potential) steps is the set of all cliques of the relation \( \text{sim} \), and the equations are given by

\[
EQ = \{ C = A \circ B \mid A \times B \subseteq \text{ser} \wedge C = A \cup B \wedge A \cap B = \emptyset \}.
\]

With this in mind, a concrete example motivated by the ENI-system in Figure 1(b), is given by:

\[
E_1 = \{ a, b, c, d \} \quad \text{and} \quad \text{sim} = \{ (c, d), (d, c) \} \quad \text{and} \quad \text{ser} = \{ (c, d) \}
\]

\[
\mathcal{S}_1 = \{ \{a\}, \{b\}, \{c\}, \{d\}, \{c,d\} \} \quad \text{and} \quad EQ_1 = \{ \{ c, d \} = \{ c \} \{ d \} \}.
\]

Then \( \{a\} \{c \} \{ d \} \equiv \{a\} \{c \} \{ d \} \) but \( \{a\} \{c, d\} \not\equiv \{a\} \{d\} \{c\} \). Moreover, if we take three comtraces:

\[
x = [[\{a\} \{c, d\}]] = \{ \{a\} \{c, d\}, \{a\} \{c\} \{d\} \} \quad y = [[\{a\} \{c\}]] = \{ \{a\} \{c\} \}
\]

\[
z = [[\{d\}]] = \{ \{d\} \},
\]

then \( x = y \circ z \) as \( \{a\} \{c \} \{ d \} = \{a\} \{c \} \circ \{d\} \).

Mazurkiewicz trace monoids can be seen as special comtrace monoids. To start with, we can treat each sequence \( w = a_1 \ldots a_k \) of action names as a step sequence
\( \xi(w) = \{a_1\} \ldots \{a_k\} \) consisting of singleton sets. Then \( S \) comprises all non-empty subsets \( A \) of \( E \) such that, for all distinct \( a,b \in A \), we have that \( a \circ b = b \circ a \) is an equation; furthermore, for each equation \( a \circ b = b \circ a \) on sequences, we introduce two equations on step sequences: \( \{a\}\{b\} = \{b\}\{a\} \) and \( \{a,b\} = \{b\}\{a\} \). It can be seen that for each Mazurkiewicz trace \( \xi(w) \), the corresponding comtrace \( \xi(w) \) is such that the set of singleton step sequences it comprises is exactly \( \xi(w) \) as well as \( \xi(w) = \xi(w) \circ \xi(w) \), for all sequences \( w \) and \( v \) of action names. Note that if a Mazurkiewicz trace monoid is defined by an independence relation \( ind \), then the corresponding comtrace monoid is given by the simultaneity and serialisability relations \( sim = ser = ind \).

### 3 Paradigms of concurrency and generalised order structures

Let \( \Delta \) be a non-empty set of stratified posets (or, equivalently, singular step sequences) with the same domain \( X \) (or \( X_{\Delta} \)). Intuitively, each poset in \( \Delta \) is an observation of an abstract history of a hypothetical concurrent system. Following the true concurrency approach, [11] attempted to represent \( \Delta \) using relational invariants on \( X \). The basic idea was to capture situations where knowing some (or all) invariant relationships between the events involved in \( \Delta \) would be sufficient to reconstruct the entire set \( \Delta \) of observations.\(^3\)

The approach of [11] identified a number of fundamental invariants which can be attributed to the observations in \( \Delta \), each invariant describing a relationship between pairs of events repeated in all observations of \( \Delta \). In particular, \( \prec_{\Delta} \) comprises all pairs \( (a,b) \) such that \( a \) precedes \( b \) in every poset belonging to \( \Delta \); in other words, \( \prec_{\Delta} \) represents causality. Other fundamental invariants are: \( =_{\Delta} \) (commutativity, where \( a =_{\Delta} b \) means that \( a \) and \( b \) are never simultaneous), \( \sqsubset_{\Delta} \) (weak causality, where \( a \sqsubset_{\Delta} b \) means that \( a \) is never observed after \( b \)) and \( \bowtie_{\Delta} \) (synchronisation, where \( a \bowtie_{\Delta} b \) means that \( a \) and \( b \) are always simultaneous). One can show that knowing \( =_{\Delta} \) and \( \sqsubset_{\Delta} \) is always sufficient to reconstruct \( \Delta \). This is done assuming that \( \Delta \) is invariant-closed in the sense that \( \Delta \) comprises all stratified posets \( spo = (X_{\Delta}, \prec_{spo}) \) which respect all the fundamental invariants generated by \( \Delta \), e.g., \( a \prec_{spo} b \) implies \( a \prec_{\Delta} b \), and \( a \sqsubset_{\Delta} b \) implies \( a \sqsubset_{spo} b \). An invariant-closed set of observations is also called a (concurrent) history. Note that being invariant-closed is a natural assumption when constructing an abstract view of a set of individual observations, and has always been tacitly assumed in the causal partial order view of concurrent computation.

Depending on the underlying system model of concurrent computation, some additional constraints on histories \( \Delta \) may be added. In particular, they may adhere to the ‘diagonal rule’ (or ‘diamond property’) by which simultaneity is the same as the possibility of occurring in any order, i.e., for all \( a,b \in \Delta_{X} \):

\[
(\exists spo \in \Delta : a \prec_{spo} b) \iff (\exists spo \in \Delta : a \prec_{spo} b) \land (\exists spo \in \Delta : b \prec_{spo} a) . \quad (\pi_{S})
\]

For example, \( \pi_{S} \) is satisfied by concurrent histories generated by EN-systems.

\(^3\) Note that [11] also considered total and interval poset observations.
Constraints like \( \pi_8 \) — called **paradigms** in \([10, 11]\) — are essentially suppositions or statements about the intended treatment of simultaneity and, moreover, allow one to simplify the invariant representation of a history \( \Delta \). In particular, if \( \Delta \) satisfies \( \pi_8 \) then one can reconstruct \( \Delta \) using just causality \( \prec_{\Delta} \) (which is always equal to the intersection of \( \sqsubseteq_{\Delta} \) and \( \sqsubset_{\Delta} \)). This is the essence of the true concurrency paradigm based on causal partial orders.

In general, knowing \( \prec_{\Delta} \) is insufficient to reconstruct \( \Delta \). For example, if we weaken \( \pi_8 \) to the paradigm:

\[
(\exists \text{spos} \in \Delta : a \prec_{\text{spos}} b) \land (\exists \text{spos} \in \Delta : b \prec_{\text{spos}} a) \implies (\exists \text{spos} \in \Delta : a \prec_{\text{spos}} b)
\] (\( \pi_3 \))

then one needs to enhance causality with weak causality \( \sqtriangleleft_{\Delta} \) to provide an invariant representation of \( \Delta \). The resulting relational structure \( (X, \prec_{\Delta}, \sqtriangleleft_{\Delta}) \) is an instance of the following notion.

**Definition 1 (stratified order structure \([8, 13, 17, 18]\)).** A stratified order structure (**SO-structure**) is a relational structure \( \text{sos} = (X, \prec, \sqsubseteq) \) where \( \prec \) and \( \sqsubseteq \) are binary relations on \( X \) such that, for all \( a, b, c \in X \):

\[
\begin{align*}
S1 &: \ a \not\sqsubseteq a \\
S2 &: \ a \prec b \implies a \sqsubseteq b \\
S3 &: \ a \sqsubseteq b \sqsubseteq c \land a \neq c \implies a \sqsubset c \\
S4 &: \ a \sqsubseteq b \prec c \lor a \prec b \sqsubseteq c \implies a \prec c .
\end{align*}
\]

The axioms imply that \( \prec \) is a partial order relation, and that \( a \prec b \) implies \( b \not\sqsubseteq a \). The relation \( \prec \) represents the ‘earlier than’ relationship on the domain of \( \text{sos} \), and the relation \( \sqsubseteq \) the ‘not later than’ relationship. The four axioms capture the mutual relationship between the ‘earlier than’ and ‘not later than’ relations between executed actions.

For every stratified poset \( \text{spos} = (X_{\text{spos}}, \prec_{\text{spos}}, \sqsubseteq_{\text{spos}}) \), the relational structure \( \text{sos}(\text{spos}) = (X_{\text{spos}}, \prec_{\text{spos}}, \sqsubseteq_{\text{spos}}) \) is an SO-structure. Moreover, \( \text{spos} \) is a stratified poset extension of an SO-structure \( \text{sos} \) whenever \( \text{sos} \subseteq \text{sos}(\text{spos}) \). We denote this by \( \text{spos} \in \text{ext}(\text{sos}) \).

Following Szpilrajn’s Theorem \([27]\) that any poset can be reconstructed by intersecting its total extensions, we have that any SO-structure can be reconstructed from its stratified poset extensions.

**Fact 1 ([13])** If \( \text{sos} \) is an SO-structure then \( \text{ext}(\text{sos}) \neq \emptyset \) and

\[
\text{sos} = \bigcap \{\text{sos}(\text{spos}) \mid \text{spos} \in \text{ext}(\text{sos})\} .
\]

Moreover, if \( \text{SPO} \) is a non-empty set of stratified posets with the same domain, then \( \bigcap \{\text{sos}(\text{spos}) \mid \text{spos} \in \text{SPO}\} \) is an SO-structure.

The set of stratified poset extensions of an SO-structure is a concurrent history satisfying paradigm \( \pi_3 \) (see \([11]\)). Moreover \((10)\), if a concurrent history \( \Delta \) satisfies \( \pi_3 \), then \( \Delta = \text{ext}(X_\Delta, \prec_\Delta, \sqsubseteq_\Delta) \). Hence each abstract history \( \Delta \) adhering to paradigm \( \pi_3 \) can be represented by the SO-structure \( (X_\Delta, \prec_\Delta, \sqsubseteq_\Delta) \).

One of the key insights of the classical Mazurkiewicz trace approach is that each trace corresponds to a unique (up to isomorphism) poset labelled by the elements forming the trace (see.e.g., \([26, 14]\)) or, equivalently, a dependence graph which underpins
such a poset [7]. For the Mazurkiewicz trace monoid of Example 1, we have that, in the partial order corresponding to \([acd] \), a precedes both c and d which are unrelated. There are similar relationships between comtraces, SO-structures and generalised dependence graphs [19, 22]. For the comtrace monoid in Example 2, we have that in the SO-structure corresponding to the comtrace \([acd] \), a precedes both c and d, and c precedes d in a weak sense.

If \( \Delta \) fails to satisfy \( \pi_{3} \), knowing \((X_{\Delta}, \preceq_{\Delta}, \sqsubseteq_{\Delta})\) may be insufficient to reconstruct \( \Delta \). In the case of paradigm \( \pi_{1} \) which places no restrictions of the kind captured by \( \pi_{8} \) or \( \pi_{3} \) (i.e., \( \Delta \) is only assumed to be invariant-closed), one needs to use general SO-structures (GSO-structures).

**Definition 2 (GSO-structure [9, 10]).** A relational structure \( gsos = (X, \equiv, \sqsubseteq) \) is a GSO-structure if \( sos(gsos) = (X, \equiv \cap \sqsubseteq, \sqsubseteq) \) is an SO-structure and the relation \( \equiv \) is symmetric and irreflexive.

In the above, \( \equiv \) represents the ‘earlier than or later than, but never simultaneous’ relationship, while \( \sqsubseteq \) again represents the ‘not later than’ relationship.

For a stratified poset \( spo \), \( gsos(spo) = (X_{spo}, \preceq_{spo}, \sqsubseteq_{spo}) \) is a GSO-structure. Also, \( spo \) is a stratified poset extension of a GSO-structure \( gsos \) if \( gsos \subseteq gsos(spo) \). We denote this by \( spo \in \text{ext}(gsos) \). Each GSO-structure can be reconstructed from its stratified poset extensions, leading to another generalisation of Szpilrajn’s Theorem.

**Fact 2 ([9, 10])** If \( gsos \) is a GSO-structure then \( \text{ext}(gsos) \neq \emptyset \) and

\[
gsos = \bigcap \{ \text{gsos}(spo) \mid spo \in \text{ext}(gsos) \} .
\]

Moreover, if \( SPO \) is a non-empty set of stratified posets with the same domain, then \( \bigcap \{ \text{gsos}(spo) \mid spo \in SPO \} \) is a GSO-structure.

The set of stratified poset extensions of a GSO-structure is a concurrent history. Moreover, if \( \Delta \) is a concurrent history, then \( \Delta = \text{ext}(X_{\Delta}, \equiv_{\Delta}, \sqsubseteq_{\Delta}) \). Hence each abstract history \( \Delta \) can be represented by the GSO-structure \( (X_{\Delta}, \equiv_{\Delta}, \sqsubseteq_{\Delta}) \), see [10].

**Constructing order structures**

Before introducing a generalisation of comtraces that matches GSO-structures, we first discuss how to construct globally defined (generalised) order structures from directly observed or locally defined relations between events. We proceed similarly as when constructing posets from acyclic relations through the operation of transitive closure. First, we recall how the notion of transitive closure was lifted to the level of SO-structures.

Let \( \mu = (X, \prec, \sqsubseteq) \) be a relational structure (not necessarily an SO-structure). Intuitively, \( \prec \) indicates which of the executed actions in \( X \) are directly causally related, and \( \sqsubseteq \) which are directly weakly causally related. The so-closure of \( \mu \) is defined as \( \mu^{so} = (X, \alpha \setminus id_{X}) \), where \( \gamma = (\prec \cup \sqsubseteq)^{*} \), \( \alpha = \gamma \circ \prec \circ \gamma \) and \( id_{X} \) is the identity on \( X \). Moreover, \( \mu \) is so-acyclic if \( \alpha \) is irreflexive. As shown [12], in this case \( \mu^{so} \) is an SO-structure.
We will now show how to construct GSO-structures. Let \( \rho = (X, \sim, \sqsubseteq, \equiv) \) be a relational structure. In addition to the two relations appearing also in \( \mu \) above, \( \equiv \) indicates which of the executed actions may be observed in any order, but not simultaneously. The gso-closure of \( \rho \) is defined as a tuple \( \rho^{\text{gso}} = (X, \psi, \gamma \setminus \text{id}_X) \), where \( \psi = \alpha^{\text{sym}} \cup \beta^{\text{sym}} \cup \equiv \) with \( \beta = \circ (\equiv \cap \circ^*) \circ^* \), in addition to \( \alpha \) and \( \gamma \) being defined as for \( \rho^{\text{so}} \). Moreover, \( \rho \) is gso-acyclic if \( \psi \) is irreflexive and symmetric.

The following two results proved in [20] are necessary in order to associate causal structures (GSO-structures) to the processes of ENIM-systems.

**Proposition 1 (20).** If \( \rho \) is gso-acyclic then \( \rho^{\text{gso}} \) is a GSO-structure.

**Proposition 2 (20).** If \( \rho \) is gso-acyclic then \( (X, \sim, \sqsubseteq) \) is an so-acyclic relational structure and \( \text{ext}(\rho^{\text{gso}}) = \{ \text{spo} \in \text{ext}((X, \sim, \sqsubseteq)^{\text{so}}) \mid \sim_{\text{spo}} \sqsubset \equiv \} \).

## 4 Generalised Comtraces

To describe in an algebraic way (as elements of a quotient monoid) the step sequences associated with a concurrent history (i.e., belonging to a GSO-structure) it is necessary to have next to simultaneity and serialisability, additional information on the relations between actions. Consider, for instance, the following three atomic assignments:

\[
\begin{align*}
\begin{array}{l l l l}
    x & \leftarrow & x + 1 & \text{ by } a \\
    x & \leftarrow & x - 1 & \text{ by } b \\
    y & \leftarrow & y + 1 & \text{ by } c
\end{array}
\end{align*}
\]

It seems reasonable to allow \{\(a, c\)\} and \{\(b, c\)\} to occur simultaneously (as steps). Simultaneous execution of all three assignments should not be allowed as concurrent writing to the same variable is unsafe. Still, it is reasonable to regard them all as ‘independent’ since any order of their executions avoiding simultaneous execution of \(a\) and \(b\) yields the same result. Thus the set

\[
\mathbf{x} = \{ \{a\}\{b\}\{c\}, \{a\}\{c\}\{b\}, \{b\}\{a\}\{c\}, \{b\}\{c\}\{a\}, \{c\}\{a\}\{b\},
\{c\}\{b\}\{a\}, \{a, c\}\{b\}, \{b, c\}\{a\}, \{b\}\{a, c\}, \{a\}\{b, c\} \}
\]

should be a valid concurrent history. However, \(\mathbf{x}\) is *not* a comtrace since in such a case we would have \(\{a\}\{b\}\{c\} \equiv \{b\}\{a\}\{c\}\) as \(\{a\}\{b\}\{c\}, \{b\}\{a\}\{c\} \in \mathbf{x}\). But this is not possible since \(\{a, b\}\) is *not* a valid step. Extending comtraces to handle cases like this has led to the introduction of generalised comtraces (or G-comtraces) [16].

To start with, a tuple \(\Psi = (E, \sim_{\text{ser}}, \sim_{\text{inl}})\) is a G-comtrace alphabet if \(E\) is an alphabet of action names and \(\text{ser}, \sim_{\text{sim}}\) and \(\text{inl}\) are three relations on \(E\), respectively called serialisability, simultaneity and interleaving. It is assumed that \(\sim_{\text{sim}}\) and \(\text{inl}\) are irreflexive and symmetric, \(\text{ser} \subseteq \sim_{\text{sim}}\) and \(\sim_{\text{inl}} \cap \text{inl} = \emptyset\). The intuition behind \(\sim_{\text{sim}}\) and \(\text{ser}\) as before, and \(\{a, b\} \in \text{inl}\) means that \(a\) and \(b\) cannot occur simultaneously, but if they occur one after the other, then the resulting executions are equivalent. As for comtraces, the (potential) steps \(S\) are the cliques of the relation \(\sim_{\text{sim}}\). The G-comtrace congruence \(\equiv_{\text{ser, inl}}\) is then generated by the set of equations \(\text{EQ} = \text{EQ}_{\text{ser}} \cup \text{EQ}_{\text{inl}}\) where:

\[
\begin{align*}
\text{EQ}_{\text{ser}} &= \{ A = BC \mid A = B \cup C \in S \land B \times C \subseteq \text{ser} \} \\
\text{EQ}_{\text{inl}} &= \{ BA = AB \mid A \in S \land B \in S \land A \times B \subseteq \text{inl} \}.
\end{align*}
\]
The quotient monoid \((S^*/_{\text{ser, inl}}, \delta, [1])\) is the monoid of G-comtraces. As usual, we can omit the subscripts \(\text{ser, inl}\) if this does not lead to ambiguity. Note also that comtraces are nothing but G-comtraces with an empty relation \(\text{inl}\).

For example, the set of step sequences \(x\) enumerated in (1) above can be seen as a G-comtrace \(x = \llbracket \{a, c\} \{b\} \rrbracket\) with:

\[
S = \text{ser} = \{a, b, c\} \quad \text{and} \quad \mathcal{S} = \{\{a\}, \{b\}, \{c\}, \{a, c\}, \{b, c\}\}
\]

\[
E = \{a, b, c\} \quad \text{and} \quad \mathcal{S} = \{\{a\}, \{b\}, \{c\}, \{a, c\}, \{b, c\}\}
\]

\[
\text{ser} = \text{sim} = \{(a, c), (c, a), (b, c), (c, b)\} \quad \text{and} \quad \text{inl} = \{(a, b), (b, a)\}.
\]

Another example, motivated by the ENIM-system in Figure 1(c), is given by:

\[
E_1 = \{a, b, c, d\} \quad \text{and} \quad \mathcal{S}_1 = \{\{a\}, \{b\}, \{c\}, \{d\}\}
\]

\[
\text{ser} = \text{sim} = \emptyset \quad \text{and} \quad \text{inl} = \{(c, d), (d, c)\}.
\]

In such a case, \(\llbracket \{a\} \{c\} \{d\} \rrbracket = \{\{a\} \{c\} \{d\}, \{a\} \{d\} \{c\}\}\).

Recall (see Section 2 or [14]) that we have a more constructive way to define the congruence induced by \(EQ_{\text{ser}}\) and \(EQ_{\text{sim}}\), by repeatedly splitting and combining steps or interchanging adjacent occurrences of commutative steps.

**Fact 3 ([14, 16])** Let \(\approx\) be the relation comprising all pairs \((t, u)\) of step sequences in \(S^*\) such that:

\[- t = wAz \text{ and } u = wBCz \text{ where } A = B \cup C \text{ and } B \times C \subseteq \text{ser}, \text{ or} \]

\[- t = wABz \text{ and } u = wBAz \text{ where } A \times B \subseteq \text{inl}, \]

for some \(w, z \in S^*\) and \(A, B, C \in \mathcal{S}\). Then the relation \(\equiv\) is equal to \((\approx_{\text{sym}})^*\).

The relationship between G-comtraces and GSO-structures is similar to that between traces and partial orders as well as that between comtraces and so-structures [14, 16]. Each G-comtrace uniquely determines a GSO-structure and each GSO-structure can be represented by a G-comtrace.

Let \(\Psi = (E, \text{sim}, \text{ser}, \text{inl})\) be a G-comtrace alphabet and \(u \in S^*\) be a step sequence. Since \(\text{occ}(u) = \text{occ}(x)\), for every step sequence \(x\) satisfying \(u \equiv x\), we can use \(\text{occ}([u]) = \text{occ}(u)\) to denote the set of all occurrences of actions from \(E\) in \([u]\). Moreover, each \(\text{spo}^{\text{occ}}(x) = (\text{occ}(u), \approx_{\text{x}})\) with \(x \in [u]\) is a stratified poset and, relying on Fact 2, we define the GSO-structure induced by \([u]\), as follows:

\[
G_{[u]} = \left(\text{occ}([u]), \bigcap_{x \in [u]} \approx_{\text{x}}, \bigcap_{x \in [u]} \approx_{\text{x}}\right).
\]

**Fact 4 ([14, 16])** Let \(u, v \in S^*\). Then \(G_{[u]}\) is a GSO-structure such that \(\text{ext}(G_{[u]}) = \{\text{spo}^{\text{occ}}(x) \mid x \in [u]\}\). Moreover, \(u \equiv v \iff G_{[u]} = G_{[v]}\).

Thus \(G_{[u]}\) is a well defined GSO-structure whose stratified poset extensions exactly match the step sequences in the G-comtrace \([u]\).

Conversely, it turns out that each GSO-structure \(G = (X, \approx, \sqsubseteq)\) can be represented by the G-comtrace generated by \(G\) which is defined as \(\text{gctr}_G = \{\text{steps}(spo) \mid \}\).
mutex causality 13

spo ∈ ext(G). To justify this definition, we first take the comtrace alphabet Ψ = (X, sim_G, ser_G, inl_G) such that, for all distinct a, b ∈ X:

(a, b) ∈ sim_G if ¬(a ≡ b)
(a, b) ∈ ser_G if ¬(a ≡ b) ∧ ¬(b ⊏ a)
(a, b) ∈ inl_G if a ≡ b ∧ ¬(a ⊏ b ∨ b ⊏ a).

Note that ser_G ⊆ sim_G, the relations sim_G and inl_G are symmetric, sim_G ∩ inl_G = ∅, and all three relations are irreflexive, so Ψ is indeed a G-comtrace alphabet. Hence we can form G-comtraces over Ψ_G. The definition of gctr_G is then backed up by the following result.

Fact 5 ([14, 16]) Let spo ∈ ext(G). Then gctr_G is the G-comtrace [steps(spo)]. Moreover, G[steps(spo)] is the GSO-structure G with every element x in the domain changed to x′.

Together with Fact 4, this means that G-comtraces and GSO-structures are equivalent models.

5 Fitting nets and order structures

The operational and axiomatic process semantics leading to a mutually consistent causality semantics of a class of a Petri net can be related according to a schema introduced in [18], that is common to different classes of Petri nets PN. It is reproduced here as Figure 2 where N is a net from PN and:

– EX are executions (or observations) of nets in PN.
– LAN are labelled acyclic nets, each representing a history.
– LEX are executions of nets in LAN.
– LCS are labelled causal structures (order structures) capturing the abstract causal relationships between executed actions.

In this paper, the executions in EX are step sequences, and the labelled executions in LEX are labelled singular step sequences.

The maps in Figure 2 relate the semantical views in EX, LAN, LEX, and LCS:
– \( \omega \) returns a set of executions, defining the operational semantics of \( N \).
– \( \alpha \) returns a set of labelled acyclic nets, defining an axiomatic process semantics of \( N \).
– \( \pi_N \) returns, for each execution of \( N \), a non-empty set of labelled acyclic nets, defining an operational process semantics of \( N \).
– \( \lambda \) returns a set of labelled executions for each process of \( N \), and after applying \( \phi \) to such labelled executions one should obtain executions of \( N \).
– \( \kappa \) associates a labelled causal structure with each process of \( N \).
– \( \epsilon \) and \( i \) allow one to go back and forth between labelled causal structures and sets of labelled executions associated with them.

The semantical framework captured by the above schema indicates how the different semantical views should agree. According to the rectangle on the left, the operational semantics of 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 executions of the original Petri net. The triangle on the right relates the labelled acyclic nets from \( \text{LAN} \) with the causal structures from \( \text{LCS} \) and the labelled executions from \( \text{LEX} \). The order structures defined by a labelled acyclic net can be obtained by combining executions of that net and, conversely, the stratified extensions of an order structure defined by a labelled acyclic net are its (labelled) executions. 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 2 and proving these properties.

**Property 1 (soundness of mappings)** The maps \( \omega, \alpha, \lambda, \phi, \pi_N|_{\omega(N)}, \kappa, \epsilon \) and \( i|_{\lambda(\text{LAN})} \) are total. Moreover, \( \omega, \alpha, \lambda, \pi_N|_{\omega(N)} \) and \( \epsilon \) always return non-empty sets.

**Property 2 (consistency)** For all \( \xi \in \text{EX} \) and \( LN \in \text{LAN}, \xi \in \omega(N) \land LN \in \pi_N(\xi) \) iff \( LN \in \alpha(N) \land \xi \in \phi(\lambda(LN)) \). 

**Property 3 (representation)** \( i \circ \epsilon = id_{\text{LCS}} \). 

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

The above four properties imply that the axiomatic (defined through \( \alpha \)) and operational (defined through \( \pi_N \circ \omega \)) process semantics of nets in \( \text{PN} \) are in full agreement. Also, the operational semantics of \( N \) (defined through \( \omega \)) coincides with the operational semantics of the processes of \( N \) (defined through \( \phi \circ \lambda \circ \alpha \)). Moreover, the causality in a process of \( N \) (defined through \( \kappa \)) coincides with the causality structure implied by its operational semantics (through \( i \circ \lambda \)). That is, we have the following.

**Aim 1** \( \alpha = \pi_N \circ \omega \).
Aim 2 $\omega = \phi \circ \lambda \circ \alpha$.  

Aim 3 $\kappa = 1 \circ \lambda$.

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

**EN-systems with inhibitor arcs**

Here, we use elementary net systems with inhibitor arcs (ENI-systems) to show how the semantical framework can be instantiated. Moreover, in the next section we will extend the definitions and constructions given here to ENI-systems with mutex arcs.

An ENI-system is a tuple $ENI = (P, T, F, Inh, M_{init})$ with $P$ and $T$ finite and disjoint sets of places and transitions — drawn as circles and rectangles, respectively; $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation of $ENI$ — represented by directed arcs in the diagrams; $Inh \subseteq P \times T$ its set of inhibitor arcs — arcs with small circles as arrowheads; and $M_{init} \subseteq P$ its initial marking. (In general, any subset of places is a marking, in diagrams indicated by tokens, i.e., small black dots.) If $ENI$ has no inhibitor arcs, $Inh = \emptyset$, then it is simply an elementary net system (EN-system).

As usual, for every transition or place $x$ we define its inputs $x^* = \{ y \mid (y, x) \in F \}$ and outputs $x^* = \{ y \mid (x, y) \in F \}$, and then $x^* = x^* \cup x^*$. We also require that $t \neq \emptyset \neq t^*$, for every transition $t$. Moreover, $Ot = \{ p \mid (p, t) \in Inh \}$ are the inhibitor places of transition $t$. We also define for any subset $U$ of $T$:

$$U = \bigcup_{t \in U} t^* \text{ and } U^* = \bigcup_{t \in U} t^* \text{ and } Ot = \bigcup_{t \in U} Ot .$$

A step of $ENI$ is a non-empty set $U$ of transitions such that $U^* \cap U^* = \emptyset$, for all distinct $t, u \in U$. A step $U$ of $ENI$ is enabled at a marking $M$ of $ENI$ if $U \subseteq M$ and $(U^* \cup Ot) \cap M = \emptyset$. Such a step can then be executed leading to the marking $M' = (M \setminus U^*) \cup U^*$. We denote this by $M[U]_{ENI}M'$ or by $M[U]M'$ if $ENI$ is clear.

Thus the operational semantics of $ENI$ is defined: $\omega(ENI)$ comprises all step sequences $\xi = U_1 \ldots U_k \ (k \geq 0)$ such that there are markings $M_{init} = M_0, \ldots, M_k$ with $M_{init}[U_i]M_i$, for $i = 0, \ldots, k - 1$. We call $M_k$ a reachable marking of $ENI$.

In what follows we will assume that each inhibitor place $p$ of an ENI-system $ENI$ has a complement place $\tilde{p}$ such that $p^* = \tilde{p}$ and $\tilde{p}^* = p^*$; moreover $\{|p, \tilde{p}| \cap M_{init} \} = 1$. It is immediate that $\{|p, \tilde{p}| \cap M| = 1$, for all reachable markings $M$ and all places $p$. Note that complement places can always be added to $ENI$ as this does not affect its operational semantics.

Thus, for ENI-systems EX are step sequences. In addition, the labelled causal structures LCS are SO-structures, and the labelled executions LEX will be labelled singular step sequences. Next we introduce the labelled acyclic nets that will form the semantical domain LAN for the process semantics of ENI-systems. These nets will have activator rather than inhibitor arcs.

**Definition 3 (activator occurrence nets).** An activator occurrence net (or AON-net) is a tuple $AON = (P', T', F', Act, \ell)$ such that:
As far as the mappings \( \rho \) and \( \iota \) are concerned, \( \iota \) is the set of stratified poset extensions (or, equivalently, singular step sequences) of an so-structure, and \( \rho \) is the intersection of the so-structures (or, equivalently, singular step sequences) corresponding to a set of stratified posets with the same domain. Thus Fact 1 immediately yields Property 3.

We define \( \epsilon(AON) = \rho_{AON}^{\prime} \) which is guaranteed to be an so-structure by the so-acyclicity of \( \rho_{AON} \) (as mentioned in Section 3, see also [12]). As far as the mappings \( \epsilon \) and \( \iota \) are concerned, \( \epsilon \) is the set of stratified poset extensions (or, equivalently, singular step sequences) of an so-structure, and \( \iota \) is the intersection of the so-structures (or, equivalently, singular step sequences) corresponding to a set of stratified posets with the same domain. Thus Fact 1 immediately yields Property 3.

To conclude this section, we give the axiomatic and operational process semantics of an ENI-system \( ENI = (P,T,F,Inh,M_{init}) \).

**Definition 4 (processes of ENI-systems).** A process of ENI is an so-net AON such that its labelling \( \ell \):

- labels the places of AON with places of ENI.
- labels the transitions of AON with transitions of ENI.
- is injective on \( M_{init}^{AON} \) and \( \ell(M_{init}^{AON}) = M_{init} \).

![Fig. 3. Two cases (a) and (b) defining \( t \prec_{loc} u \), and one case (c) defining \( t \preceq_{loc} u \).](image-url)
should however not be regarded as an unusual feature as, for example, Petri nets with undirected edges). Note that mutex arcs are relating transitions in a direct way. This allows us to model such a scenario we use a mutex arc between transitions to enforce direct relations between transitions.

We denote this by $AON \in \alpha(ENI)$.

**Definition 5 (processes construction).** An $\infty$-net generated by a step sequence $\sigma = U_1 \ldots U_n \in \omega(ENI)$ is the last element in the sequence $AON_0, \ldots, AON_n$ where each $AON_k = (F_k, T_k, F_k, A_k, t_k)$ is an $\infty$-net such that:

**Step 0:** $P_0 = \{p^1 \mid p \in M_{init}\}$ and $T_0 = F_0 = A_0 = \emptyset$.

**Step k:** Given $AON_{k-1}$ the sets of nodes and arcs are extended as follows:

$$
\begin{align*}
P_k &= P_{k-1} \cup \{p^{1+\Delta p} \mid p \in U_k^\bullet\} \\
T_k &= T_{k-1} \cup \{p^{1+\Delta t} \mid t \in U_k\} \\
F_k &= F_{k-1} \cup \{(p^{1+\Delta p}, t^{1+\Delta t}) \mid t \in U_k \land p \in t^\bullet\} \\
&\quad \cup \{(t^{1+\Delta t}, p^{1+\Delta p}) \mid t \in U_k \land p \in t^\bullet\} \\
A_k &= A_{k-1} \cup \{(p^{\Delta p}, t^{1+\Delta t}) \mid t \in U \land p \in U^\bullet\}.
\end{align*}
$$

In the above, the label of each node $\ell_k(x^\circ)$ is set to be $x$, and $\Delta x$ denotes the number of the nodes of $AON_{k-1}$ labelled by $x$. We denote this by $AON_n \in \pi_{ENI}(\sigma)$.

Note that $\pi_{ENI}(\sigma)$ comprises exactly one net (up to isomorphism). The same holds for $\pi_{ENIM}(\sigma)$ defined later. Note also that $occ(\sigma)$ is the transition set of $AON_n$.

As one can show that the remaining properties are also satisfied, the semantical framework for ENI-systems holds [18].

## 6 Mutually exclusive transitions

We now extend ENI-systems with mutex arcs that prohibit pairs of transitions from occurring simultaneously (i.e., in the same step). Consider Figure 4 which shows a variant of the producer/consumer scheme. In this case, the producer is allowed to retire (transition $r$), but never at the same time as the consumer finishes the job (transition $f$). Other than that, there are no restrictions on the executions of transitions $r$ and $f$. To model such a scenario we use a mutex arc between transitions $r$ and $f$ (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.

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

It follows immediately from the definitions that...
Fig. 4. An ENIM-system modelling a producer/consumer system with the actions: ‘make item’ \(m\), ‘add item to buffer’ \(a\), ‘loss of item from buffer’ \(l\), ‘get item from buffer’ \(g\), ‘use item’ \(u\), ‘producer retires’ \(r\), and ‘consumer finishes’ \(f\). Note: the producer can only retire if the buffer is empty (i.e., \(p_3\) is empty).

\[\text{Proposition 3. } \omega(\text{ENIM}) = \{U_1 \ldots U_k \in \omega(\text{und}(\text{ENIM})) \mid Mtx\cap\bigcup_i U_i \times U_i = \emptyset\}.\]

For the ENIM-system of Figure 4, we have that \(M[\{r\}]M'[\{f\}]M''\) as well as \(M[\{f\}]M'[\{r\}]M''\), where \(M = \{p_2, p_4, p_6\}\) and \(M' = \{p_0, p_4, p_7\}\). However, \(M[\{r, f\}]M''\) which holds for the underlying ENI-system does not hold now as \(r\) and \(f\) cannot be executed in the same step.

To deal with the behaviours of ENIM-systems in the context of the semantical framework, we adapt the approach followed for ENI-system as recalled above. The labelled causal structures, LCS, are now GSO-structures, while labelled executions, LEX, are labelled singular step sequences, as before. The labelled acyclic nets, LAN, used for the process semantics of ENIM-systems are introduced next.

**Definition 6 (activator mutex occurrence nets).** An activator mutex occurrence net (or AMO-net) is a tuple \(\text{AMON} = (P', T', F', \text{Act}', \text{Mtx}', \ell)\) such that:

- \(\text{und}(\text{AMON}) = (P', T', F', \text{Act}', \ell)\) is the AO-net underlying AMON and \(Mtx' \subseteq T' \times T'\) is a symmetric irreflexive relation specifying the mutex arcs of AMON.
- \(\rho_{\text{AMON}} = (T', \prec_{\text{loc}}, \sqsubseteq_{\text{loc}}, \text{Mtx}')\), where \(\prec_{\text{loc}}\) and \(\sqsubseteq_{\text{loc}}\) are defined as for AO-nets in Definition 3, is a gso-acyclic relational structure.

The part of the gso-acyclicity of \(\rho_{\text{AMON}}\) which deals with the mutex arcs is illustrated in Figure 5. We have there two transitions satisfying \(b \sqsubseteq_{\text{loc}} c \sqsubseteq_{\text{loc}} b\). Hence, in any execution involving both of them, they have to belong to the same step. This, however, is inconsistent with the mutex arc between \(b\) and \(c\), and the gso-acyclicity fails to hold because \((b, b)\) belongs to \(\sqsubseteq_{\text{loc}}\circ (Mtx'\cap \sqsubseteq_{\text{loc}})\circ \sqsubseteq_{\text{loc}}\).

Then we let \(\kappa(\text{AMON}) = \rho_{\text{AMON}}^g\) be the GSO-structure generated by AMON. Note that Proposition 1 guarantees the correctness of this definition. Moreover, it is consistent with the SO-structure defined by its underlying AO-net.

**Proposition 4.** \((T', \prec_{\text{loc}}, \sqsubseteq_{\text{loc}})\) is an so-acyclic relational structure.

**Proof.** Follows from Proposition 2.
As far as the mappings $\epsilon$ and $\iota$ are concerned, $\epsilon$ is the set of stratified poset (or, equivalently, singular step sequences) extensions of a GSO-structure, and $\iota$ is the intersection of the GSO-structures corresponding to a set of stratified posets with the same domain. Thus Fact 2 immediately yields Property 3. Other properties are dealt with later in this section.

The default initial and final markings of $AMON$, as well as its step sequence executions are defined in exactly the same way as for the underlying $AO$-net under the proviso that steps do not contain transitions joined by mutex arcs.

The following results yield more insight into the labelled executions of an activator mutex occurrence net relative to its underlying $AO$-net.

Let $AMON = (P', T', F', Act, Mtx', \ell)$ be an AMO-net and $AON = \text{und}(AMON)$.

**Proposition 5.** $\lambda(AMON) = \{U_1 \ldots U_k \in \lambda(AON) \mid Mtx' \cap \bigcup_i U_i \times U_i = \emptyset\}$. ◦

**Proof.** Follows from the definitions.

**Proposition 6.** Let $\sigma = U_1 \ldots U_k \in \lambda(AON)$ be such that there is no $i \leq k$ for which there exists a partition $U, U'$ of $U_i$ such that $U_1 \ldots U_{i-1}UU'U_{i+1} \ldots U_k \in \lambda(AON)$. Then $\sigma \in \lambda(AMON)$.

**Proof.** By Proposition 5, it suffices to show that, for every $i \leq k$, $(U_i \times U_i) \cap Mtx' = \emptyset$. Suppose this does not hold for some $i \leq k$. Let $\kappa(AON) = (T', \prec, \sqsubseteq)$. From the assumption made about $\sigma$ it follows that $t \sqsubseteq u$, for all distinct $t,u \in U_i$. This, however, contradicts the gso-acyclicity of $\rho_{AMON}$.

**Proposition 7.** $\text{reach}(AMON) = \text{reach}(AON)$.

**Proof.** $(\subseteq)$ Follows from Proposition 5.

$(\supseteq)$ Follows from Proposition 6 and the fact that each step sequence in $\lambda(AON)$ can be ‘sequentialised’ into the form from the formulation of Proposition 6 by splitting the steps into smaller ones.

**Proposition 8.** A marking $M$ belongs to $\text{reach}(AMON)$ iff there are no places $p, p' \in M$ for which $(p, p') \in F' \circ (\preceq_{\text{loc}} \cup \sqsubseteq_{\text{loc}})^* \circ F'$.

**Proof.** Follows from Proposition 7 and Proposition 5.15 in [18].

Figure 6 depicts an AMO-net labelled with places and transitions of the ENIM-system of Figure 4. We have that both $\{l\} \{a\} \{g\} \{r\} \{f\}$ and $\{a\} \{g\} \{f\} \{r\}$ belong to $\phi(\lambda(AMON_0))$, however, $\{l\} \{a\} \{g\} \{f,r\}$ does not.

Now we are ready to introduce a process semantics for ENIM-systems.
Definition 7 (processes of ENIM-systems). A process of ENIM is an AMO-net AMON such that $\text{und}(AMON)$ is a process of $\text{und}(ENIM)$ and, for all $t, u \in T'$, $(t, u) \in \text{Mtx}'$ iff $(\ell(t), \ell(u)) \in \text{Mtx}$. We denote this by $AMON \in \alpha(ENIM)$.

Definition 8 (processes construction). An AMO-net generated by a step sequence $\sigma = U_1 \ldots U_n \in \omega(ENIM)$ is the last net in the sequence $AMON_0, \ldots, AMON_n$ where each $AMON_k = (P_k, T_k, F_k, A_k, M_k, \ell_k)$ is as in Definition 5 except that $M_k = \{(e, f) \in T_k \times T_k \mid (\ell_k(e), \ell_k(f)) \in \text{Mtx}\}$ is an added component. We denote this by $AMON_n \in \pi_{\text{ENIM}}(\sigma)$.

The way in which mutex arcs are added in the process construction entails that some of them may be redundant when, for example, the transitions they join are causally related. As argued in [20], eliminating such redundant mutex arcs (which is possible by analysing paths in the AMO-net) would go against the locality principle which is the basis of the process approach. Indeed, this approach does not remove redundant causalities as this would compromise the local causes and effects in the definition and construction of process nets.

The AMON-net shown in Figure 6 is a process of the ENIM-system of Figure 4 with $\phi(\lambda(AMON_0)) = \{\{l\} \{a\} \{g\} \{f\}, \{l\} \{a\} \{g\} \{r\} \{f\}\}$. Figure 7 shows the result of applying the construction from Definition 8 to the ENIM-system of Figure 4 and one of its step sequences. Note that the resulting AMO-net is isomorphic to that shown in Figure 6.

Fig. 6. An AMO-net $AMON_0$ with labels shown inside places and transitions.

Fig. 7. Process generated for the ENIM-system in Figure 4 and $\sigma = \{l\} \{a\} \{g\} \{r\} \{f\}$. 
Having instantiated the semantical framework for ENIM-systems, we can now formally establish their connection with GSO-structures by proving the remaining Properties 1, 2, and 4. Below we assume that ENIM is an ENIM-system.

**Proposition 9.** Let $\sigma$ be a step sequence of ENIM, AMON an AMO-net, gsos a GSO-structure, and $SPO$ a set of stratified posets with the same domain.

1. $\omega(ENIM), \alpha(ENIM), \lambda(AMON)$ and $\epsilon(gsos)$ are non-empty sets.
2. $\kappa(AON)$ and $\epsilon(SPO)$ are GSO-structures.
3. $\pi_{ENIM}(\sigma)$ comprises an AMO-net.

**Proof.** In what follows, we use the notations introduced throughout this section.

1. We have $\omega(ENIM) \neq \emptyset$ as the empty string is a valid step sequence of ENIM. To show $\alpha(ENIM) \neq \emptyset$ one can take the AMO-net consisting of the initial marking of ENIM with the identity labelling and no transitions. That $\epsilon(gsos) \neq \emptyset$ follows from Fact 2. That $\lambda(AMON) \neq \emptyset$ follows from Proposition 6, $\lambda(AON) \neq \emptyset$ and the fact that each step sequence in $\lambda(AON)$ can be ‘sequentialised’ into the form from the formulation of Proposition 6 by splitting the steps into smaller ones.

2. Follows from Fact 2 and Proposition 1.

3. We have that an element of $\pi_{ENIM}(\sigma)$ with deleted mutex arcs is an AO-net. It therefore suffices to show that the relation $\sqsubseteq_{loc} \circ (Mtx' \cap \sqsubseteq_{loc}) \circ \sqsubseteq_{loc}$ is irreflexive.

Suppose that $(t, t) \in \sqsubseteq_{loc} \circ (Mtx' \cap \sqsubseteq_{loc}) \circ \sqsubseteq_{loc}$. Then there are $t = t_1, \ldots, t_k = t$ such that $(t_i, t_{i+1}) \in \sqsubseteq_{loc}$ for all $i < k$, and $(t_m, t_j) \in M_n$ for some $m < j \leq k$. But this means that $t_1, \ldots, t_k$ have been generated in the same step of the construction, contradicting the definition of executability in ENIM-systems.

**Proposition 10.** Let $\xi \in \omega(ENIM)$ and $AMON \in \pi_{ENIM}(\xi)$.

1. $AMON \in \alpha(ENIM)$.
2. $\xi \in \phi(\lambda(AMON))$.

**Proof.** (1) By Proposition 9(3), AMON is an AMO-net. Moreover, by [18], we have that $\text{und}(AMON) \in \alpha(\text{und}(ENIM))$. Finally, the condition involving mutex arcs follows from the construction in Definition 8.

(2) By [18], $\xi \in \phi(\lambda(\text{und}(AMON)))$. Hence $\xi = \phi(\sigma)$ for some $\sigma = U_1 \ldots U_k \in \lambda(\text{und}(AMON))$. The latter, together with $\xi \in \omega(ENIM)$ and the consistency between mutex arcs in ENIM and AMON, means that there is no mutex arc joining two elements of any $U_i$. Hence, by Proposition 5, $\sigma \in \lambda(AMON)$. Thus $\xi \in \phi(\lambda(AMON))$.

**Proposition 11.** Let $AMON \in \alpha(ENIM)$ and $\xi \in \phi(\lambda(AMON))$.

1. $\xi \in \omega(ENIM)$.
2. $AMON \in \pi_{ENIM}(\xi)$. 
Proof. (1) By [18], \( \xi \in \omega(\text{und}(\text{ENIM})) \). Also there is \( \sigma = U_1 \ldots U_k \in \lambda(\text{AMON}) \) such that \( \xi = \phi(\sigma) \). The latter, together with the consistency between mutex arcs in \( \text{ENIM} \) and \( \text{AMON} \), means that there is no mutex arc joining two elements of any \( U_i \). Hence, by Proposition 3, \( \xi \in \omega(\text{ENIM}) \).

(2) By [18], \( \text{und}(\text{AMON}) \in \pi_{\text{und}(\text{ENIM})}(\xi) \). Moreover, the mutex arcs are added in the same (deterministic) way to the underlying process nets, leading to \( \text{AMON} \in \pi_{\omega(\text{ENIM})}(\xi) \).

Hence Property 2 holds. We then observe that Property 3 is simply Fact 2, and Property 4 is proved below.

**Proposition 12.** Let \( \text{AMON} \) be an \( \text{AMO} \)-net. Then \( \lambda(\text{AMON}) = \epsilon(\kappa(\text{AMON})) \).

**Proof.** We have:

\[
\begin{align*}
\epsilon(\kappa(\text{AMON})) &= \text{ext}(\rho^{\text{AMON}}) = \text{ext}((T', \prec_{\text{loc}}, \sqsubseteq_{\text{loc}}, \text{Mtx}')^{\text{AMO}}) = (\text{Prop. 2}) \\
\{ \text{spo} \in \text{ext}((T', \prec_{\text{loc}}, \sqsubseteq_{\text{loc}})^{\text{AMO}}) | \sim_{\text{spo}} \cap \text{Mtx}' = \varnothing \} &= \\
\{ \text{spo} \in \epsilon(\kappa(\text{AON})) | \sim_{\text{spo}} \cap \text{Mtx}' = \varnothing \} &= \\
\{ \text{spo} \in \lambda(\text{AON}) | \sim_{\text{spo}} \cap \text{Mtx}' = \varnothing \} = (\text{Prop. 5}) \lambda(\text{AMON}).
\end{align*}
\]

Note that we identify stratified posets with their corresponding singular labelled step sequences.

Finally, we can claim the semantical aims for \( \text{ENIM} \)-systems.

**Theorem 1.** Let \( \text{ENIM} \) be an \( \text{ENIM} \)-system, and \( \text{AMON} \) be an \( \text{AMO} \)-net.

\[
\begin{align*}
\alpha(\text{ENIM}) &= \pi_{\text{ENIM}}(\omega(\text{ENIM})) \\
\omega(\text{ENI}) &= \phi(\lambda(\alpha(\text{ENIM}))) \\
\kappa(\text{AMON}) &= \iota(\lambda(\text{AMON})).
\end{align*}
\]

### 7 ENIM-systems and generalised comtraces

Now we are ready to express the behaviour of an \( \text{ENIM} \)-system in terms of \( \text{G} \)-comtraces. First we define the \( \text{G} \)-comtrace alphabet of an \( \text{ENIM} \)-system \( \text{ENIM} = (P, T, F, \text{Inh}, \text{Mtx}, M_{\text{init}}) \) as

\[
\Psi_{\text{ENIM}} = (T, \text{sim}_{\text{ENIM}}, \text{ser}_{\text{ENIM}}, \text{inl}_{\text{ENIM}}),
\]

where the three relations on \( T \) are as follows:

\[
\begin{align*}
(e, f) &\in \text{sim}_{\text{ENIM}} \text{ if } (e, f) \in \text{noc}_{\text{ENIM}} \text{ and } (e, f) \notin \text{Mtx} \\
(e, f) &\in \text{ser}_{\text{ENIM}} \text{ if } (e, f) \in \text{sim}_{\text{ENIM}} \text{ and } e \cap_{\text{loc}} f = \varnothing \\
(e, f) &\in \text{inl}_{\text{ENIM}} \text{ if } (e, f) \in \text{ind}_{\text{ENIM}} \text{ and } (e, f) \notin \text{Mtx},
\end{align*}
\]

and the two auxiliary relations, \( \text{noc}_{\text{ENIM}} \) and \( \text{ind}_{\text{ENIM}} \), are given by:

\[
\begin{align*}
(e, f) &\in \text{noc}_{\text{ENIM}} \text{ if } e \cap_{\text{loc}} f = \varnothing \cap_{\text{loc}} e = \varnothing \cap_{\text{loc}} f = \varnothing \\
(e, f) &\in \text{ind}_{\text{ENIM}} \text{ if } \varnothing \cap e = \varnothing \cap f = \varnothing \cap (e, f) \in \text{noc}_{\text{ENIM}}.
\end{align*}
\]
Thus \( \text{sim}_{\text{ENIM}} \) comprises all pairs of distinct transitions which are neither mutually exclusive nor conflicting (i.e., those with disjoint neighbourhoods w.r.t. normal arcs and disjoint sets of input places and inhibitor places). A pair of transitions \((e, f)\) in \( \text{sim}_{\text{ENIM}} \) can be serialised (in the order \(e f\)) if, in addition, the occurrence of \(e\) does not fill any inhibitor place of \(f\). Finally, two transitions are commutative (interleaving) if they could occur simultaneously as well as in any order were it not the case that they are mutually exclusive due to belonging to \(\text{Mtx}\). Clearly, \(\text{sim}_{\text{ENIM}}\) and \(\text{inl}_{\text{ENIM}}\) are irreflexive and symmetric, \(\text{sim}_{\text{ENIM}} \cap \text{inl}_{\text{ENIM}} = \emptyset\), and \(\text{ser}_{\text{ENIM}} \subseteq \text{sim}_{\text{ENIM}}\). It is worth stressing that all three relations are structurally defined, independent of any marking or concrete dynamic behaviour.

According to \(\psi_{\text{G}}\), the set of all (potential) steps of \(\text{ENIM}\) is given by:

\[
S_{\text{ENIM}} = \{ U \subseteq T \mid \forall a \neq b \in U : (a, b) \in \text{sim}_{\text{ENIM}} \}.
\]

In other words, \(S_{\text{ENIM}}\) is the set of all cliques of the relation \(\text{sim}_{\text{ENIM}}\). Clearly, \(S_{\text{ENIM}}\) is subset closed. The \(G\)-comtrace congruence defined by \(\psi_{\text{G}}\) will be denoted by \(\equiv\), and its equivalence class containing a step sequence \(\sigma\) by \([\sigma]\). The equivalence relation \(\equiv\) on \(\omega(\text{ENIM})\) is generated by the set of equations \(\text{EQ}_{\text{ENIM}} = \text{EQ}_{\text{ser}} \cup \text{EQ}_{\text{inl}}\) where:

\[
\begin{align*}
\text{EQ}_{\text{ser}} &= \{ A = BC \mid A = B \cup C \in S_{\text{ENIM}} \wedge B \times C \subseteq \text{ser}_{\text{ENIM}} \} \\
\text{EQ}_{\text{inl}} &= \{ BA = AB \mid A \in S_{\text{ENIM}} \wedge B \in S \wedge A \times B \subseteq \text{inl}_{\text{ENIM}} \}.
\end{align*}
\]

Whenever \(\sigma \in \omega(\text{ENIM})\), we may refer to \([\sigma]\) as a \(G\)-comtrace of \(\text{ENIM}\). As we shall see next, \(\equiv\) provides a means to add a meaningful structure to \(\omega(\text{ENIM})\).

By Fact 3, we can view \(\equiv\) as being defined through the reflexive, transitive closure of \(\sim_{\text{ENIM}}\), which is the symmetric closure of the relation comprising all pairs \((t, u)\) of step sequences in \(S_{\text{ENIM}}^*\) such that:

- \(t = wAz\) and \(u = wBCz\) with \(B \cup C = A\) and \(B \times C \subseteq \text{ser}_{\text{ENIM}}\), or
- \(t = wABz\) and \(u = wBAz\) where \(A \times B \subseteq \text{inl}_{\text{ENIM}}\).

One can easily check that splitting or combining a step occurring in a step sequence \(\sigma\) of \(\text{ENIM}\) according to \(\text{ser}_{\text{ENIM}}\) and interchanging adjacent occurrences of commutative steps in \(\sigma\) according to \(\text{inl}_{\text{ENIM}}\) yields a step sequence of \(\text{ENIM}\). As a consequence, the set \(\omega(\text{ENIM})\) of step sequences of \(\text{ENIM}\) is consistent with the equivalence \(\equiv\) in the sense that all step sequences equivalent with a step sequence of \(\text{ENIM}\) are themselves step sequences of \(\text{ENIM}\). We can therefore partition the set of step sequences of \(\text{ENIM}\) into (disjoint) \(G\)-comtraces:

**Theorem 2.** \(\omega(\text{ENIM}) = \bigcup_{\sigma \in \omega(\text{ENIM})} [\sigma]\).  

Similarly, the construction of a process \(\pi_{\text{ENIM}}(\sigma)\) from a given step sequence \(\sigma\) of \(\text{ENIM}\) (see Definitions 5 and 8) is not affected when in \(\sigma\) steps are split or combined or adjacent commutative steps are interchanged in accordance with \(\text{ser}_{\text{ENIM}}\) and \(\text{inl}_{\text{ENIM}}\), respectively.

**Proposition 13.** For all \(\sigma, \sigma' \in \omega(\text{ENIM})\), \(\equiv \sigma' \implies \pi_{\text{ENIM}}(\sigma) = \pi_{\text{ENIM}}(\sigma')\).
Hence, for each \( \sigma \in \omega(\text{ENIM}) \), \( \pi_{\text{ENIM}}([\sigma]) = \pi_{\text{ENIM}}(\sigma) \) is well-defined. In this way, we can associate a unique process to each \( g \)-comtrace of \( \text{ENIM} \). As we will see shortly also, conversely, each process of \( \text{ENIM} \) determines a single \( g \)-comtrace.

First, using the above proposition, we prove as a main result that the causal structure associated with (the process of) a step sequence of \( \text{ENIM} \) is the \( \text{GSO} \)-structure of its \( g \)-comtrace.

**Theorem 3.** For every \( \sigma \in \omega(\text{ENIM}) \), \( G_{[\sigma]} = \kappa(\pi_{\text{ENIM}}(\sigma)) \).

**Proof.** (sketch) By Definition 6, we have that:

\[
\kappa(\pi_{\text{ENIM}}(\sigma)) = (\text{occ}(\sigma), \text{loc}_\text{oc}, \text{loc}_\text{oc}, \text{Mtx}' \text{gso}) = (\text{occ}(\sigma), \psi, \gamma \setminus \text{id}_{\text{occ}(\sigma)}) ,
\]

\[
\psi = \alpha\text{sym} \cup \beta\text{sym} \cup \text{Mtx}' \text{gso} \quad \text{and} \quad \gamma = (\text{loc}_\text{oc} \cup \text{loc}_\text{oc})^*,
\]

with \( \alpha = \gamma \circ \text{loc}_\text{oc} \circ \gamma \) and \( \beta = \text{loc}_\text{oc} \circ (\text{Mtx}' \cap \text{loc}_\text{oc}) \circ \text{loc}_\text{oc} \). On the other hand, the \( \text{GSO} \)-structure induced by \( [\sigma] \) is given by:

\[
G_{[\sigma]} = \left( \text{occ}([\sigma]), \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sym}, \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sym} \right).
\]

One can then see that \( G_{[\sigma]} = \kappa(\pi_{\text{ENIM}}(\sigma)) \) as \( \psi = \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sym} \) and \( \gamma \setminus \text{id}_{\text{occ}(\sigma)} = \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sy}.\)

To show the latter equality, suppose first that two different occurrences \( a^i \) and \( b^j \) in \( \text{occ}([\sigma]) \) are such that \( (a^i, b^j) \in \text{loc}_\text{oc} \cup \text{loc}_\text{oc} \). Then, by the definition of \( \text{loc}_\text{oc} \cup \text{loc}_\text{oc} \) (see Figure 3), we have that \( (a, b) \notin \text{ser}_\text{ENIM} \) or \( (b, a) \notin \text{ser}_\text{ENIM} \). Moreover, we have that \( (a, b) \notin \text{ind}_\text{ENIM} \). It therefore follows that in every \( x \in [\sigma] \), the \( i \)-th occurrence of \( a \) will never be after the \( j \)-th occurrence of \( b \). Hence \( (a^i, b^j) \in \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sy} \). Clearly, the same argument is reached if \( (a^i, b^j) \in \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sym} \setminus \text{id}_{\text{occ}(\sigma)} \).

Suppose now that \( (a^i, b^j) \notin \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sym} \) and \( (a^i, b^j) \in \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sy} \). If \( (b^j, a^i) \in \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sy} \) then, from what we already know, \( a^i \) and \( b^j \) must always occur in the same step which means that \( (a^i, b^j) \in \text{loc}_\text{oc}^* \). Let us take any \( p \in a^i \) and \( p' \in b^j \). From \( (a^i, b^j), (b^j, a^i) \notin \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sym} \) and Proposition 8 it follows that there is a reachable marking to which \( p \) and \( p' \) belong. But this means that there is a step sequence belonging to \([\sigma]\) in which the \( j \)-th occurrence of \( b \) comes before the \( i \)-th occurrence of \( a \). Hence \( (a^i, b^j) \notin \bigcap_{x \in [\sigma]} \text{loc}_x^\text{sy} \).

Combining the above results with the consistency (Property 2) and fitting (Property 4) of the \( \text{ENIM} \) process semantics as expressed in Propositions 10 and 12, respectively, yields:

**Theorem 4.** For every \( \sigma \in \omega(\text{ENIM}) \), \( [\sigma] = \phi(\lambda(\pi_{\text{ENIM}}(\sigma))) \).

**Proof.** (\( \subseteq \)) Suppose that \( \sigma' \in [\sigma] \). By Proposition 13, we have that \( \pi_{\text{ENIM}}(\sigma') = \pi_{\text{ENIM}}(\sigma) \). Hence, by Proposition 10, \( \sigma' \in \phi(\lambda(\pi_{\text{ENIM}}(\sigma))) \).

(\( \supseteq \)) Suppose that \( \sigma' \in \phi(\lambda(\pi_{\text{ENIM}}(\sigma))) \). Then, by Property 4, \( \sigma' \in \phi(\kappa(\pi_{\text{ENIM}}(\sigma))) \). Hence, by Theorem 3, \( \sigma' \in \phi(\kappa(G_{[\sigma]}) \mid x \in [\sigma]) \). Thus, by Fact 4, \( \sigma' \in \phi(\{ \text{spo}_{\text{ENIM}}(x) \mid x \in [\sigma] \}) \), and so \( \sigma' \in [\sigma] \).
Thus if we take all step sequences of a process from its default initial to default final marking and apply the labelling, then what we get is exactly its defining trace. To conclude, there exists a a one-to-one correspondence between the $G$-comtraces defined by ENIM and its processes.

8 Concluding remarks

The results presented establish a link between ENIM-systems and trace theory and allow one to identify different observations of concurrent behaviour in a way that is consistent with the causality semantics defined by the operationally defined processes. They contribute to the development of the full causality semantics of the most general elementary net systems model.

Modelling mutually exclusive transitions can be done in PT-nets using self-loops linking mutually exclusive transitions to a place marked with a single token (which has no other arcs attached to it). An alternative would be to use a mutex arc. Though at a modelling level there is no real difference between these two representations, we argued in [20] that at the semantical level the differences can be significant. The point is that mutex arcs represent concurrent histories in a compact way which should have a direct impact on the size of net unfolding used, in particular, for model checking. Intuitively, mutex arc stem from a different philosophy to self-loops. Whereas the latter are related to resource sharing, mutex arcs are derived from semantical considerations and so can provide a more convenient modelling tool.

In our future work we plan to investigate the relationship between mutex arcs and other modelling concepts such as localities [21] and policies [3], also from the point of view of the synthesis of nets where unorderedness does not imply simultaneity of executed actions. We also plan to integrate quotient monoids of step sequences into the semantical framework of [18] outlined in Section 5.

References