A Reversible Perspective on Petri Nets and Event Structures

HERNÁN MELGRATTI, ICC - Universidad de Buenos Aires, Conicet, Argentina
CLAUDIO ANTARES MEZZINA, Dipartimento di Scienze Pure e Applicate, Università di Urbino, Italy
G. MICHELE PINNA, Dipartimento di Matematica e Informatica, Università di Cagliari, Italy

Event structures have emerged as a foundational model for concurrent computation, explaining computational processes by outlining the events and the relationships that dictate their execution. They play a pivotal role in the study of key aspects of concurrent computation models, such as causality and independence, and have found applications across a broad range of languages and models, spanning realms like persistence, probabilities, and quantum computing. Recently, event structures have been extended to address reversibility, where computational processes can undo previous computations. In this context, reversible event structures provide abstract representations of processes capable of both forward and backward steps in a computation. Since their introduction, event structures have played a crucial role in bridging operational models, traditionally exemplified by Petri nets and process calculi, with denotational ones, i.e., algebraic domains. In this context, we revisit the standard connection between Petri nets and event structures under the lenses of reversibility. Specifically, we introduce a subset of contextual Petri nets, dubbed *reversible causal nets*, that precisely correspond to reversible prime event structures. The distinctive feature of reversible causal nets lies in deriving causality from inhibitor arcs, departing from the conventional dependence on the overlap between the post and preset of transitions. In this way, we are able to operationally explain the full model of reversible prime event structures.

CCS Concepts: • **Theory of computation → Concurrency**; *Operational semantics.*

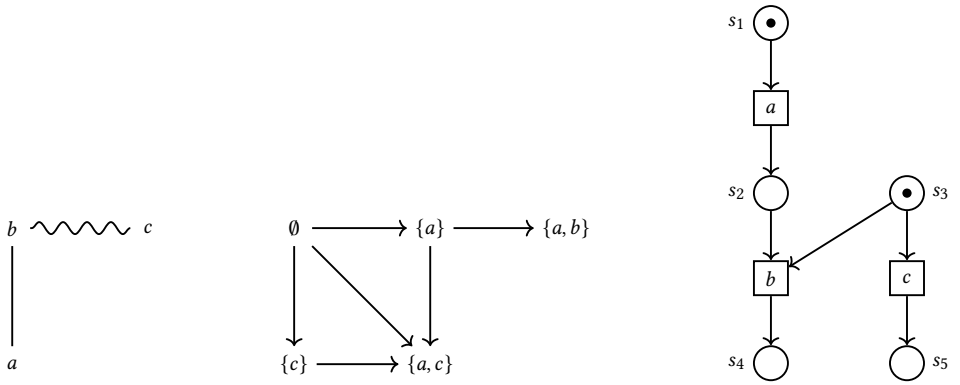
Additional Key Words and Phrases: Event Structures, Petri Nets, Reversibility

ACM Reference Format:
Hernán Melgratti, Claudio Antares Mezzina, and G. Michele Pinna. 2023. A Reversible Perspective on Petri Nets and Event Structures. 1, 1 (July 2023), 38 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Event structures [33] are a well-established model of concurrency, that describe computational processes through event occurrences and constraints that regulate such occurrences (i.e., relations over events). In their simplest form, known as *prime event structures*, the relations are *causality* and *conflict*. Causality dictates the order and precedence of events in valid executions, while conflict denotes the mutual exclusion of events – conflicting events cannot coexist in any valid system execution. Consider a computational process denoted as P, involving three distinct events: a, b, and c, where b causally depends on a, and b and c are in conflict. Describing the behaviour of this system as a prime event structure necessitates defining two binary relations over the set of events: $<$ for causality and $\#$ for conflicts. Specifically, the relations are defined as follows: $< = \{(a, b)\}$ and $\# = \{(b, c), (c, b)\}$. A graphical representation is shown in Figure 1a,

Authors' addresses: Hernán Melgratti, ICC - Universidad de Buenos Aires, Conicet , Buenos Aires, Argentina; Claudio Antares Mezzina, Dipartimento di Scienze Pure e Applicate, Università di Urbino, Urbino, Italy; G. Michele Pinna, Dipartimento di Matematica e Informatica, Università di Cagliari, Cagliari, Italy.



(a) P (b) Transition system (c) N

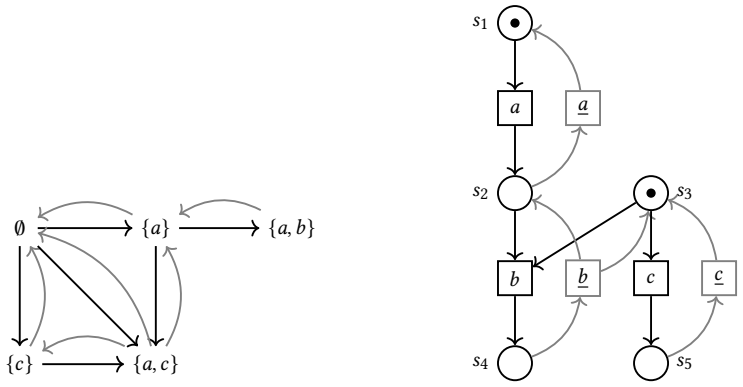
Fig. 1. A simple Prime Event Structure and its associated Petri net

where causality is drawn with straight lines (to be read from bottom to top) and binary conflicts are represented by curly lines.

The behaviour associated with such event structure can be understood in terms of a transition system defined over *configurations* (i.e., sets of events), as illustrated in Figure 1b. For instance, the transition $\emptyset \rightarrow \{a, c\}$ indicates that the initial state $\emptyset$ (i.e., no event has been executed yet) may evolve to the state $\{a, c\}$ by concurrently executing a and c. Moreover, the same configuration can be achieved by initially executing a followed by c (as outlined in the transitions $\emptyset \rightarrow \{a\} \rightarrow \{a, c\}$). Conversely, one can attain the identical configuration by first transitioning through c and then through a (i.e., $\emptyset \rightarrow \{c\} \rightarrow \{a, c\}$). In other words, a and c are concurrent. In contrast, the configuration $\{a, b\}$ can only be reached by executing a before b since the occurrence of b causally depends on the prior execution of a. Note that neither $\{b\}$ nor $\{a, b, c\}$ are configurations because, on the one hand, b cannot occur without a and, on the other hand, b and c cannot occur in the same run of the system.

Event structures were originally proposed by Nielsen, Plotkin and Winskel [23] as an intermediate abstraction in between Scott domains (i.e., a denotational model) and Petri nets (i.e., an operational model). While event structures encompass a set of event occurrences and constraints that govern the occurrence of such events, Petri nets describe the behaviour of a system in terms of the consumption and production of data items (i.e., tokens) from repositories (i.e., places). A Petri net N corresponding to the prime event structure P above, is depicted in Figure 1c. It consists of five places s_1, s_2, s_3, s_4 and s_5, three transitions a, b and c and two tokens (depicted as bullets) respectively placed on s_1 and s_3. The edges connecting places to transitions describe the consumption and production of tokens; e.g., the firing (i.e., execution) of b consumes a token from each s_2 and s_3 and produces a token in s_4. The tokens available in N enable the firing of the transitions a and c but not that of b because there is no token in s_2. That missing token is produced by the firing of a; hence, b can be fired only after a is so. For this reason, b *causally depends* on a. However, b can be fired only if c is not (and vice versa) because each transition requires a token from s_3, which just contains one. In this case, b and c are in *conflict*.

The foundational work in [33] shows a tight connection, established through a chain of correflections, between the category of safe nets and prime event structures. Subsequent research efforts have systematically explored the



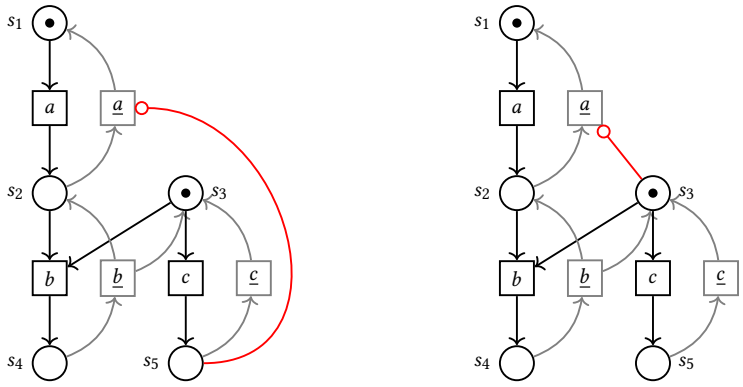
(a) Transition system (b) $\underline{N}$

Fig. 2. Causal-consistent reversible version of N

alignment of distinct incarnations of Petri nets with their respective classes of event structures. Examples of this line of work include [4, 5, 16, 32].

Recently, there has been a noteworthy extension of event structures to accommodate *reversible* concurrent systems, namely a class of systems that have lately received lot of attention because of their applications in different fields [1, 21]. These applications span programming abstractions for reliable systems [7, 12, 17], program analysis and debugging [13], bio-chemical simulation modeling [11], and quantum computing [24]. The distinctive feature of a reversible system is that the execution of actions is liable to be undone. *Reversible prime event structures* (RPES) [26] accommodate the undoing of executed actions by allowing configurations to evolve by removing events. For instance, if c were an undoable event of the event structure P in Figure 1a, then the associated transition system would include the transition $\{a, c\} \rightarrow \{a\}$. This is a disruptive feature in event structures since it breaks the underlying assumption by which configurations evolve by adding events. In fact, if X and Y are two configurations of an *r*PES then $X \rightarrow Y$ does not imply $X \subseteq Y$. As a consequence, the existing approaches to recover Petri nets out of event structures, even the most general ones [31], are not applicable. As a matter of fact, we still lack a procedure to associate a Petri net to a given RPES. Indeed, the approach presented here stands as the first to provide a systematic procedure for associating a Petri net with a given RPES. Previous attempts [18] do this job just for the subclass of *cause-respecting* RPESes, i.e., RPESes that allow the reversing of an event once all events it caused have been reversed. For instance, the transition system associated with a cause-respecting reversible version of the event structure P in Figure 1a is depicted in Figure 2a. Note that each transition $X \rightarrow Y$ is paired with a reversing one $Y \rightarrow X$; consequently, the configuration $\{a, b\}$ can be reversed only by undoing first b and then a, i.e., $\{a, b\} \rightarrow \{a\} \rightarrow \emptyset$. Contrastingly, the transition $\{a, b\} \rightarrow \{b\}$ is not included because it accounts for the reversal of a before the reversal of the event b, which causally depends on a.

As shown in [18], the transition system of a *cause-respecting* RPES can be implemented (concurrently / distributedly) as a Petri net where the undoing of events is achieved via *reversing* transitions, i.e., each transition t (corresponding to some event of the RPES) is accompanied by another transition $\underline{t}$ that undoes the effects of the firing of t, i.e., $\underline{t}$ (i) consumes the tokens produced by t; and (ii) produces the tokens consumed by t. The transition system in Figure 2a is implemented by the net $\underline{N}$ in Figure 2b, which is essentially the extension of N (Figure 1c) with the reversing transitions $\underline{a}$, $\underline{b}$ and $\underline{c}$.

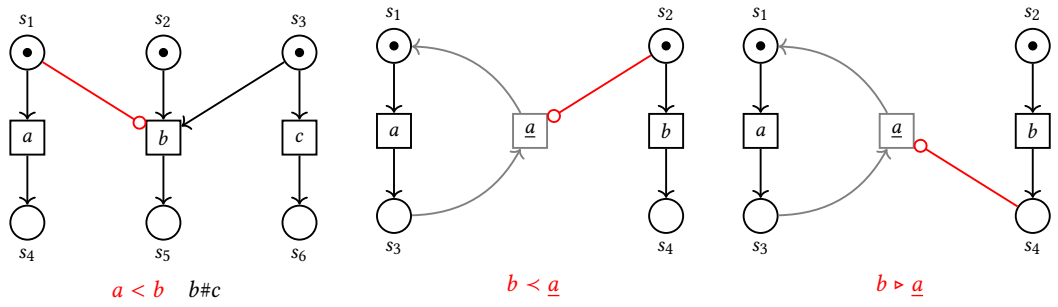


(a) $c \triangleright \underline{a}$ (b) $c \prec \underline{a}$

Fig. 3. Prevention and reverse causality operationally

This approach however falls short when addressing the full expressivity of RPESes, which accommodates different flavours of reversibility [1, 21]. The reversing mechanism of an RPES is defined in terms of two relations (additional to the classical causality and conflicts): *prevention* and *reverse causality*. For instance, an RPES can stipulate that some event can be undone only when some other events have not occurred (prevention). For instance, if we stipulate that c prevents the undoing of a (written $c \triangleright \underline{a}$), then $\{a, c\} \rightarrow \{c\}$ is banned from the transition system even though a is undoable and c does not causally depend on a. We can also specify that a particular event can be undone only when some other events have already occurred (reverse causality). For instance, if c is a reverse cause of a (written $c \prec \underline{a}$) then a cannot be reversed until c occurs, i.e., the transition $\{a\} \rightarrow \emptyset$ is not admissible. We note that these constraints can be translated into Petri nets in the form of contextual arcs; in particular, inhibitor arcs [2, 22] that prevent the firing of a transition if a token is present in some place of the net. For instance, the prevention $c \triangleright \underline{a}$ can be represented in a Petri net with an inhibitor arc in $\underline{a}$, as shown in Figure 3a; the added arc (depicted as $\multimap$) forbids the firing of $\underline{a}$ when s_5 contains a token. Note that s_5 contains a token only when c has been fired, hence $\underline{a}$ cannot be fired if c has occurred. Along the same lines, the reverse causality $c \prec \underline{a}$ can be represented as shown in Figure 3b: in this case, the inhibitor arc is connected to s_3, which will contain a token if c has not been fired. Hence, $\underline{a}$ will be enabled only after c is fired.

Unfortunately, the previous observation is insufficient for capturing the full spectrum of RPESes due to the interplay among causality, prevention and reverse causality. This becomes clear when addressing RPESes enjoying *out-of-causal* order reversibility, which is typical in bio-chemical reactions [27]. Consider again the reversible system in Figure 1a. Assume now that a can be undone also in an out-of-causal order fashion, i.e., a can be reversed independently of the events that it may have caused (which in this case is b). Hence, the transition system would be extended to include the transition $\{a, b\} \rightarrow \{b\}$, in which a is reversed even though b is not, and also $\{b\} \rightarrow \{b, a\}$, in which the minimal event a is executed. When looking at the net $\underline{N}$ in Figure 2b, the transition $\{a, b\} \rightarrow \{b\}$ would require to be able to fire the reversing transition $\underline{a}$ also when the place s_2 does not contain any token (because the firing of b has consumed that token). Hence, a more involved definition of $\underline{a}$ would be needed for handling the undoing of a. Moreover, after reversing a we should be able to fire a again, since the transition system associated to the RPES allows both $\{a\} \rightarrow \emptyset \rightarrow \{a\}$ and $\{a, b\} \rightarrow \{b\} \rightarrow \{a, b\}$. It should be noted that the execution of a has different effects in the two computations above: while the configuration $\{a\}$ allows for the firing of b, the configuration $\{a, b\}$ does not (because, b has been already



(a) N_1 (b) N_2 (c) N_3

Fig. 4. Examples on how inhibitor acts ($\multimap$ arcs) can be used to model causality $<$, reverse causality $\prec$ and prevention $\triangleright$ relation among events

fired). The way in which the causality relation between a and b is described in $\underline{N}$ (Figure 2b) would be insufficient to distinguish the cases above.

In this paper, we take a different approach by observing that inhibitor arcs can be used to model also causality. This simple idea is rendered by the net in Figure 4a, which is an operational counterpart of the event structure P in Figure 1a. The inhibitor arc in Figure 4a is used to model causality among the events a and b. Indeed, b can happen only after a has happened, hence $a < b$. As previously discussed, we represent prevention and reverse causality with inhibitor arcs, as illustrated in Figures 4b and 4c. On the one hand, the inhibitor arc in Figure 4b models the reverse causality $b \prec \underline{a}$, i.e., a can be reversed only when b has been executed. On the other hand, the inhibitor arc in Figure 4c models prevention $b \triangleright \underline{a}$, i.e., the undoing of a cannot be executed if the event b has happened. A liberal usage of inhibitor arcs would not do the work. Therefore, we impose some (structural) constraints on nets to achieve our purpose, namely to identify a subclass of nets with inhibitor arcs that corresponds to reversible prime event structures.

The main contribution of our work, which is a revised and extended version of [19], is the definition of an operational interpretation of RPESes in terms of a proper subclass of Petri nets with inhibitor arcs, called *reversible causal nets*.

The considered subclass is expressive enough to recover the classical operational interpretation of PESes in terms of Petri nets, meaning that this is a conservative extension of the classic notion of occurrence net. The novel way to model dependencies using inhibitor arcs can be fruitfully used to model non standard dependencies among events, e.g., asymmetric conflicts.

The schema below summarizes part of our findings: reversible causal nets (RCN) extend causal nets by introducing reversibility, and RCNs are equivalent to RPESs. On the other hand, prime event structures (PES) are equivalent to CN.

PES ⇠ CN ↪ RCN ⇠ RPES

Structure of the paper

This paper is structured as follows: after setting some notions that will be used in the paper, we start by recalling the basics of prime event structures and reversible prime event structures (Section 3), and those of nets with inhibitor arcs (Section 4). Then, we introduce *causal nets*, which are subclass of nets with inhibitor arcs, and show that they are a suitable counterpart of prime event structures (Section 5). In Section 7, we introduce a *reversible* notion of causal

nets and prove that they fully correspond to reversible prime event structures, thus giving a proper operational model of them. In Section 8 we compare and discuss the relationship between a more classical operational model for prime event structures, namely occurrence nets, with our proposal, and illustrates the reasons that undermine the possibility of generalising occurrence nets to cope with richer models of reversibility as, e.g., out-of-causal order reversibility. Section 9 illustrates some real use of our approach and the last section, after recapping our achievements, provides a brief comparison with the results we extend.

2 PRELIMINARIES

We begin by revisiting some key concepts that will be employed throughout this paper. The symbol $\mathbb{N}$ signifies the set of natural numbers. A *multiset* over a set A is a function $m : A \to \mathbb{N}$. Multisets are assumed to be equipped with the standard operations of union (+) and difference (−). We write $m \subseteq m'$ if $m(a) \leq m'(a)$ for all $a \in A$. The multiset $[\![m]\!]$ is defined such that $[\![m]\!](a) = 1$ if $m(a) > 0$ and $[\![m]\!](a) = 0$ otherwise. We often confuse a multiset m with the set $\{a \in A \mid m(a) \neq 0\}$ when $m = [\![m]\!]$. In such cases, $a \in m$ denotes $m(a) \neq 0$, and $m \subseteq A$ signifies that $m(a) = 1$ implies $a \in A$ for all a. The underlying set of a multiset m, namely the one formed by the elements a with $m(a) > 0$, is precisely $[\![m]\!]$. Additionally, we will employ standard set operations like $\cap$, $\cup$, or $\setminus$. The set of all multisets over A is denoted as μA; the symbol 0 stands for the unique multiset defined such that $[\![0]\!] = \emptyset$.

Given a partial function $f : A \to B$, the domain of f is defined as $dom(f) = \{a \in A \mid \exists b \in B.\ f(a) = b\}$. A sequence of elements of A is a mapping $\rho : \mathbb{N} \to A$ such that if $n \in dom(\rho)$, then $\forall n' < n.\ n' \in dom(\rho)$. A sequence ρ is often represented as $a_1 a_2 \cdots$ where $a_i = \rho(i)$. The length of a sequence ρ, denoted by $len(\rho)$, is the cardinality of its domain, i.e., $len(\rho) = |dom(\rho)|$. A sequence is considered finite when its length is finite. Requiring that a sequence ρ has distinct elements implies asserting that ρ is injective on $dom(\rho)$.

A binary relation $< \subseteq A \times A$ is an *irreflexive partial order* when it is irreflexive and transitive, and we use $\leq$ to denote its reflexive closure. We write $\lfloor a \rfloor_<$ for the set $\{a' \in A \mid a' < a\} \cup \{a\}$ and shall omit the subscript $<$ when it is clear from the context.

3 EVENT STRUCTURES

In this section, we provide an overview of the fundamentals of *prime* event structures. Subsequently, we delve into the *reversible* variant of prime event structures, by following the presentation in [26].

3.1 pre-PES and PES

Pre-*prime event structures* represent a relaxed form of prime event structures where conflict heredity may not necessarily hold. They play a key role in the definition of reversible prime event structures.

Definition 3.1. A *pre-prime event structure* (PPES) is a triple $\mathsf{P} = (E, <, \#)$, where

(1) E is a countable set of *events*;
(2) $\# \subseteq E \times E$ is an irreflexive and symmetric relation, called the *conflict relation*; and
(3) $< \subseteq E \times E$ is an irreflexive partial order, called the *causality relation*, defined such that $\forall e \in E.\ \lfloor e \rfloor_<$ is finite and $\forall e', e'' \in \lfloor e \rfloor_<.\ \neg(e' \# e'')$.

We say a PPES P is a *prime event structure* (PES) when $\#$ is *hereditary* with respect to $<$, i.e., if $e \# e' < e''$ then $e \# e''$ for all $e, e', e'' \in E$.

The third condition prevents events from depending on an infinite number of causes, rendering their execution unfeasible. It also prohibits conflicts between two causes of the same event, which would likewise make the occurrence of the event unfeasible. Moreover, it ensures that an event is not in conflict with any of its causes.

Conflict inheritance is intuitively clear in systems whose state evolves monotonically. If an event e is in conflict with another event e' that causes a third event e'', then e would necessarily be in conflict with e''. This is because e'' cannot occur without e', and the conflict between e and e' prevents the occurrence of e. However, this straightforward relationship breaks down in situations where certain events can be undone, as will be discussed further in Section 3.2.

Example 3.2. Let $P = (E, <, \#)$ be defined such that

$$E = \{a, b, c, d\} \qquad \# = \{(a, b), (b, a)\} \qquad < = \{(b, c)\}$$

It is immediate to check that $\#$ is irreflexive and symmetric, and $<$ is an irreflexive partial order. Furthermore, if $e \in E$ and $e \neq c$, then $\lfloor e \rfloor_< = e$—a finite and conflict-free set. Additionally, $\lfloor c \rfloor_< = \{b, c\}$ is also finite and conflict-free. Hence, P is a PPES. However, P is not a PES because conflicts are not inherited along $<$. Indeed, while $a \# b < c$ holds, the relation $a \# c$ does not. P would be a PES if $\#$ were defined as $\{(a, b), (b, a), (a, c), (c, a)\}$.

A set of events $X \subseteq E$ is *conflict-free* if for every pair of events e and e' in X, it holds that $\neg(e \# e')$. We write $\mathsf{CF}(X)$ if X is conflict-free. Note that for any subset Y of X, if X is conflict-free ($\mathsf{CF}(X)$), then Y is also conflict-free (i.e., $\mathsf{CF}(Y)$).

A configuration is a conflict-free set of events, encompassing, for each event e, the set of its causes, namely $\lfloor e \rfloor_<$—as formally defined below.

Definition 3.3. Let $\mathsf{P} = (E, <, \#)$ be a PPES and $X \subseteq E$ be a subset of events. We say that X is a *configuration* if

(1) $\mathsf{CF}(X)$, and
(2) $\forall e \in X.\ \lfloor e \rfloor_< \subseteq X$.

We denote the set of all configurations of P as $\mathsf{Conf}_{\mathit{PPES}}(P)$. When P is a PES, we write $\mathsf{Conf}_{\mathrm{PES}}(P)$ instead of $\mathsf{Conf}_{\mathit{PPES}}(P)$.

The subsequent definition introduces the concept of (labeled) transitions between sets of events in a PPES.

Definition 3.4. Let $\mathsf{P} = (E, <, \#)$ be a PPES and $X \subseteq E$ a conflict-free set of events. We say $A \subseteq E$ is *enabled* at X if

(1) $A \cap X = \emptyset$ and $\mathsf{CF}(X \cup A)$, and
(2) $\forall e \in A.$ if $e' < e$ then $e' \in X$.

If A is enabled at X, then $X \xrightarrow{A}_{\mathit{PPES}} Y$ with $Y = X \cup A$.

Intuitively, a set of events that are not present in X is considered enabled if there are no conflicts among them or with the events in X, and all their causes are already included in X. It is worth observing that condition (2) in the preceding definition can be formulated as follows: for every event e in A, it holds that $\lfloor e \rfloor_< \setminus \{e\} \subseteq X$, i.e., all its causes are in X.

Example 3.5. Consider the PPES P in Example 3.2. The sets $\{a\}$, $\{b\}$, $\{d\}$, $\{a, d\}$, and $\{b, d\}$ are all enabled at $\emptyset$ because they are all conflict-free and they contain just minimal elements (according to $<$). This enables the derivation of transitions such as $\emptyset \xrightarrow{\{a\}}_{\mathit{PPES}} \{a\}$ and $\emptyset \xrightarrow{\{b,d\}}_{\mathit{PPES}} \{b, d\}$. Conversely, neither $\{a, b\}$ nor $\{c\}$ are enabled at $\emptyset$. The former, because a and b are in conflict; the latter because $\emptyset$ does not contain b, which is a cause of c. Moreover, $\{c\}$ is enabled at $\{b\}$, because the unique cause of c is b; consequently, $\{b\} \xrightarrow{\{c\}}_{\mathit{PPES}} \{b, c\}$ holds.

The notion of configurations of a PPES can be reformulated based on the transitions of enabled events.

Definition 3.6. Let $\mathsf{P} = (E, <, \#)$ be a PPES. A set of events $X \subseteq E$ is a *reachable configuration* if it is conflict-free, i.e., $\mathsf{CF}(X)$, and there exists a sequence $A_1, A_2, A_3, \ldots$ of nonempty set of events such that

(1) $X_0 = \emptyset$, $X_i = \bigcup_{j \leq i} A_i$,
(2) $X_i \xrightarrow{A_i}_{\text{PES}} X_{i+1}$ for all i, and
(3) $X = \bigcup_i A_i$.

In simpler terms, a reachable configuration is constructed step by step, adding enabled sets of events in each step.

Proposition 3.7. *Let* $\mathsf{P} = (E, <, \#)$ *be a PPES and* $X \subseteq E$*. Then,* X *is a configuration iff it is a reachable configuration.*

Proof. $\Rightarrow$) We need to demonstrate that a reachable configuration X is indeed a configuration. Given that X is a reachable configuration, there exists a sequence $A_1, A_2, A_3, \ldots$ of sets of events such that $X = \bigcup_i A_i$, $X_{i-i} = \bigcup_{j<i} A_j$ and $X_{i-1} \xrightarrow{A_i}_{p\text{PES}} X_i$. Since X is conflict-free by definition, we need to show that $\lfloor e \rfloor_< \subseteq X$ for all $e \in X$. Suppose $e \in A_i$ for some i. As X is a reachable configuration, we have $X_{i-1} \xrightarrow{A_i}_{p\text{PES}} X_i$ and $X_{i-1} \subseteq X$. Due to $X_{i-1} \xrightarrow{A_i}_{p\text{PES}} X_i$, all $e' < e$ are in X_{i-1} and, consequently, in X. Thus, the thesis holds.

$\Leftarrow$) We need to show that a configuration X is indeed a reachable configuration. As X is a configuration, its elements can be totally ordered with respect to $<$, resulting in the sequence $e_0, e_1, e_2, \ldots$. By defining $A_i = \{e_i\}$, we demonstrate the thesis. In fact each subset of X is conflict-free, and for all i, $X_{i-1} = \{e_0, e_1, e_2, \ldots, e_{i_1}\}$ contains all events preceding e_i. Additionally, $X_{i-1} \cup A_i = X_i$ is conflict-free, and therefore, $X_{i-1} \xrightarrow{A_i}_{p\text{PES}} X_i$.

□

Example 3.8. Consider once again the PPES P in Example 3.2. It is noteworthy that certain conflict-free sets of events do not correspond to states in the computation of P. For example, the set $\{c, d\}$—despite being conflict-free—is not a configuration of P. This is due to the fact that it cannot be reached from the initial state $\emptyset$: the introduction of c to a configuration necessitates the presence of b.

The following definition and result from [26] highlight that we can reconstruct a PESfrom a PPES by enforcing the heredity of conflicts.

Definition 3.9. Let $\mathsf{P} = (E, <, \#)$ be a PPES. Then $\mathsf{hc}(\mathsf{P}) = (E, <, \sharp)$ is the *hereditary closure* of P, where $\sharp$ is inductively defined by the following rules

$$\frac{e \# e'}{e \sharp e'} \qquad \frac{e \sharp e' \quad e' < e''}{e \sharp e''} \qquad \frac{e' \sharp e}{e \sharp e'}$$

Proposition 3.10. *Let* $\mathsf{P} = (E, <, \#)$ *be a PPES, then*

- $\mathsf{hc}(\mathsf{P}) = (E, \leq, \sharp)$ *is a* PES*,*
- *if* P *is a* PES*, then* $\mathsf{hc}(\mathsf{P}) = \mathsf{P}$*, and*
- $\mathsf{Conf}_{p\text{PES}}(\mathsf{P}) = \mathsf{Conf}_{\text{PES}}(\mathsf{hc}(\mathsf{P}))$.

3.2 Reversible prime event structures

We now revisit the concept of a *reversible prime event structure* presented in [26]. Reversible event structures extend PESes by introducing the possibility for *some* events to be *reversible* or *undoable*. A reversible event, denoted as u,

implicitly possesses a corresponding reversing event, symbolized as $\underline{u}$, capable of undoing its effects. In other words, the execution of u followed by $\underline{u}$ is indistinguishable from no execution at all. Due to this characteristic, the configurations of a reversible prime event structure may not evolve monotonically, as reversible events have the ability to disappear because of the execution of their corresponding reversing events.

To accommodate various nuances of reversibility, a reversible prime event structure incorporates two relations known as *prevention* and *reverse causality*. These relations define the manner in which reversing events can be executed.

Definition 3.11. A *reversible prime event structure* (RPES) is a tuple $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ where $(E, <, \#)$ is a PPES, $U \subseteq E$ are the *reversible/undoable* events (with corresponding reverse events being denoted by $\underline{U} = \{\underline{u} \mid u \in U\}$ and disjoint from E, i.e., $\underline{U} \cap E = \emptyset$) and

(1) $\prec \subseteq E \times \underline{U}$ is the *reverse causality* relation; which is defined such that $u \prec \underline{u}$ for all $u \in U$; and moreover $\{e \in E \mid e \prec \underline{u}\}$ is finite and conflict-free for all $u \in U$,
(2) $\triangleright \subseteq E \times \underline{U}$ is the *prevention* relation defined such that $\prec \cap \triangleright = \emptyset$,
(3) the *sustained causation* $\ll$ is a transitive relation defined such that if $e \ll e'$ then
   - $e < e'$,
   - if $e \in U$ then $e' \triangleright \underline{e}$,
(4) $\#$ is hereditary with respect to $\ll$: if $e \# e' \ll e''$, then $e \# e''$.

The reverse causality relation prescribes the events necessary for the execution of each reversing event. In other words, $e \prec \underline{u}$ signifies that $\underline{u}$ can be executed (or equivalently, u can be undone) only when the event e is present. Therefore, the condition $u \prec \underline{u}$ asserts that an event u can be undone solely when it is present in a configuration. Conversely, the prevention captures cases where an event can be reversed only if some other event is absent in the configuration. Specifically, $e \triangleright \underline{u}$ denotes that u can be reversed only if e is not present in the configuration.

Despite the fact that the underlying structure $(E, <, \#)$ is a PPES, where conflicts may not necessarily be inherited through causality, the definition mandates that conflicts must be inherited through the newly introduced *sustained causation* relation $\ll$. This relation, which is coarser than $<$, accommodates situations where the causes of certain events may vanish from a configuration, as illustrated in Example 3.21.

Example 3.12. Let $\mathsf{P}_1 = (E, U, <, \#, \prec, \triangleright)$ be an RPES defined such that

$$E = \{a, b, c, d\} \qquad U = \{b, c\}$$
$$< = \{(b, c)\} \qquad \# = \{(a, b), (b, a), (a, c), (c, a)\}$$
$$\prec = \{(b, \underline{b}), (c, \underline{c}))\} \qquad \triangleright = \{(c, \underline{b})\}$$

While b and c are reversible in P_1, a and d are not because $a, d \notin U$. Moreover, there is a causal dependency between b and c ($b < c$); and a is in conflict with both b and c because of the definition of $\#$. As a result, d is concurrent w.r.t. a, b and c. In the reverse causality relation, each reversible event is undone only if it has been executed (i.e., $b \prec \underline{b}$ and $c \prec \underline{c}$). The prevention relation stipulates that b cannot be reversed if c has been executed, i.e., $c \triangleright \underline{b}$, which is typical of causal reversible models. In this example, sustained causation coincides with causality ($\ll = <$): specifically, $b \ll c$ holds because $b < c$ and $c \triangleright \underline{b}$ implies so. Also, it is important to note that conflicts are inherited along sustained causation and, consequently, along causality, as demonstrated by $a \# b < c$ and $a \# c$.

Example 3.13. Consider $\mathsf{P}_2 = (E, U, <, \#, \prec, \triangleright)$, a variant of P_1 obtained by extending the definition of $\prec$ with the pair $(d, \underline{c})$. In this scenario, the event c can only be reversed after the execution of d. This illustrates a case of non-causal

reversibility, where concurrency interferes with reversibility: the reversal of c is contingent on the execution of a concurrent (and hence, unrelated) event.

The following definition extends the notion of transitions between sets of events to account for reversing events.

Definition 3.14. Let $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ be an rPES and $X \subseteq E$ a conflict-free set of events, i.e., $\mathsf{CF}(X)$. For $A \subseteq E$ and $B \subseteq U$, we say that $A \cup \underline{B}$ is *enabled* at X if

(1) $A \cap X = \emptyset$, $B \subseteq X$ and $\mathsf{CF}(X \cup A)$,
(2) $\forall e \in A, e' \in E$. if $e' < e$ then $e' \in X \setminus B$,
(3) $\forall e \in B, e' \in E$. if $e' \prec \underline{e}$ then $e' \in X \setminus (B \setminus \{e\})$,
(4) $\forall e \in B, e' \in E$. if $e' \triangleright \underline{e}$ then $e' \notin X \cup A$.

If $A \cup \underline{B}$ is enabled at X then $X \xrightarrow{A \cup \underline{B}}_{r\text{PES}} Y$ with $Y = (X \setminus B) \cup A$.

Example 3.15. In the rPES P_1 from Example 3.12, we have, e.g., the following transitions:

- $\emptyset \xrightarrow{\{a\}}_{r\text{PES}} \{a\} \xrightarrow{\{d\}}_{r\text{PES}} \{a, d\}$,
- $\emptyset \xrightarrow{\{b,d\}}_{r\text{PES}} \{b, d\} \xrightarrow{\{\underline{b}\}}_{r\text{PES}} \{d\}$, and
- $\emptyset \xrightarrow{\{b\}}_{r\text{PES}} \{b\} \xrightarrow{\{c\}}_{r\text{PES}} \{b, c\} \xrightarrow{\{d,\underline{c}\}}_{r\text{PES}} \{b, d\} \xrightarrow{\{c\}}_{r\text{PES}} \{b, c, d\}$.

Now, consider the rPES P_2 presented in Example 3.13. Notice that the set $\{d, \underline{c}\}$ is not enabled at $\{b, c\}$. Despite the presence of c, the reverse causality $d \prec \underline{c}$ indicates that d is also required for the reversal of c.

The reachable configurations of an rPES are the sets of events that can be reached from the empty set by performing events or undoing previously performed events, as stated below.

Definition 3.16. Let $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ be an rPES and $X \subseteq E$ a conflict-free set of events, i.e., $\mathsf{CF}(X)$ holds. We say that X is a *(reachable) configuration* if there exist two sequences of sets A_i and B_i, for $i = 1, \ldots, n$, such that

(1) $A_i \subseteq E$ and $B_i \subseteq U$ for all i, and
(2) $X_i \xrightarrow{A_i \cup \underline{B_i}}_{r\text{PES}} X_{i+1}$ for all i with $X_1 = \emptyset$ and $X_{n+1} = X$.

The set of configurations of P is denoted by $\mathsf{Conf}_{r\text{PES}}(\mathsf{P})$.

Definition 3.17. Let P_1 and P_2 be two rPESes. We say that they are *equivalent*, written $\mathsf{P}_1 \equiv \mathsf{P}_2$, if they have the same set of configurations, i.e., $\mathsf{Conf}_{r\text{PES}}(\mathsf{P}_1) = \mathsf{Conf}_{r\text{PES}}(\mathsf{P}_2)$.

Example 3.18. Consider the rPES P_1 introduced in Example 3.12. The set of configurations of P_1 is

$$\mathsf{Conf}_{r\text{PES}}(\mathsf{P}_1) = \{\emptyset, \{a\}, \{b\}, \{d\}, \{a, d\}, \{b, d\}, \{b, c\}, \{b, c, d\}\}$$

(Refer to Example 3.15 for the corresponding sequences of transitions.)

3.3 Reversing disciplines

The interplay between the *forward* relations (causality and conflicts) and the *backward* relations (reverse causality and prevention) in rPESes determines different ways of reversing events, referred to as reversing disciplines. As an example, consider the rPES P_1 in Example 3.12, where the undoing of the event b is prevented by the event c, which causally depends on b. This scenario illustrates a causal reversing discipline, where an event can only be reversed after the events it has caused have been reversed.

Definition 3.19. Let $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ be an RPES. We say that P is

- *cause-respecting* if for any $e, e' \in E$, if $e < e'$ then $e \ll e'$;
- *causal* if for any $e \in E$, $u \in U$ we have
  (1) $e \prec \underline{u}$ iff $e = u$, and
  (2) $e \triangleright \underline{u}$ iff $u < e$;
- *inverse cause-respecting* if for any $e \in E$, $u \in U$, if $e < u$ then $e \triangleright \underline{u}$; and
- *inverse causal* if for any $e \in E$, $u \in U$ we have
  (1) $e \prec \underline{u}$ iff $e = u$, and
  (2) $e \triangleright \underline{u}$ iff $e < u$.

In a cause-respecting RPES, sustained causality coincides with the causality relation, but the undoing of an event may depend, not only on the reversing of those that it has caused, but also on others, as illustrated in P_2 of Example 3.13. Conversely, P_1 in Example 3.12 is an example of a causal RPES. In an inverse cause-respecting RPES, an event can only be reversed after all its causes have been undone.

The following example illustrates an inverse-cause-respecting RPES.

Example 3.20. Let $\mathsf{P}_3 = (E, U, <, \#, \prec, \triangleright)$ be an RPES defined such that

$$E = \{a, b, c\} \qquad U = \{a, b, c\}$$
$$< = \{(a, b), (a, c), (b, c)\} \qquad \# = \emptyset$$
$$\prec = \{(a, \underline{a}), (b, \underline{b}), (c, \underline{c})), (c, \underline{a})\} \qquad \triangleright = \{(a, \underline{b}), (a, \underline{c}), (b, \underline{c})\}$$

Here, there is just a causal dependency between a and b ($a < b$). Since $a \triangleright \underline{b}$, to reverse b we must first reverse a. This RPES is inverse-cause-respecting but not inverse causal, because we also have $c \prec \underline{a}$. Removing this constraint would make P_3 inverse causal as well.

The out-of-causal-order reversibility of RPESes poses a significant challenge when trying to integrate it into an operational model, as illustrated by the next example. It is worth noting, as observed in various papers (e.g., [26]), that there are examples of biochemical systems where the *computations* exhibit this characteristic. Interested readers can refer to [26] for more details.

Example 3.21. Consider $\mathsf{P}_4 = (E, U, <, \#, \prec, \triangleright)$ as a variant of the RPES P_2 in Example 3.13, defined as follows

$$E = \{a, b, c, d\} \qquad U = \{b, c\}$$
$$< = \{(b, c)\} \qquad \# = \{(a, b), (b, a)\}$$
$$\prec = \{(b, \underline{b}), (c, \underline{c}), (d, \underline{c})\} \qquad \triangleright = \emptyset$$

In this scenario, the induced sustained causation is empty, meaning $b \not\ll c$ holds because $b < c$ does so and $c \triangleright \underline{b}$ does not. Since $c \triangleright \underline{b}$ is not satisfied in P_4, the reversal of b can occur even after executing c, which, in turn, depends causally on b. For instance, we can observe the sequence: $\emptyset \xrightarrow{\{b\}}_{\text{RPES}} \{b\} \xrightarrow{\{c\}}_{\text{RPES}} \{b, c\} \xrightarrow{\{\underline{b}\}}_{\text{RPES}} \{c\} \xrightarrow{\{a\}}_{\text{RPES}} \{a, c\}$. It is important to note that the configuration $\{a, c\}$ would be prohibited in standard PESes because $a \# b < c$ holds. However, RPESes permit such configurations since conflicts are not necessarily inherited through causality, crucial for accommodating out-of-causal-order reversibility.

The previous examples highlight the primary challenges to address when formulating a net semantics for RPESes. These challenges can be summarised as follows:

- configurations may drop events during computation, i.e., $\{b, c\} \xrightarrow{\{b\}}_{r\text{PES}} \{c\}$ in P_4;
- reachable configurations may not necessarily encompass all the causes; for instance, in P_4, the configuration $\{c\}$ does not include b despite $b < c$
- causes can be re-enabled; for example, b is enabled at $\{c\}$ in P_4, but the re-execution of a disables b again, i.e., b is not enabled at $\{a, c\}$;
- conflicts are not inherited through causality, as in P_4;
- reversibility induces dependencies on concurrent events, for example, c in P_2 can be reversed only after the concurrent event d has been executed.

4 PETRI NETS WITH INHIBITOR ARCS

We summarize the fundamentals of Petri nets with inhibitor arcs following the presentation in [2, 22].

Definition 4.1. A *Petri net* is a 4-tuple $N = \langle S, T, F, \mathsf{m}\rangle$ where S is a set of *places*, T is a set of *transitions* with $S \cap T = \emptyset$, $F \subseteq (S \times T) \cup (T \times S)$ is the *flow* relation, and $\mathsf{m} \in \mu S$ is the *initial marking*.

Definition 4.2. A *Petri net with inhibitor arcs* (IPT for short) is a tuple $N = \langle S, T, F, I, \mathsf{m}\rangle$, where $\langle S, T, F, \mathsf{m}\rangle$ is a Petri net, and $I \subseteq S \times T$ is the *inhibiting* relation.

Given an IPT $N = \langle S, T, F, I, \mathsf{m}\rangle$ and $x \in S \cup T$, the *pre-* and *postset* of x are respectively defined as the (multi)sets ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$. If $x \in S$, then ${}^\bullet x \in \mu T$ and $x^\bullet \in \mu T$; analogously, if $x \in T$, then ${}^\bullet x \in \mu S$ and $x^\bullet \in \mu S$. The *inhibitor set* of a transition t is the (multi)set ${}^\circ t = \{s \mid (s, t) \in I\}$. The definitions of ${}^\bullet\cdot, \cdot^\bullet, {}^\circ\cdot$ generalise straightforwardly to multisets of transitions.

Example 4.3. Consider the simple IPT N_1 depicted in Figure 5, which consists of six places depicted as circles and three transitions drawn as boxes. The flow relation is represented by black arrows, while the inhibitor relation is shown by red lines ended with a small circle. The initial marking $\mathsf{m} = \{s_1, s_2, s_3\}$ is represented by the bullets drawn within the corresponding places. Let us consider the transition b, which consumes tokens from s_2 and s_3, produces a token in s_5, and is inhibited by s_1. Hence, its pre-, post- and inhibiting sets are respectively ${}^\bullet b = \{s_2, s_3\}$, $b^\bullet = \{s_5\}$, and ${}^\circ b = \{s_1\}$.

A (multiset of) transition(s) $A \in \mu T$ is *enabled at a marking* $m \in \mu S$, written $m\,[A\rangle$, whenever ${}^\bullet A \subseteq m$ and $\forall s \in [\![{}^\circ A]\!]$. $m(s) = 0 \ \wedge \ A^\bullet(s) = 0$. The last condition requires the absence of tokens in all places connected via inhibitor arcs to the transitions in $[\![A]\!]$. Observe that the multiset 0 is enabled at every marking. A (multiset of) transition(s) A enabled at a marking m can *fire* and its firing produces the marking $m' = m - {}^\bullet A + A^\bullet$. The firing of A at

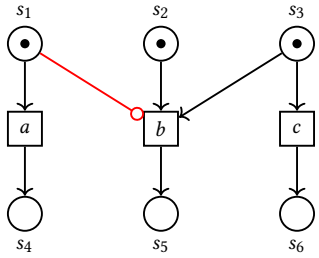


Fig. 5. N_1

a marking m is denoted by $m\,[A\rangle\,m'$. We assume that each transition t of an IPT N is defined such that ${}^\bullet t \neq \emptyset$, i.e., it cannot fire *spontaneously* in an uncontrolled manner without consuming tokens.

Example 4.4. Consider the IPT introduced in Example 4.3 and note that both a and c are enabled at the initial marking m. On the contrary, b is not enabled because its inhibitor place s_1 contains a token. The firing of a produces the marking $m' = \{s_2, s_3, s_4\}$, denoted as $\mathsf{m}\,[a\rangle\,m'$. In this marking, b becomes enabled since its preset s_2 and s_3 are marked, while its inhibitor place s_1 is not.

We now give a precise definition of firing sequences, reachable markings and states of a IPT in terms of sequences of firings. Given an IPT $N = \langle S, T, F, I, \mathsf{m}\rangle$, a *firing sequence* (shortened as fs) of N is a sequence σ of markings such that for each $0 \leq i < \mathit{len}(\sigma)$ there is a multiset of transitions A_i enabled at $\sigma(i)$ and $\sigma(i)\,[A_i\rangle\,\sigma(i+1)$. A firing sequence σ is written as $m_0\,[A_0\rangle\,m_1 \cdots m_n\,[A_n\rangle\,m_{n+1} \cdots$ and we use $\mathit{start}(\sigma)$ to indicate its initial marking $\sigma(0) = m_0$; should σ be finite (i.e., $\mathit{len}(\sigma) < \infty$), $\mathit{lead}(\sigma)$ designates its final marking $\sigma(\mathit{len}(\sigma))$. We say that σ starts at a marking m if $\mathit{start}(\sigma) = m$, and let $\mathcal{R}_m^N$ denote the set of all firing sequences of the IPT N that start at m. A marking m is *reachable* in N iff there exists an fs $\sigma \in \mathcal{R}_{\mathsf{m}}^N$ such that $m = \sigma(i)$ for some $i \leq \mathit{len}(\sigma)$. The set of all reachable markings of N is $\mathcal{M}_N = \bigcup_{\sigma \in \mathcal{R}_{\mathsf{m}}^N} \{\sigma(i) \mid i \leq \mathit{len}(\sigma)\}$. Given an fs σ, $\mathcal{X}_\sigma$ is the set of all sequences of multisets of transitions that *agree* with σ, namely the set $\{\theta \mid \mathit{len}(\theta) = \mathit{len}(\sigma) \ \wedge\ (\sigma(i)\,[\theta(i)\rangle\,\sigma(i+1)\ \mathit{with}\ i < \mathit{len}(\sigma))\}$, and $X_\sigma = \{\sum_{i=1}^{\mathit{len}(\theta)} \theta(i) \mid \theta \in \mathcal{X}_\sigma\}$ for the set of multisets of transitions associated to an fs. Each multiset of X_σ is a *state* of the net and write $\mathrm{St}(N) = \bigcup_{\sigma \in \mathcal{R}_{\mathsf{m}}^N} X_\sigma$ for the set of states of the net N, and $\theta \in \mathcal{X}_\sigma$ is an *execution* of the net.

Definition 4.5. We say that N_1 and N_2 are *equivalent*, written $N_1 \equiv N_2$, if they have the same set of states, i.e., $\mathrm{St}(N_1) = \mathrm{St}(N_2)$.

Example 4.6. The set of reachable markings of the IPT in Example 4.3 is $\mathcal{M}_{N_1} = \{\mathsf{m}, \{s_2, s_3, s_4\}, \{s_1, s_2, s_6\}, \{s_4, s_5\}, \{s_2, s_4, s_6\}\}$ while its set of states is $\mathrm{St}(N_1) = \{\emptyset, \{a\}, \{c\}, \{a, b\}, \{a, c\}\}$.

An IPT N is *safe* if every reachable marking is a set, i.e., $\forall m \in \mathcal{M}_N . m = [\![m]\!]$. Henceforth, our focus will be exclusively safe IPT.

For a given net $N = \langle S, T, F, \mathsf{m}\rangle$, we denote the transitive closure of F as $<_N$. The net N is *acyclic* if the relation $\leq_N$ is a partial order.

5 CAUSAL NETS

We now restrict our attention to representing causal dependencies through inhibitor arcs. We introduce a class of contextual Petri nets, dubbed *causal nets*, where causality is determined by the inhibiting relation rather than the typical flow relation. The main result in this section (Theorem 6.9) states that causal nets are an adequate operational counterpart of PESes. As a matter of fact, they are tightly connected with occurrence nets (as discussed in Section 8).

We define the relation $\lessdot$ between transitions t, t' of an IPT. We stipulate $t \lessdot t'$ iff ${}^\bullet t \cap {}^\circ t' \neq \emptyset$, i.e., the firing of t consumes (at least) one of the tokens that inhibit the firing of t'. Similarly, we define $\natural$ by $t \natural t'$ iff ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$.

Definition 5.1. Let $C = \langle S, T, F, I, \mathsf{m}\rangle$ be an IPT. C is a *pre-causal* net (PCN) if the following conditions are satisfied:

(1) $<_C \cap (T \times T) = \emptyset$;
(2) $[\![T^\bullet]\!] = T^\bullet$;
(3) $\forall t \neq t' \in T.\ {}^\bullet t \cap {}^\bullet t' \cap {}^\circ T = \emptyset$;
(4) $\forall t \in T.\ {}^\circ t$ is finite;

(5) $\lessdot$ is an irreflexive partial order;
(6) $\forall t', t'' \in \lfloor t \rfloor_{\lessdot}.\ t' \natural t'' \Rightarrow\ t' = t''$; and
(7) $\mathsf{m} = {}^\bullet T$ and ${}^\circ T \subseteq \mathsf{m}$.

We say a PCN C is a *causal* net (CN) if it also satisfies the following condition

$$\forall t, t', t'' \in T.\ t \natural t' \ \wedge\ t' \lessdot t'' \ \Rightarrow\ t \natural t''.$$

The conditions imposed on CNs share motivations with those posed on occurrence nets, unravel nets [5, 6, 28] and flow nets [4], which are aimed at explaining computations without resorting to firing sequences. The first condition, alternatively expressed as $\forall t \in T.\ \forall s \in {}^\bullet t.\ {}^\bullet s = \emptyset$, signifies that causal dependencies within CNs do not originate from the flow relation. This is evident as $t^\bullet \cap {}^\bullet t' = \emptyset$ holds for all $t, t' \in T$. The second condition guarantees the absence of backward conflicts, meaning that a place is, at most, part of the postset of a single transition. The third condition asserts that any place present in the preset of at least two transitions cannot be connected via inhibitor arcs to other places in the net. This restriction prohibits or-causality, where the firing of a transition could have multiple distinct causes. Together with the fourth requirement, these conditions ensure that each transition has a finite set of causes, specifically, the set $\{t' \in T \mid t' \lessdot t\}$ is finite. The fifth condition prevents the occurrence of cycles in the dependencies arising from inhibitor arcs. Given that $\lessdot$ is irreflexive, it ensures that none of the transitions is blocked, i.e., ${}^\bullet t \cap {}^\circ t = \emptyset$ for all t. The sixth condition bears resemblance to the prerequisite of PPES, ensuring local inheritance of conflicts. In other words, any pair of different transitions t', t'' in $\lfloor t \rfloor_{\lessdot}$ are not in *conflict*. The last requirement states that the preset of all transitions are initially marked ($\mathsf{m} = {}^\bullet T$). Additionally, ${}^\circ T \subseteq \mathsf{m}$, combined with the first requirement, implies that inhibitor places cannot be part of the postset of any transition.

The supplementary condition for CNs enforces the inheritance of conflicts along the relation $\lessdot$, where conflicts are represented by shared input places.

Example 5.2. The IPT C_1, illustrated in Figure 6a, is a PCN net, as detailed below. Conditions 1 and 2 of Definition 5.1 are satisfied because each place in the postset of a transition has precisely one incoming and no outgoing arcs. The only place belonging to the preset of at least two transitions is s_2, yet there is no inhibitor arc connected to s_2. Therefore, Condition 3 is also met. Condition 4 is immediately satisfied since there is only one inhibitor arc. The induced causality relation is the irreflexive partial order $\lessdot = \{(b, c)\}$, satisfying Condition 5. If $t \in T$ and $t \neq c$, then $\lfloor t \rfloor_{\lessdot} = \{t\}$. For $t = c$, we have $\lfloor c \rfloor_{\lessdot} = \{b, c\}$, and ${}^\bullet b$ and ${}^\bullet c$ are disjoint, fulfilling Condition 6. Condition 7 is immediate. We remark that C_1 is not a CN because conflicts are not inherited along $\lessdot$. Specifically, $a \natural b$ and $b \lessdot c$, yet we do not have $a \natural c$. In contrast, the IPT C_2 in Figure 6b, which explicitly captures the conflict between a and c, is a CN.

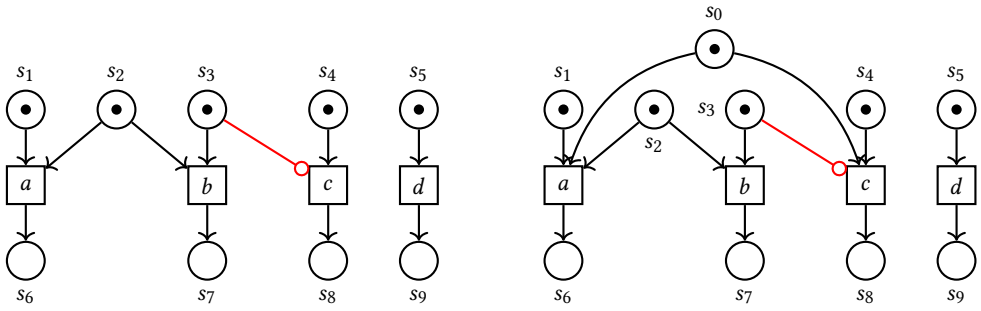
The initial condition outlined in Definition 5.1 asserts that $t^\bullet \cap {}^\bullet t = \emptyset$ for any pair of distinct transitions t and t'. As a consequence, each transition is restricted to being executed at most once. This limitation arises from the fact that its preset cannot be marked again after firing, as formally expressed below.

Proposition 5.3. *Let $C = \langle S, T, F, I, \mathsf{m} \rangle$ be a CN. For every $t \in T$ and every $X \in \mathrm{St}(C)$, it holds that $X(t) \leq 1$.*

Proof. Since ${}^\bullet T = \mathsf{m}$ and $t^\bullet \cap {}^\bullet t = \emptyset$ for any pair of transitions $t, t' \in T$, it is evident that each transition can be fired at most once. □

Definition 5.4. Let $C = \langle S, T, F, I, \mathsf{m} \rangle$ be a PCN. A set of transitions $X \subseteq T$ is a *configuration* of C if:

(1) $\forall t, t' \in X.t \natural t' \Rightarrow\ t = t'$ (*conflict freeness*), and



(a) C_1 (b) C_2

Fig. 6. Two pre-causal nets

(2) $\forall t \in X.\ \lfloor t \rfloor_{\lessdot} \subseteq X$ (*left closedness* with respect to $\lessdot$).

The set of configurations of a pcn C is denoted by $\mathsf{Conf}_{pcn}(C)$. If C is a cn, we use $\mathsf{Conf}_{cn}(C)$ in lieu of $\mathsf{Conf}_{pcn}(C)$.

Example 5.5. Consider the pcn C_1 and the cn C_2 in Figure 6. Their configurations are identical, as illustrated below

$$\mathsf{Conf}_{pcn}(C_1) = \mathsf{Conf}_{cn}(C_2) = \{\{a\}, \{b\}, \{d\}, \{a, d\}, \{b, c\}, \{b, d\}, \{b, c, d\}\}$$

Example 5.6. Consider the cn C_2 in Figure 6b. The marking $m' = \{s_1, s_7, s_8, s_9\}$ is reachable due to the following firing sequence:

$$\mathsf{m}\ [\{b, d\}\rangle\ \{s_0, s_1, s_4, s_7, s_9\}\ [\{c\}\rangle\ m'$$

Then, it holds that ${}^\bullet m' = \{b, c, d\}$ is a configuration of C_2 (where $\mathsf{Conf}_{cn}(C_2)$ is defined in Example 5.5). Conversely, take $X = \{b, c, d\} \in \mathsf{Conf}_{cn}(C_2)$, and note that ${}^\bullet X = \{s_0, s_2, s_3, s_4, s_5\}$ and $X^\bullet = \{s_7, s_8, s_9\}$. In this case, $\mathsf{m} - {}^\bullet X + X^\bullet = \{s_1, s_4, s_7, s_9\} = m'$ is is a reachable marking of C_2.

6 CAUSAL NETS AND EVENT STRUCTURES

In this section, we establish the connection between causal nets and prime event structures.

6.1 From ppeses to pcns

We demonstrate that every ppes can be linked to a pcn with identical configurations. To begin, we introduce the mapping $\mathcal{A}$ from ppeses to pcn.

Definition 6.1. Let $\mathsf{P} = (E, <, \#)$ be a ppes. The corresponding pcn is $\mathcal{A}(\mathsf{P}) = \langle S, E, F, I, \mathsf{m}\rangle$ where

(1) $S = \{(*, e) \mid e \in E\} \cup \{(e, *) \mid e \in E\} \cup \{(\{e, e'\}, \#) \mid e \# e'\}$,
(2) $F = \{(s, e) \mid s = (*, e) \vee (s = (W, \#) \wedge e \in W)\} \cup \{(e, s) \mid s = (e, *)\}$,
(3) $I = \{(s, e) \mid s = (*, e') \wedge e' < e\}$, and
(4) $\mathsf{m} = \{(*, e) \mid e \in E\} \cup \{(\{e, e'\}, \#) \mid e \# e'\}$.

The construction associates the ppes P with a pcn that has as many transitions as there are events in P. Places are identified with pairs, taking one of the following forms: (i) $(*, e)$ for the precondition of e; (ii) $(e, *)$ for the postcondition of e; and (iii) $(\{e, e'\}, \#)$ for the conflict $e \# e'$. The flow relation is defined so that each transition e consumes tokens

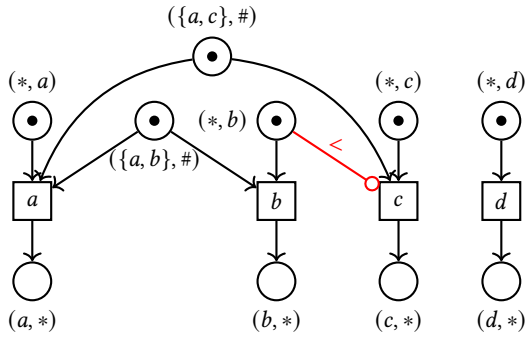


Fig. 7. $\mathcal{A}(\mathsf{P})$

from $(*, e)$ and every $(W, \#)$ where $e \in W$ (W is a subset of events containing two pairwise conflicting events); and only produces a token in $(e, *)$. In contrast to the classical construction of occurrence nets from PPESes [33], places do not convey causal dependencies. Instead, causal dependencies are modeled through inhibitor arcs: if e causally depends on e' (i.e., $e' < e$), then there is an inhibitor arc between the transition e and the place $(*, e')$, which is a place in the preset of e'. The initial marking m assigns a token to any place appearing in the preset of a transition.

Example 6.2. Let $\mathsf{P} = (E, <, \#)$ be a PPES where $E = \{a, b, c, d\}$, $< = \{(b, c)\}$ and $\# = \{(a, b), (b, a), (a, c), (c, a)\}$. The corresponding PCN is illustrated in Figure 7. Note that the transition c is not initially enabled due to the inhibitor arc from $(*, b)$; however, all other transitions are enabled. Additionally, a and b are in conflict since they both consume tokens from $(\{a, b\}, \#)$.

The adequacy of the mapping $\mathcal{A}(\cdot)$ is formally stated by demonstrating the equivalence of the respective sets of configurations.

Proposition 6.3. *Let* P *is a PPES. Then,* $\mathcal{A}(\mathsf{P})$ *is a PCN and* $\mathsf{Conf}_{PPES}(\mathsf{P}) = \mathsf{Conf}_{PCN}(\mathcal{A}(\mathsf{P}))$.

Proof. We begin by examining the conditions to ensure that $\mathcal{A}(\mathsf{P})$ is a PCN.

(1) According to the definition of F, for every event e, we have that $e^\bullet = \{(e, *)\}$. Additionally, $(e, *) \notin {}^\bullet e'$ for all $e' \in E$. Consequently, $<_{\mathcal{A}(\mathsf{P})} \cap (E \times E) = \emptyset$.
(2) By the definition of F, $E^\bullet = \{(e, *) \mid e \in E\}$. Hence, $[\![E^\bullet]\!] = E^\bullet$.
(3) By the definition of F, for every pair of transitions e and e', if a place s belongs to ${}^\bullet e \cap {}^\bullet e'$, then it has the form $(\{e, e'\}, \#)$. According to the definition of I, there are no inhibitor arcs connected to places of the form $(\{e, e'\}, \#)$. Therefore, $\forall e \neq e' \in E.\ {}^\bullet e \cap {}^\bullet e' \cap {}^\circ E = \emptyset$ holds.
(4) According to the definition of I, $\forall e \in E.\ {}^\circ e = \{(s, e) \mid s = (*, e') \ \wedge\ e' < e\}$. Since $\lfloor e \rfloor_<$ is finite, ${}^\circ e$ is so.
(5) According to the definitions of F and I, we have ${}^\bullet e \cap {}^\circ e' = \{(s, e) \mid s = (*, e') \ \wedge\ e' < e\}$. Therefore, $e' \lessdot e$ if and only if $e' < e$. Consequently, $\lessdot$ is an irreflexive partial order, given that $<$ is also such.
(6) By analogous reasoning to the previous point, $\lfloor e \rfloor_\lessdot = \lfloor e \rfloor_<$. We argue by contradiction. Suppose there exist $e', e'' \in \lfloor e \rfloor_<$ such that $e' \natural e''$ and $e' \neq e''$. From $e' \natural e''$, we deduce that ${}^\bullet e' \cap {}^\bullet e'' \neq \emptyset$. According to the definition of F, ${}^\bullet e' \cap {}^\bullet e'' = (\{e', e''\}, \#)$ and $e' \# e''$. Since $\lfloor e \rfloor_\lessdot = \lfloor e \rfloor_<$, it follows that $e' < e$ and $e'' < e$. Therefore, two causes of e are in conflict in P, contradicting the hypothesis that P is a PPES.
(7) $\mathsf{m} = {}^\bullet E$ and ${}^\circ E \subseteq \mathsf{m}$ both hold by construction.

It remains to show that configurations of P and $\mathcal{A}(\mathsf{P})$ coincide.

- $\subseteq$) Consider a configuration X of P, i.e., $X \in \mathsf{Conf}_{p\text{PES}}(\mathsf{P})$. Assume a total ordering $\{e_1, \ldots, e_n\}$ of the events in X that is compatible with the causality relation ($<$). Let X_i represent the set of events $\{e_1, \ldots, e_i\}$ for $i \in \{1, \ldots, n\}$. Through a straightforward induction on i, it can be demonstrated that the transition e_i is enabled at $\mathsf{m} - {}^\bullet X_{i-1} + X_{i-1}{}^\bullet$. Consequently, $\mathsf{m} - {}^\bullet X + X^\bullet$ is a reachable marking, and, as a result, $X \in \mathsf{Conf}_{p\text{CN}}(\mathcal{A}(\mathsf{P}))$.
- $\supseteq$) Consider a configuration X in $\mathsf{Conf}_{p\text{CN}}(\mathcal{A}(\mathsf{P}))$. For all $e, e' \in X$, it holds that ${}^\bullet e \cap {}^\bullet e' = \emptyset$, implying $\neg(e \natural e')$. Consequently, $\neg(e \# e')$. Additionally, $\lfloor e \rfloor_{\lessdot} = \lfloor e \rfloor_{<}$. Hence, X is downward closed with respect to $<$. Therefore, $X \in \mathsf{Conf}_{p\text{PES}}(\mathsf{P})$. □

Corollary 6.4. *Let* P *be a* PES*. Then* $\mathcal{A}(\mathsf{P})$ *is a* CN *and* $\mathsf{Conf}_{PES}(\mathsf{P}) = \mathsf{Conf}_{CN}(\mathcal{A}(\mathsf{P}))$.

6.2 From pCNs to pPESes

Now, we introduce the mapping $\mathcal{Q}$ from pCNs to pPESes. Its definition relies on the facts that $\lessdot$ is an irreflexive partial order, and conflicts are locally preserved in a *p*CN.

Definition 6.5. Let $C = \langle S, T, F, I, \mathsf{m}\rangle$ be a pCN. The associated pPES is $\mathcal{Q}(C) = (T, \lessdot, \#)$ where

$$\# = \{(t, t') \mid t \neq t' \in T \ \wedge \ t \natural t'\}.$$

Example 6.6. Consider the pCN C_1 of Example 5.2. Note that, $b \lessdot c$ holds because ${}^\bullet b \cap {}^\circ c \neq \emptyset$. Additionally, a and b are in conflict (i.e., both $a\#b$ and $b\#a$ holds) because ${}^\bullet a \cap {}^\bullet b \neq \emptyset$. Hence,

$$\mathcal{Q}(C_1) = (\{a, b, c, d\}, \{(b, c)\}, \{(a, b), (b, a)\})$$

The PES associated with the CN C_2 of Example 5.2 is

$$\mathcal{Q}(C_2) = (\{a, b, c, d\}, \{(b, c)\}, \{(a, b), (b, a), (a, c), (b, c))\}$$

which is actually a PES because conflicts are hereditary.

Proposition 6.7. *Let* C *be a* pCN*. Then,* $\mathcal{Q}(C)$ *is a* pPES *and* $\mathsf{Conf}_{pCN}(C) = \mathsf{Conf}_{pPES}(\mathcal{Q}(C))$.

Proof. Consider the pCN$C = \langle S, T, F, I, \mathsf{m}\rangle$ and its encoding $\mathcal{Q}(C) = (T, \lessdot, \#)$. We first show that $\mathcal{Q}(C)$ is a pPES. Note that $\#$ is irreflexive by definition. It is also symmetric because $\natural$ is so. Since C is a pCN, $\lessdot$ is an irreflexive partial order. Moreovoer, $\{e' \in E \mid e' \lessdot e\}$ is finite and conflict free because $\lfloor e \rfloor_{\lessdot}$ is finite and conflict free. Since C is a pCN, $e \lessdot e'$ implies $\neg(e \natural e')$. Therefore, we conclude that $e < e'$ implies $e \# e'$. Hence, $\mathcal{Q}(C)$ is a pPES.

We now show that $\mathsf{Conf}_{p\text{PES}}(\mathcal{Q}(C)) = \mathsf{Conf}_{p\text{CN}}(C)$. The proof is analogous to the one in Proposition 6.3 but we spell it in some details.

- $\subseteq$) Consider a reachable marking m' of C. It is easy to see that ${}^\bullet m'$ is a configuration of $\mathcal{Q}(C)$, as the firing sequence leading to m' guaranties conflict freeness of ${}^\bullet m'$ and also that for each $e \in {}^\bullet m'$ it holds that $\lfloor e \rfloor_{\lessdot}$ is a subset of ${}^\bullet m'$.
- $\supseteq$) Consider a configuration X of $\mathcal{Q}(C)$. Assume a total ordering $\{e_1, \ldots, e_n\}$ of the events in X that is compatible with the causality relation ($\lessdot$). Let X_i represent the set of events $\{e_1, \ldots, e_i\}$ for $i \in \{1, \ldots, n\}$. Through a straightforward induction on i, it can be demonstrated that the transition e_i is enabled at X_i. Consequently, X is a reachable marking, and, as a result, $X \in \mathsf{Conf}_{p\text{PES}}(\mathcal{Q}(C))$.

Corollary 6.8. *Let C be a CN. Then, $\mathcal{Q}(C)$ is a PES and* $\mathsf{Conf}_{PES}(\mathcal{Q}(C)) = \mathsf{Conf}_{CN}(C)$.

Proof. If C is a CN, then conflicts are also inherited along $\lessdot$. Hence, $\mathcal{Q}(C)$ is a PES. □

The following result ensures that the notion of causal nets is adequate for PESes. It is obtained by combining Proposition 6.3 and Corollary 6.4 (in Section 6.1) with Proposition 6.7 and Corollary 6.8 above.

Theorem 6.9. *If C is a causal net, then $C \equiv \mathcal{A}(\mathcal{Q}(C))$. If* P *is a PES, then* $\mathsf{P} \equiv \mathcal{Q}(\mathcal{A}(\mathsf{P}))$.

7 REVERSIBLE CNS AND REVERSIBLE PESES

In this section we introduce a reversible version of PCNs and show that they are an operational counterpart for RPESes.

7.1 Reversible causal nets

The intuition behind the definition of a reversible causal net is that of extending a (pre) causal net with transitions referred to as *backward*. These backward transitions are designated to reverse or undo the effects of previously executed ordinary transitions, which are called *forward*. Consider an IPT $N = \langle S, T, F, I, \mathsf{m}\rangle$, and let $t \in T$ be a transition. We use S_t to denote the set of places $\{s \in S \mid s \in {}^\bullet t \ \wedge\ s^\bullet = \{t\}\}$. In other words, S_t represents the places in the preset of t that do not appear in the preset of any other transitions.

Definition 7.1. An IPT $V = \langle S, T, F, I, \mathsf{m}\rangle$ is a *reversible causal net* (RCN) if there exists a partition $\{\overline{T}, \underline{T}\}$ of T satisfying the following conditions (here, $\overline{T}$ represents the set of forward transitions, and $\underline{T}$ comprises the backward transitions):

(1) $\langle S, \overline{T}, F_{|\overline{T}\times\overline{T}}, I_{|\overline{T}\times\overline{T}}, \mathsf{m}\rangle$ is a PCN net;
(2) $[[{}^\bullet\underline{T}]] = {}^\bullet\underline{T}$ and $\forall \underline{t} \in \underline{T}.\ \exists!\ t \in \overline{T}$ such that $t^\bullet = {}^\bullet\underline{t}$, ${}^\bullet t = \underline{t}^\bullet$, and $S_t \cap {}^\circ\underline{t} \neq \emptyset$;
(3) $\forall \underline{t} \in \underline{T}.\ K = \{t \mid t \in \overline{T} \wedge {}^\circ\underline{t} \cap {}^\bullet\overline{T} \neq \emptyset\}$ is finite and $[[{}^\bullet K]] = {}^\bullet K$;
(4) $\forall \underline{t} \in \underline{T}. \forall t \in \overline{T}$. if ${}^\bullet t \cap {}^\circ\underline{t} \neq \emptyset$ then $t^\bullet \cap {}^\circ\underline{t} = \emptyset$;
(5) $\lll \subseteq \overline{T} \times \overline{T}$ is a transitive relation defined such that $t \lll t'$ if $t \lessdot t'$ and if there exists $\underline{t} \in \underline{T}$ such that ${}^\bullet\underline{t} = t^\bullet$ then ${}^\circ\underline{t} \cap t'^\bullet \neq \emptyset$; and
(6) $\forall t, t', t'' \in \overline{T}.\ {}^\bullet t \cap {}^\bullet t' \neq \emptyset \ \wedge\ t' \lll t'' \ \Rightarrow\ {}^\bullet t \cap {}^\bullet t'' \neq \emptyset$.

We write $V^{\underline{T}}$ for an RCN V with backward transitions $\underline{T}$.

The first condition states that the subnet consisting of just forward transitions is a pre-causal net. The second condition establishes that each backward transition $\underline{t}$ precisely reverses one forward transition t. As a result, $\underline{t}$ consumes the tokens produced by t (i.e., $t^\bullet = {}^\bullet\underline{t}$) and produces the tokens consumed by t (i.e., $\underline{t}^\bullet = {}^\bullet t$). The condition $S_t \cap {}^\circ\underline{t} \neq \emptyset$ ensures that $\underline{t}$ causally depends on t. Requiring $[[{}^\bullet\underline{T}]] = {}^\bullet\underline{T}$ ensures that a forward transition has at most one reversing transition. The remaining conditions mirror those imposed on PESes, where inhibitor arcs model the reverse causality relation ($\prec$) and the prevention relation ($\triangleright$). If an inhibitor arc connects a backward transition to a place in the preset of some forward transition, then the modeled relation is reverse causality. Conversely, prevention is represented by linking a backward transition to a place in the postset of a forward transition. Consequently, the third condition requires each backward transition to causally depend on a finite number of forward transitions (i.e., those in K), which should also be conflict-free (i.e., not sharing places in their preset). The fourth condition stipulates that a backward transition $\underline{t}$ causally dependent on a forward transition t (i.e., ${}^\bullet t \cap {}^\circ\underline{t} \neq \emptyset$) cannot be prevented by the same transition (i.e., $t^\bullet \cap {}^\circ\underline{t} = \emptyset$). The relation $\lll$ defined by the fifth condition is analogous to sustained causation, coinciding with causality only when prevention enforces causal reversibility. The last condition asserts that conflicts are inherited along $\lll$.

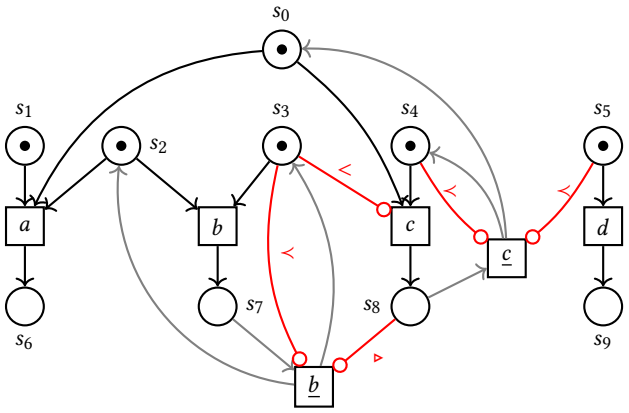


Fig. 8. A simple RCN

To illustrate better which inhibitor arcs are used to model the various relations (causality, reverse causality and prevention) we decorate them with the relation symbol.

Example 7.2. Consider the IPT V in Figure 8, which is an *r*CN with reversing transitions $\underline{T} = \{\underline{b}, \underline{c}\}$. Firstly, observe that $\langle S, \overline{T}, F_{|\overline{T}\times\overline{T}}, I_{|\overline{T}\times\overline{T}}, \mathsf{m}\rangle$ corresponds to the causal net C_2 illustrated in Figure 6b. There are two backward transitions: $\underline{b}$, for reversing b, and $\underline{c}$ for reversing c. Each of them reverses the markings of its associated forward transition, e.g., $\underline{b}$ consumes from s_7 and produces on s_2 and s_3. The inhibitor arcs linking s_3 to $\underline{b}$ and s_4 to $\underline{c}$ encode reverse causality, e.g., establishing a causal dependency of $\underline{b}$ on b. In this case, the behaviour of the net would remain unaffected if these inhibitor arcs were omitted. However, their inclusion is mandated for a consistent representation of causality through $\prec$, as opposed to relying on the flow relation. Moreover, the inhibitor arc connecting s_5 to $\underline{c}$ encodes also reverse causality by stipulating that $\underline{c}$ cannot be fired until d is fired and the token in s_5 is consumed. Finally, the inhibitor arc linking s_8 to $\underline{b}$ models prevention. Specifically, $\underline{b}$ is not enabled if s_8 is marked, signifying that c has been fired and preventing the execution of $\underline{b}$.

7.2 From RPESes to RCNes

This section delves into the relation between RPESes and RCNes. We establish the adequacy of the identified connection in Section 7.4.

Definition 7.3. Let $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ be an RPES and $\mathcal{A}(E, <, \#) = \langle S, T', F', I', \mathsf{m}\rangle$ the PCN associated with the forward flow. Then, the corresponding RCN is $\mathcal{A}_r(\mathsf{P}) = \langle S, T, F, I, \mathsf{m}\rangle$ where

(1) $T = T' \cup \{(u, \mathsf{r}) \mid u \in U\}$;
(2) $F = F' \cup \{(s, (u, \mathsf{r})) \mid s \in u^\bullet\} \cup \{((u, \mathsf{r}), s) \mid s \in {}^\bullet u\}$;
(3) $I = I' \cup \{((*, e), (u, \mathsf{r})) \mid e \prec \underline{u}\} \cup \{((e, *), (u, \mathsf{r})) \mid e \triangleright \underline{u}\}$.

The construction employs the mapping $\mathcal{A}(\_)$ to derive a PCN from the underlying PPES that consists solely of forward events. This resulting structure is then extended with an equal number of backward transitions as there are reversible events in P. A backward transition (u, r) reversing u, is defined with its preset being the postset of u and its poset being the preset of u. Inhibitor arcs are implemented as anticipated: the reverse causality relation $e \prec \underline{u}$ translates into an

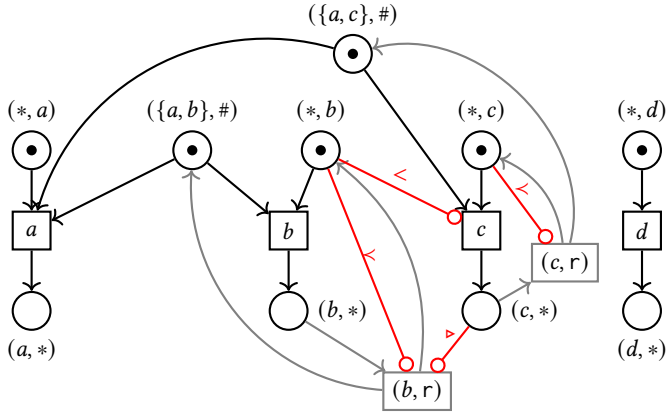


Fig. 9. $\mathcal{A}_r(\mathsf{P}_1)$

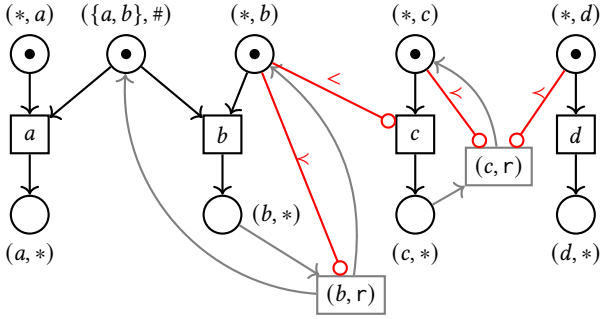


Fig. 10. $\mathcal{A}_r(\mathsf{P}_4)$

arc connecting the backward transition (u,r) to the enabling condition of e, i.e., $(*,e)$; the prevention relation $e \triangleright \underline{u}$ is represented by an arc connecting (u,r) to the post condition of e, i.e., $(e,*)$.

Example 7.4. Consider the RPES $\mathsf{P}_1 = (E, U, <, \#, \prec, \triangleright)$ presented in Example 3.12. Notably, the structure $(E, <, \#)$ and its associated CN were discussed in Example 6.2. Then, the RCN $\mathcal{A}_r(\mathsf{P}_1)$ illustrated in Figure 8 is obtained by extending the CN in Figure 7 with the transitions (b,r) and (c,r) corresponding to the reversing events $\underline{b}$ and $\underline{c}$. The inhibitor arcs from $(b,*)$ to (b,r) and $(c,*)$ to (c,r) indicate that the backward transitions are only enabled after the respective forward transitions have been executed. Furthermore, the inhibitor arc from $(*,c)$ to (b,r) indicates that the firing of b is prohibited if c has already been executed.

The RCN $\mathcal{A}(\mathsf{P}_4)$ corresponding to the RPES P_4 in Example 3.21 is depicted in Figure 10. Note the absence of the place $(\{a,c\},\#)$, reflecting that a and c do not conflict in P_4. Furthermore, there is not an inhibitor arc from $(*,c)$ to (b,r), indicating that b can be reversed even when $(*,c)$ is marked. Moreover, the inhibitor arc from $(d,*)$ to (c,r) serves to prevent the reversal of c until d has been fired.

The encoding of the RPES P_2 in Example 3.13 results in a net that is isomorphic to the one shown in Figure 8, with only the names of places and transitions being different.

Proposition 7.5. *If* P *is an* rpes*, then* $\mathcal{A}_r(\mathsf{P})$ *is an* rcn*.*

Proof. Assume $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$. We show that $\mathcal{A}_r(\mathsf{P}) = \langle S, T, F, I, \mathsf{m}\rangle$ with $\underline{T} = \{(u, \mathsf{r}) \mid u \in U\}$ satisfies the conditions in Definition 7.1.

(1) $\mathcal{A}(E, <, \#) = \langle S, T', F', I', \mathsf{m}\rangle$ is a pcn by Proposition 6.3. Moreover, $T' = E$ by Definition 6.1. Hence, $\overline{T} = E$.
(2) It is immediate that, for any (u, r) there exists a unique $u \in T$ such that $u^\bullet = {}^\bullet(u, \mathsf{r})$ and ${}^\bullet u = (u, \mathsf{r})^\bullet$ based on the definition of F. Moreover, the condition $S_u \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$ is satisfied by the definition of F and I.
(3) By construction, $\{e \mid e \in E \ \wedge \ {}^\circ(u, \mathsf{r}) \cap {}^\bullet e \neq \emptyset\} = \{e \mid ((*, e), (u, \mathsf{r})) \in I \wedge \ e \prec \underline{u}\}$ is finite. This is due to the fact that P is an rpes, and thus $\{e \in E \mid e \prec \underline{u}\}$ is finite.
(4) If ${}^\bullet e \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$ then $((e, *), (u, \mathsf{r})) \in I$. Hence, $e \prec \underline{u}$ by the definition of $\mathcal{A}_r(\_)$. Since P is an rpes, it follows that $\neg(e \triangleright \underline{u})$. Therefore, $((*, e), (u, \mathsf{r})) \notin I$ and $e^\bullet \cap {}^\circ(u, \mathsf{r}) = \emptyset$ by the definition of $\mathcal{A}_r(\_)$.
(5) Take $\lll = \ll$, i.e., $e \lll e'$ iff $e \ll e'$. Given that P is an rpes, $e \ll e'$ holds if $e < e'$ and if $e \in U$, then $e' \triangleright \underline{e}$. Therefore, $e < e'$ implies ${}^\bullet e \cap {}^\circ e' \neq \emptyset$, and hence $e \lessdot e'$ by Definition 6.1. Moreover, $e \in U$ and $e' \triangleright \underline{e}$, implies $(e, \mathsf{r}) \in \underline{T}$ and $((e', *), (e, \mathsf{r})) \in I$ (by Definition 7.3). Hence, ${}^\circ(e, \mathsf{r}) \cap e'^\bullet \neq \emptyset$.
(6) If ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$, then $e \# e'$ by Definition 6.1. If $e' \lll e''$, then $e' \ll e''$ because $\lll$ and $\ll$ coincide. Since P is an rpes, $e \# e' \ll e''$ implies $e \# e''$. Hence, ${}^\bullet e \cap {}^\bullet e'' \neq \emptyset$ by Definition 6.1. □

7.3 From rcnes to rpeses

We now address the transformation of rcns into rpeses, presenting the encoding function as $\mathcal{Q}_r(\_)$.

Definition 7.6. Let $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ be an rcn with backward transitions $\underline{T}$. Then, the associated rpes is $\mathcal{Q}_r(V^{\underline{T}}) = (E, U, <, \#, \prec, \triangleright)$ where:

(1) $E = T \setminus \underline{T}$;
(2) $U = \{t \mid \underline{t} \in \underline{T} \wedge t \in T \wedge {}^\bullet t = \underline{t}^\bullet\}$;
(3) $< \ = \lessdot_{|E \times E}$;
(4) $\# = \{(e, e') \mid e \neq e' \in E \ \wedge e \natural e'\}$;
(5) $\prec = \{(t, \underline{t}') \mid t \in E \wedge \underline{t}' \in \underline{T} \wedge {}^\bullet t \cap {}^\circ \underline{t}' \neq \emptyset\}$;
(6) $\triangleright = \{(t, \underline{t}') \mid t \in E \wedge \underline{t}' \in \underline{T} \wedge t^\bullet \cap {}^\circ \underline{t}' \neq \emptyset\}$.

The construction establishes a mapping from rcn $V^{\underline{T}}$ to an rpes where events correspond to the forward transitions $T \setminus \underline{T}$ of $V^{\underline{T}}$ (Condition (1)). By Condition (2), only forward transitions with corresponding reversing transitions in $\underline{T}$ are designated as undoable events (belonging to U). Conditions (3) and (4) specify that causality ($<$) and conflict ($\#$) relations are obtained as proper restrictions of those induced by the structure of $V^{\underline{T}}$. Conditions (5) and (6) describe the reconstruction of reverse causation ($\prec$) and prevention ($\triangleright$) from inhibitor arcs.

Example 7.7. Consider the rcn depicted in Figure 8. The events of the associated rpes are $\{a, b, c, d\}$, with $\{b, c\}$ identified as reversible events. The sole causal dependency is $b \lessdot c$. Conflicts arise from shared places in the presets of transitions, specifically s_0 and s_2, resulting in $a \# c$ and $b \# d$. The inhibitor arc connecting s_3 to $\underline{b}$ is translated into the reverse causality $b \prec \underline{b}$ because s_3 belongs to the preset of b. Similarly, the arcs from s_4 to $\underline{c}$ and from s_5 to $\underline{c}$ are mapped to $c \prec \underline{c}$ and $d \prec \underline{c}$, respectively. In contrast, the arc from s_8 to $\underline{b}$ results in $c \triangleright \underline{b}$ since s_8 is part of the postset of c. Consequently, the derived rpes corresponds to the one defined in Example 3.13.

Proposition 7.8. *If* $V^{\underline{T}}$ *is an* rcn*, then* $\mathcal{Q}_r(V^{\underline{T}})$ *is an* rpes*.*

Proof. Consider $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ with backward transitions $\underline{T}$, and let $\mathcal{Q}_r(V^{\underline{T}}) = (\overline{T}, U, <, \#, \prec, \triangleright)$ be defined as in Definition 7.6. According to Definition 7.1(1), $\langle S, \overline{T}, F_{|\overline{T}\times\overline{T}}, I_{|\overline{T}\times\overline{T}}, \mathsf{m}\rangle$ is a CN. By Proposition 6.7, $(\overline{T}, \lessdot_{|\overline{T}\times\overline{T}}, \#)$ is a PPES. It remains to show the conditions in Definition 3.11.

(1) According to the definition of $\mathcal{Q}_r(\_)$, $\prec = \{(t, \underline{t}) \mid t \in \overline{T} \wedge \underline{t} \in \underline{T} \wedge {}^\bullet t \cap {}^\circ\underline{t} \neq \emptyset\}$. From Definition 7.1(2), it is established that $S_t \cap {}^\circ\underline{t} \neq \emptyset$. Hence, $t \prec \underline{t}$ holds for all $\underline{t} \in \underline{T}$. Analogously, by using Definition 7.1(3), it can be deduced that $\{t \in \overline{T} \mid t \prec \underline{u}\}$ is finite and conflict-free for every $u \in U$.

(2) Immediate.

(3) If $t \prec \underline{u}$, then ${}^\bullet t \cap {}^\circ\underline{u} \neq \emptyset$ according to the definition of $\mathcal{Q}_r(\_)$. Based on Definition 7.1(4), it is established that $t^\bullet \cap {}^\circ\underline{u} = \emptyset$. Hence, $\neg(t \triangleright \underline{u})$ holds by the definition of $\mathcal{Q}_r(\_)$.

(4) It follows by setting $\lll = \ll$ and reasoning as it is done in the proof of Proposition 7.5.

(5) It follows from Definition 7.1(6).

□

7.4 Correspondence between RCNes and RPESes

The mappings introduced in the preceding sections establish a close correspondence between RPESes and RCNs concerning configurations, as will be evident in what follows.

Definition 7.9. Let $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ be an RCN. A *pre-configuration* X of $V^{\underline{T}}$ is a conflict-free set of forward transitions, i.e., $X \subseteq T \setminus \underline{T}$ and $\forall t, t' \in X.\ {}^\bullet t \cap {}^\bullet t' \neq \emptyset \implies t = t'$. The *associated marking* is $m_X = \mathsf{m} - {}^\bullet X + X^\bullet$.

If X is a pre-configuration of $V^{\underline{T}}$, it is noteworthy that m_X may not necessarily be a reachable marking. For example, consider the pre-configuration $\{c, d\}$ of the RCN in Figure 8. The associated reachable marking $m_X = \{s_1, s_2, s_3, s_8, s_9\}$ is not reachable itself, as firing c is inhibited when s_3 is marked.

Definition 7.10. A pre-configuration X of an RCN $V^{\underline{T}}$ is a *configuration* if m_X is a reachable marking of $V^{\underline{T}}$. The set of all configurations of $V^{\underline{T}}$ is denoted by $\mathsf{Conf}_{rcn}(V^{\underline{T}})$.

Proposition 7.11. *Let $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ be an RCN, and $m \in \mathcal{M}_{V^{\underline{T}}}$ a reachable marking. Then, $X = [\![{}^\bullet m]\!] \cap (T \setminus \underline{T})$ is a configuration.*

Proof. Trivial. □

The subsequent auxiliary results are pivotal for establishing the proofs of the main results in this section.

Proposition 7.12. *Given an RPES $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$, every conflict-free set of events $X \subseteq E$ is a pre-configuration of $\mathcal{A}_r(\mathsf{P})$.*

Proof. It follows straightforwardly by observing that the conflict-free condition for X implies that in the corresponding RPES $\mathcal{A}_r(\mathsf{P})$, for any pair of events e and e' in X, ${}^\bullet e \cap {}^\bullet e' = \emptyset$ holds. □

Lemma 7.13. *Let $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ be an RPES and $X \subseteq E$ a conflict-free set of events. If $A \cup \underline{B}$ is enabled at X, then $m_X\ [A \cup \{(u, \mathsf{r}) \mid u \in B\}\rangle$ in $\mathcal{A}_r(\mathsf{P})$.*

Proof. Assuming that $A \cup \underline{B}$ is enabled at X, we demonstrate that $m_X\ [A \cup \{(u, \mathsf{r}) \mid u \in B\}\rangle$. Since X is conflict-free, it is a pre-configuration, and m_X is the associated marking. The enabling of $A \cup \underline{B}$ at X implies:

(1) $A \cap X = \emptyset$, $B \subseteq X$ and $\mathsf{CF}(X \cup A)$,

(2) $\forall e \in A, e' \in E$. if $e' < e$ then $e' \in X \setminus B$,
(3) $\forall e \in B, e' \in E$. if $e' \prec \underline{e}$ then $e' \in X \setminus (B \setminus \{e\})$,
(4) $\forall e \in B, e' \in E$. if $e' \triangleright \underline{e}$ then $e' \notin X \cup A$.

We examine the different conditions:

(1) The condition $A \cap X = \emptyset$ implies that ${}^\bullet A \subseteq m_X$, while $B \subseteq X$ means that $B^\bullet \subseteq m_X$. Consequently, $B \times \{\mathsf{r}\} \subseteq m_X$. Finally, the conflict-freeness condition $\mathsf{CF}(X \cup A)$ is trivially verified as ${}^\bullet A \subseteq m_X$.
(2) For every $e \in A$ and $e' \in E$, if $e' < e$, then $e' \in X \setminus B$. Now, $e' < e$ implies that $(*, e') \in {}^\circ e$, and since $e' \in X \setminus B$, we have $m_X(*, e') = 0$.
(3) For every $e \in B$ and $e' \in E$, if $e' \prec \underline{e}$, then $e' \in X \setminus (B \setminus \{e\})$. Once more, as $e' \prec \underline{e}$, the inhibitor set ${}^\circ(e, \mathsf{r})$ includes $(e', *)$. Since $e' \in X \setminus (B \setminus \{e\})$, it follows that $m_X(e', *) = 0$.
(4) For every $e \in B$ and $e' \in E$, if $e' \triangleright \underline{e}$, then $e' \notin X \cup A$. Once again, $e' \triangleright \underline{e}$ implies that $(e', *) \in {}^\circ(e, \mathsf{r})$, and since $e' \notin X \cup A$, we have $m_X(e', *) = 1$, indicating that e' must not be in X, as $m_X(e', *) = 1$ shows.

Combining these arguments, it follows that $m_X\,[A \cup \{(u, \mathsf{r}) \mid u \in B\}\rangle$: in fact the first condition implies that ${}^\bullet A + {}^\bullet(B \times \{\mathsf{r}\}) \subseteq m_X$, whereas the other three imply that all inhibiting places are empty. □

Lemma 7.14. *Let $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ be an rpes, $X \subseteq E$ a conflict-free set of events, and $Y \subseteq E \cup (U \times \{\mathsf{r}\})$ a set of transitions of $\mathcal{A}_r(\mathsf{P})$. If $m_X\,[Y\rangle$ then $(Y \cap E) \cup \{\underline{u} \mid (u, \mathsf{r}) \in Y\}$ is enabled at X.*

Proof. First, we observe that X is a pre-configuration and, according to Proposition 7.12, $m_X = \mathsf{m} - {}^\bullet X + X^\bullet$ is its associated marking. Assume that $m_X\,[Y\rangle$. We will demonstrate that $(Y \cap E) \cup (Y \cap U \times \{\mathsf{r}\})$ is enabled at X. We need to show that, denoting $A = (Y \cap E)$ and $B = (Y \cap U)$, the conditions of Definition 3.14 hold:

(1) For $A \cap X = \emptyset$, $B \subseteq X$ and $\mathsf{CF}(X \cup A)$, we proceed as follows.
- Observe that $m_X\,[Y\rangle$ implies $m_X\,[A\rangle$. Then, ${}^\bullet A \subseteq m_X$, which implies $A \cap X = \emptyset$.
- From $B = (Y \cap U)$ and $m_X\,[Y\rangle$, we conclude ${}^\bullet(Y \cap U \times \{\mathsf{r}\}) \subseteq m_X$. Consequently, $B = Y \cap U \subseteq X$,
- $\mathsf{CF}(X \cup A)$ holds trivially because $m_X\,[A\rangle$ and conflicts cannot be introduced in the net.

(2) Condition $\forall e \in A, e' \in E$. if $e' < e$ then $e' \in X \setminus B$ is obtained as follows.
We have that $((*, e'), e) \in I$ and $m_X\,[Y\rangle$, which means $e' \in X$. Additionally, $e' \notin B$ because if $e' \in B$, it would imply that $m_X((e',)) = 1$ and $Y^\bullet((e',)) = 1$, contradicting the fact that $m_X\,[Y\rangle$.

(3) Condition $\forall e \in B, e' \in E$. if $e' \prec \underline{e}$ then $e' \in X \setminus (B \setminus \{e\})$ is shown below:
$e' \prec \underline{e}$ implies that $(*, e') \in {}^\circ(e, \mathsf{r})$. As $m_X\,[Y\rangle$ we have that (e, r) is in Y and then $m_X((*, e')) = 0$, implying that $e' \in X$. Furthermore, $e' \notin B$ because otherwise $m_X\,[Y\rangle$ would not hold. Thus, we can conclude that $e' \in X \setminus (B \setminus \{e\})$.

(4) Condition $\forall e \in B, e' \in E$. if $e' \triangleright \underline{e}$ then $e' \notin X \cup A$ is shown below.
The condition $e' \triangleright \underline{e}$ implies that $(e', *) \in {}^\circ(e, \mathsf{r})$. Given that $m_X\,[Y\rangle$ and $(e', *)$ is not marked, it follows that $e' \notin X \cap A$.

As all the conditions of Definition 3.14 are satisfied, we conclude that $(Y \cap E) \cup (Y \cap U \times \{\mathsf{r}\})$ is enabled at X. □

Lemma 7.15. *Let $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ be an rcn, $X \subseteq T \setminus \underline{T}$ a pre-configuration of $V^{\underline{T}}$, $A \subseteq T \setminus \underline{T}$ a set of forward transitions, $B \subseteq \underline{T}$ a set of backward transitions, and $\underline{B} = \{\underline{u} \mid u \in T \wedge \underline{t} \in \underline{T} \wedge u^\bullet = {}^\bullet\underline{T}\}$. If $A \cup \underline{B}$ is enabled at X in $\mathcal{Q}_r(V^{\underline{T}})$ then $m_X\,[A \cup B\rangle$.*

Proof. Along the same line of Lemma 7.13

Lemma 7.16. *Let $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$, X a pre-configuration of $V^{\underline{T}}$, $A \subseteq T \setminus \underline{T}$ a set of forward transitions, $B \subseteq \underline{T}$ a set of backward transitions, and $\underline{B} = \{\underline{u} \mid u \in T \wedge \underline{t} \in \underline{T} \wedge u^\bullet = {}^\bullet\underline{T}\}$. If $m_X\ [A \cup B\rangle$ then $A \cup \underline{B}$ is enabled at X in $\mathcal{Q}_r(V^{\underline{T}})$.*

Proof. Following a similar line of reasoning as in Lemma 7.14. □

As a consequence of the above result, we have the operational correspondence for $\mathcal{A}_r(\_)$ and $\mathcal{Q}_r(\_)$ as stated below.

Theorem 7.17. *Let P be an rpes. Then, $X \in \mathsf{Conf}_{rPES}(\mathsf{P})$ iff $X \in \mathsf{Conf}_{rCN}(\mathcal{A}_r(\mathsf{P}))$.*

Proof. Follows from Lemma 7.13 and Lemma 7.14 □

Theorem 7.18. *Let $V^{\underline{T}}$ be an rcn. $X \in \mathsf{Conf}_{rCN}(V^{\underline{T}})$ iff $X \in \mathsf{Conf}_{rPES}(\mathcal{Q}_r(V^{\underline{T}}))$.*

Proof. Follows from Lemma 7.15 and Lemma 7.16 □

Combining the previous results, we conclude that rcns are a proper counterpart of rpeses.

Theorem 7.19. *If $V^{\underline{T}}$ is an rcn then $V^{\underline{T}} \equiv \mathcal{A}_r(\mathcal{Q}_r(V^{\underline{T}}))$. If P is an rpes then $\mathsf{P} \equiv \mathcal{Q}_r(\mathcal{A}_r(\mathsf{P}))$.*

7.5 Reversing disciplines

Similarly to the characterisation of rpeses, it is possible to structurally characterise various reversing disciplines in rcns. In the following, given an rcn $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ and a reversing transition $\underline{t} \in \underline{T}$, we denote $t \in \overline{T}$ the corresponding forward transition. Conversely, for any $t \in \overline{T}$ we denote $\underline{t} \in \underline{T}$ the corresponding reversing one, if it exists.

Definition 7.20. Let $V^{\underline{T}} = \langle S, T, F, I, \mathsf{m}\rangle$ an rcn. Then, the reversing disciplines are defined as follows.

- *cause-respecting*: for all $t, t' \in \overline{T}$, if ${}^\bullet t \cap {}^\circ t' \neq \emptyset$ and there exists $\underline{t} \in \underline{T}$ such that ${}^\bullet\underline{t} = t^\bullet$ and $\underline{t}^\bullet = {}^\bullet t$ then ${}^\circ\underline{t} \cap t'^\bullet \neq \emptyset$;
- *causal*: for all $t \in \overline{T}$ and $\underline{t'} \in \underline{T}$, if ${}^\bullet t \cap {}^\circ\underline{t'} \neq \emptyset$ then ${}^\bullet\underline{t'} = t^\bullet$ and $\underline{t'}^\bullet = {}^\bullet t$, and $t^\bullet \cap {}^\circ\underline{t'} \neq \emptyset$ iff ${}^\bullet t \cap {}^\circ t' \neq \emptyset$;
- *inverse cause-respecting*: for all $t \in \overline{T}$ and $\underline{t'} \in \underline{T}$, if ${}^\bullet t \cap {}^\circ t' \neq \emptyset$ then $t^\bullet \cap {}^\circ\underline{t'} \neq \emptyset$; and
- *inverse causal*: for all $t \in \overline{T}$ and $\underline{t'} \in \underline{T}$, if ${}^\bullet t \cap {}^\circ\underline{t'} \neq \emptyset$ then ${}^\bullet\underline{t'} = t^\bullet$ and $\underline{t'}^\bullet = {}^\bullet t$, and $t^\bullet \cap {}^\circ\underline{t'} \neq \emptyset$ iff ${}^\bullet t' \cap {}^\circ t \neq \emptyset$;

The definition above reformulates the conditions imposed on the forward and backward relations in rpes. In a cause-respecting rcn, the forward dependencies among transitions align with the sustained causality $\lll$. In a causal rcn, forward transitions prevent the reversing of their causes.

Similarly, in an inverse cause-respecting rcn, if one transition causes another, it also prevents the reversal of the second. In contrast, in an inverse causal rcn, the reversal of a transition is prevented by all its causes, and the reversing causality relation behaves similarly to a causal rcn.

Example 7.21. The net depicted in Figure 9 is both cause-respecting and casual. In this rcn, we have ${}^\bullet b \cap {}^\circ c \neq \emptyset$ and there exists a reversing transition (b, r). According to the requirement in Definition 7.20, we have ${}^\circ(b, \mathsf{r}) \cap c^\bullet \neq \emptyset$. This implies that the relation $\lll$ coincides with $\lessdot$.

Furthermore, it is also causal because each reversing transition (x, r) (where $x \in a, b$) satisfies ${}^\circ(x, \mathsf{r}) \cap {}^\bullet x = \emptyset$. The condition ${}^\bullet b \cap {}^\circ c \neq \emptyset$ holds if and only if ${}^\circ(b, \mathsf{r}) \cap c^\bullet \neq \emptyset$, provided that the reversing transition exists.

The net depicted in Figure 11 is both inverse cause-respecting and inverse causal. For the inverse cause-respecting case, we observe, for instance, that ${}^\bullet t_1 \cap {}^\circ t_3 \neq \emptyset$ and ${}^\bullet t_2 \cap {}^\circ t_3 \neq \emptyset$, implying that $t_1{}^\bullet \cap {}^\circ\underline{t}_3 \neq \emptyset$ and $t_2{}^\bullet \cap {}^\circ\underline{t}_3 \neq \emptyset$ must hold true. The net is also inverse causal because, for example, ${}^\bullet t_3 \cap {}^\circ t_3 \neq \emptyset$, where ${}^\circ t_3$ precisely corresponds to the place s_3.

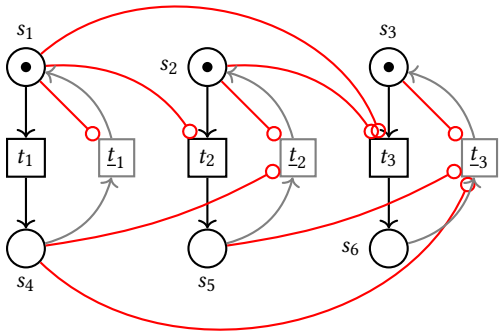


Fig. 11. An inverse causal RCN $V^{\underline{T}}$

Proposition 7.22. *Let* P *be an* RPES *and* $\mathcal{A}_r(\mathsf{P})$ *the associated* RCN. *If* P *is* cause-respecting, causal, inverse cause-respecting, *or* inverse causal, *then* $\mathcal{A}_r(\mathsf{P})$ *is also so.*

Proof. Consider $\mathsf{P} = (E, U, <, \#, \prec, \triangleright)$ and the associated RCN $\mathcal{A}_r(\mathsf{P}) = \langle S, T, F, I, \mathsf{m}\rangle$ with $\underline{T} = \{(u, \mathsf{r}) \mid u \in U\}$.

(1) Assume that P is cause-respecting, then for every $e, e' \in E$, if $e < e'$ then $e \ll e'$. First consider $e, e' \notin U$ and $e < e'$. Then (e, f) and (e', f) in $\overline{T}$ are such that ${}^\bullet(e, \mathsf{f}) \cap {}^\circ(e', \mathsf{f}) \neq \emptyset$ as $e < e'$. Suppose now that $e = u \in U$. We have again (u, f) and (e', f) in $\overline{T}$, and since $u < e'$ it holds that ${}^\bullet(u, \mathsf{f}) \cap {}^\circ(e', \mathsf{f}) \neq \emptyset$. Since $u \in U$, there exists the transition $(u, \mathsf{r}) \in \underline{T}$. Since $u \ll e'$ we have that $e' \triangleright \underline{(u, \mathsf{r})}$, but this implies that ${}^\circ(u, \mathsf{r}) \cap (e', \mathsf{f})^\bullet \neq \emptyset$.
(2) If P is causal then for every $e \in E$, $u \in U$ we have
   (a) $e \prec \underline{u}$ iff $e = u$, and
   (b) $e \triangleright \underline{u}$ iff $u < e$

   We have $(e, \mathsf{f}) \in \overline{T}$ and $(u, \mathsf{r}) \in \underline{T}$. Assume that ${}^\bullet(e, \mathsf{f}) \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$. Since P is causal by condition $e \prec \underline{u}$ iff $e = u$ we have that $e = u$. Assume now that there is another $(u', \mathsf{r}) \in \underline{T}$ such that ${}^\bullet(e, \mathsf{f}) \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$. Being P causal $u' = e = u$. Assume now $(e, \mathsf{f})^\bullet \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$, this implies that $e \triangleright \underline{u}$ and being P causal we have that $u < e$ which means ${}^\bullet(u, \mathsf{f}) \cap {}^\circ(e, \mathsf{f}) \neq \emptyset$. The same reasoning applies when ${}^\bullet(u, \mathsf{f}) \cap {}^\circ(e, \mathsf{f}) \neq \emptyset$ as we can conclude that $(e, \mathsf{f})^\bullet \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$.
(3) P is inverse cause-respecting then for every $e \in E$, $u \in U$, if $e < u$ then $e \triangleright \underline{u}$, but then if ${}^\bullet(e, \mathsf{f}) \cap {}^\circ(u, \mathsf{f}) \neq \emptyset$ $(e < u)$ and $(e, \mathsf{f})^\bullet \cap {}^\circ(u, \mathsf{r}) \neq \emptyset$.
(4) For case in which P is inverse causal we can use the same reasoning of the previous cases.

□

Proposition 7.23. *Let* $V^{\underline{T}}$ *be an* RCN *and* $\mathcal{Q}_r(V^{\underline{T}})$ *the associated* RPES. *If* $V^{\underline{T}}$ *is* cause-respecting, causal, inverse cause-respecting *or* inverse causal *then also* $\mathcal{Q}_r(V^{\underline{T}})$ *is so.*

Proof. Along the same lines of proof of Proposition 7.22. □

Example 7.24. Consider again the RCN $V^{\underline{T}}$ in Figure 11. Up to renaming of transitions $\mathcal{Q}_r(V^{\underline{T}})$ is the inverse causal RPES P'_3 which corresponds to the RPES P_3 of Example 3.20 where $(c, \underline{a})$ has been removed.

In causal RCN, a transition can be undone provided that all its consequences are undone beforehand. If the set of backward transitions coincides with the forward one, we have that each transition can be reversed. Hence, one could state that the basic property of a reversible calculus holds: the loop lemma. Furthermore, by using the framework of [14] one could prove that this kind of RCN is causally consistent, and this results can be brought also to RPES. More in general, other reversing disciplines can be studied on RCN, and the theorems in the previous subsections guarantee that these disciplines can be *transferred* from nets to event structures and vice versa.

8 CAUSAL NETS AND OCCURRENCE NETS

The developments in Sections 6.1 and 6.2 have shown that CNS serve as counterparts to prime event structures, as highlighted in Theorem 6.9. Now, we delve into the connections between CNS and occurrence nets, typically associated with prime event structures. Finally, we present additional evidence that indicates the difficulty of adapting occurrence nets to model out-of-causal order reversibility.

Let us start by introducing some key concepts about nets that will be essential throughout this section. An IPT N is deemed *single-execution* if its states are sets, meaning that for all $X \in \mathrm{St}(N)$, $X = [[X]]$. In simpler terms, each transition in a single-execution net can be fired at most once in a firing sequence. Single-execution nets (albeit without inhibitor arcs) are also known as *1-occurrence* nets [30, 32]. The net N_1 in Figure 4a is single-execution.

An IPT is said *conflict-saturated* if any pair of transitions, not present in any firing sequence of the net, share a common place in their presets.

Definition 8.1. Let $N = \langle S, T, F, I, \mathsf{m}\rangle$ be an IPT. Two transitions $t, t' \in T$ are deemed to be in *conflict* if they do not simultaneously appear in any state of N, expressed as $\{t, t'\} \not\subseteq [[X]]$ for all $X \in \mathrm{St}(N)$. The IPT N is said *conflict-saturated* when, for all $t, t' \in T$, if t and t' are in conflict, then ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$.

A conflict-saturated net is one in which conflicts are characterised structurally.

Example 8.2. The IPT shown in Figure 12 is conflict-saturated since the absence of pairs a and c, as well as b and c, in any configuration of the net implies that each pair of transitions shares a place in their preset (s_7 for a and c, and s_3 for b and c).

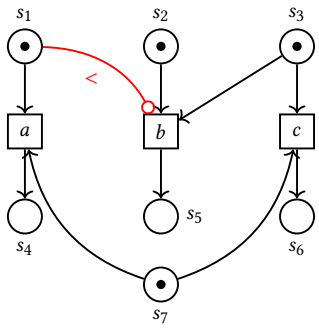


Fig. 12. A conflict-saturated net

Every single-execution IPT can be transformed into a conflict-saturated, equivalent one, i.e., without changing the executions of the net. This can be achieved by adding a new place for any pair of conflicting transitions and then

extending the presets of the transitions with the corresponding new places. The formalisation of this procedure is provided below.

Consider a single-execution IPT $N = \langle S, T, F, I, \mathsf{m}\rangle$. Define the set $\mathsf{confl}(N)$ of pairs of transitions in conflict as

$$\mathsf{confl}(N) = \{T' \subset T \mid |T'| = 2 \ \wedge \ \forall X \in \mathsf{St}(N).\ T' \not\subseteq X\}.$$

Then, the conflict-saturated version of N is given by $N^{\mathsf{confl}(N)} = \langle S', T, F', I, \mathsf{m}'\rangle$ where

- $S' = S \cup \{s_T \mid T' \in \mathsf{confl}(N)\}$,
- $F' = F \cup \{(s'_T, t) \mid t \in T'\}$;
- $\mathsf{m} = \mathsf{m} \cup \{s_T \mid T' \in \mathsf{confl}(N)\}$.

It should be noted that for every transition $t \in T$ and each reachable marking $m \in \mathcal{M}_N$ such that $m\,[t\rangle$, there exists a reachable marking $m' \in \mathcal{M}_{N^{\mathsf{confl}(N)}}$ such that $m'\,[t\rangle$ and $\forall s \in S.\ m'(s) = m(s)$. In other words, the transformation preserves the behaviour despite m and m' may differ in the places added by the transformation. The correspondence also holds in the opposite direction: for each transition $t \in T$ and each reachable marking $m' \in \mathcal{M}_{N^{\mathsf{confl}(N)}}$ such that $m'\,[t\rangle$, there exists a reachable marking $m \in \mathcal{M}_N$ such that $m\,[t\rangle$ and $\forall s \in S.\ m'(s) = m(s)$.

The construction above provides a procedure for obtaining an equivalent, conflict-saturated version of an IPT, as stressed by the following proposition.

PROPOSITION 8.3. *Let N be a single-execution* IPT*, then $N^{\mathsf{confl}(N)} \equiv N$.*

PROOF. Straightforward, as N is a single-execution IPT, the added places do not have any incoming arc and are initially marked, and the transitions in their postsets are in conflict. □

It is worth stressing that PCNs and CNs are single execution nets; moreover, CNs are also conflict-saturated.

8.1 Occurrence nets

We recall the notion of *occurrence* nets[1]. We adopt the usual convention by which places and transitions are called *conditions* and *events*; and write B and E for their respective sets (instead of S and T) and c for the initial marking. We may confuse conditions with places and events with transitions. Moreover, we will omit the inhibiting relation I since occurrence nets do not have inhibitor arcs (i.e., $I = \emptyset$).

Definition 8.4. An *occurrence net* (ON) $O = \langle B, E, F, \mathsf{c}\rangle$ is an acyclic, safe net satisfying the following restrictions:

(1) $\forall b \in B.\ {}^\bullet b$ is either empty or a singleton, and $\forall b \in \mathsf{c}.\ {}^\bullet b = \emptyset$,
(2) $\forall b \in B.\ \exists b' \in \mathsf{c}$ such that $b' \leq_O b$,
(3) for all $e \in E$ the set $\{e' \in E \mid e' \leq_O e\}$ is finite, and
(4) $\#$ is an irreflexive and symmetric relation defined as follows:
   - $e \#_0 e'$ iff $e, e' \in E$, $e \neq e'$ and ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$,
   - $x \# x'$ iff $\exists y, y' \in E$ such that $y \#_0 y'$ and $y \leq_O x$ and $y' \leq_O x'$.

Each condition b in an ON represents the occurrence of a token. Hence, b either belongs to the initial marking c or is produced by the *unique* event in ${}^\bullet b$. The flow relation is interpreted as the *causality* relation among the elements of the

[1]Occurrence nets are often the result of the *unfolding* of a (safe) net. In this perspective an occurrence net is meant to describe precisely the non-sequential semantics of a net, and each reachable marking of the occurrence net corresponds to a reachable marking of the unfolded net. Here we focus purely on occurrence nets and not on the nets they are the unfoldings of, nor on the relation between the net and its unfolding.

net; for this reason we say that x *causally depends* on y iff $y \leq_O x$. Observe that $\leq_O \cap (E \times E)$ is a partial order and # is inherited along $\leq_O$; hence, $e \# e' \leq_O e''$ implies $e \# e''$.

Proposition 8.5. *Let $O = \langle B, E, F, \mathsf{c}\rangle$ be an occurrence net. Then, O is a single execution net.*

As for Proposition 8.3, every single execution IPT can be converted into an equivalent conflict-saturated net. Since the construction does not modify the inhibitor relation, the same holds for occurrence nets.

Definition 8.6. Let $O = \langle B, E, F, \mathsf{c}\rangle$ be an ON. A set of events $X \subseteq E$ is a *configuration* of O if:

(1) $\forall e, e' \in X. e \neq e' \implies \neg(e \# e')$, i.e., it is *conflict-free*; and
(2) $\forall e \in X. \lfloor e \rfloor \subseteq X$, i.e., it is *left-closed*.

The set of configurations of O is denoted by $\mathsf{Conf}_{\text{ON}}(O)$.

If a ON is conflict-saturated then the first condition can be rewritten as $\forall e, e' \in X. {}^\bullet e \cap {}^\bullet e' \neq \emptyset \implies e = e'$. Moreover, observe that each configuration of an ON is also a state.

8.2 From ONs to CNs

We demonstrate that it is possible to transform every occurrence net into a causal one, establishing that the latter notion constitutes a conservative extension of the former.

Definition 8.7. Given an ON $O = \langle B, E, F, \mathsf{c}\rangle$ we can associate with it a net $\mathcal{O}(O)$ defined as $\langle S, E, F', I, \mathsf{m}\rangle$ where

(1) $S = \{(*, e) \mid e \in E\} \cup \{(e, *) \mid e \in E\} \cup \{(\{e, e'\}, \#) \mid e \# e'\}$;
(2) $F' = \{(s, e) \mid s = (*, e) \vee (s = (W, \#) \wedge e \in W)\} \cup \{(e, s) \mid s = (e, *)\}$;
(3) $I = \{(s, e) \mid s = (*, e') \wedge e' <_C e\}$; and
(4) $\mathsf{m} : S \to \mathbb{N}$ is such that $\mathsf{m}(s) = 0$ if $s = (e, *)$ and $\mathsf{m}(s) = 1$ otherwise,

The construction resembles the one from PESes to CNs: each event e of an occurrence net is associated with a homonymous transition and two places $(*, e)$ and $(e, *)$ in the corresponding causal net. When marked, $(*, e)$ represents the fact that e has not been fired yet, while $(e, *)$ describes the fact that e has been executed. Moreover, there is one additional place $(\{e, e'\}, \#)$ for any pair of conflicting events $e \# e'$ in O. As expected, the preset of an event e consists of the place $(*, e)$ and all the places of the form $(\{e, e'\}, \#)$, which represent the conflicts of e with some other event e'. Causal dependencies are mapped into inhibitor arcs so that $(e', *)$ belongs to the inhibitor set of e only if e causally depends on e' in O. The initial marking assigns a token to every place belonging to the preset of some transition.

Example 8.8. Figure 13 shows the encoding of the ON O in Figure 13a as the CN $\mathcal{O}(O)$ in Figure 13b.

Proposition 8.9. *Let O be an occurrence net. Then, $\mathcal{O}(O)$ is a CN and $O \equiv \mathcal{O}(O)$.*

Proof. Consider the ON $O = \langle B, E, F, \mathsf{c}\rangle$ and let $\mathcal{O}(O) = \langle S, E, F', I, \mathsf{m}\rangle$ be its corresponding net. We first show that $\mathcal{O}(O)$ is a CN.

The requirements (1), (2), and (7) in Definition 5.1 are trivially satisfied by the construction of $\mathcal{O}(O)$. For the third one, we proceed as follows: observe that ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$ implies $e \# e'$ and, by construction, ${}^\bullet e \cap {}^\bullet e' = \{(\{e, e'\}, \#)\}$ but $(\{e, e'\}, \#)$ does not belong to the inhibitor set of any transition in $\mathcal{O}(O)$. Condition (4) reduces to showing that $\lfloor e \rfloor_<$ is finite, which holds because $\lfloor e \rfloor$ is so in O. Condition (5) follows because O is an ON, and hence $F^+ \cap E \times E$ is acyclic.

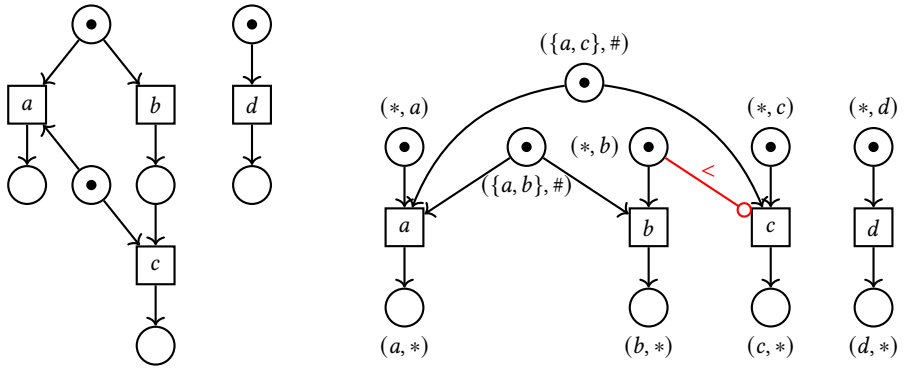


Fig. 13. An occurrence net as a causal net

Finally observe that $\lfloor e \rfloor_{\lessdot} = \lfloor e \rfloor$ and this implies that for all $e', e'' \in \lfloor e \rfloor_{\lessdot}$, if $e' \natural e''$ then $e' = e''$. Therefore, we can conclude that $\mathcal{O}(O)$ is indeed a CN.

We now show that $\mathrm{St}(O) = \mathrm{St}(\mathcal{O}(O))$.

- $\mathrm{St}(O) \subseteq \mathrm{St}(\mathcal{O}(O))$. Assume $X \in \mathrm{St}(O)$ and a firing sequence σ such that $X = X_\sigma$. We prove by induction on the length of a firing sequence $\sigma \in \mathcal{R}_{\mathsf{c}}^{O}$ that there exists a firing sequence $\sigma' \in \mathcal{R}_{\mathsf{m}}^{\mathcal{O}(O)}$ such that $X_\sigma = X_{\sigma'}$. The base case ($len(\sigma) = 0$) is immediate since $0 \in \mathrm{St}(O)$ and $0 \in \mathrm{St}(\mathcal{O}(O))$ holds. For the inductive case, we assume that $len(\sigma) = n$ and $\sigma \in \mathcal{R}_{\mathsf{c}}^{O}$ implies there exists $\sigma' \in \mathcal{R}_{\mathsf{m}}^{\mathcal{O}(O)}$ such that $X_\sigma = X_{\sigma'}$. Assume now that $\sigma\,[e\rangle\,c'$. Then, $X_\sigma \cup \{e\}$ is conflict-free, and $\lfloor e \rfloor \subseteq X_\sigma$. Hence, $X_{\sigma'} \cup \{e\}$ is conflict-free, and $\lfloor e \rfloor_{\lessdot} \subseteq X_{\sigma'}$. Therefore, $\sigma'\,[e\rangle$.
- $\mathrm{St}(\mathcal{O}(O)) \subseteq \mathrm{St}(O)$ follows along the lines of the case above.

Hence $O \equiv \mathcal{O}(O)$. □

We stress that $\mathcal{O}(O)$ is conflict-saturated for any ON O.

8.3 From PCNS to ONS

The encoding from PCNS to ONS basically requires to embed causal dependencies expressed in terms of inhibitor arcs into the the flow relation. For this reason, the encoding incorporates new places of the form (t, t') as a way for representing the dependency $t \lessdot t'$.

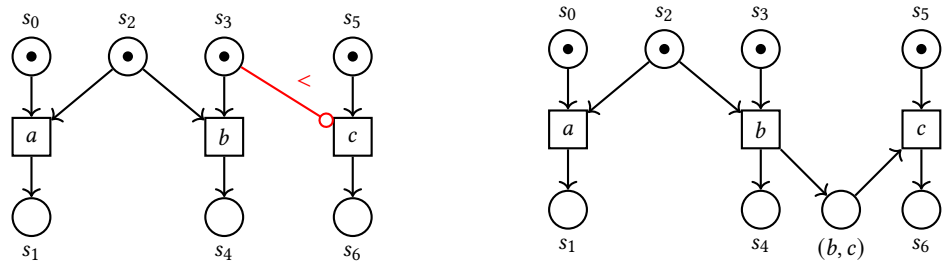
Definition 8.10. Let $C = \langle S, T, F, I, \mathsf{m}\rangle$ be a PCN. The associated ON is a net $\mathcal{Z}(C) = \langle B, T, F', \mathsf{c}\rangle$ where

- $B = S \cup \{(t, t') \mid t \lessdot t'\}$;
- $F' = F \cup \{(s, t) \mid s = (t', t)\} \cup \{(t, s) \mid s = (t, t')\}$; and
- $\mathsf{c} : B \to \mathbb{N}$ is such that $\mathsf{c}(b) = \mathsf{m}(b)$ if $b \in S$ and $\mathsf{c}(b) = 0$ if $b \notin S$.

Note that the flow relation is extended to account for causal dependencies: each transition t produces tokens in the places (t, t') (i.e., $(t, (t, t')) \in F$) and consumes tokens from (t'', t) (i.e., $(t, (t'', t))$). In this way, a transition t is enabled when all its causes have been already fired (i.e., the places (t'', t) are marked).

The following results highlights that the transformation preserves the behaviour of the net.

PROPOSITION 8.11. *Let C be a PCN, then $\mathcal{Z}(C)$ is an ON and $C \equiv \mathcal{Z}(C)$.*



(a) C (b) $\mathcal{Z}(C)$

Fig. 14. A causal net as an occurrence net

Proof. Take the CN $C = \langle S, T, F, \mathsf{m}\rangle$. We first check that $\mathcal{Z}(C) = \langle B, T, F', \mathsf{c}\rangle$ is an ON. Observe that each place in a CN has at most one input arc. The new conditions introduced in the transformation have just one input transition, so for all $b \in B$, it holds that $|{}^\bullet b| \leq 1$. Furthermore, the added conditions are not marked, and since c is equal to m when the condition in S are considered, we have that for all $b \in \mathsf{c}$, ${}^\bullet b = \emptyset$ holds. The transitive closure of F' restricted to $T \times T$ coincides with $\lessdot$, hence $\lfloor t \rfloor_{<_{F'}}$ is finite as $\lfloor t \rfloor_{\lessdot}$ is finite, and furthermore $<_{F'}$ is an irreflexive partial order, and the net is acyclic. We prove that the conflict relation in $\mathcal{Z}(C)$ is inherited along the causality one. Assume that $t \#_0 t'$ as ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ and assume that $t' < t''$, which means that there is a condition $b = (t', t'')$ such that ${}^\bullet b = t'$ and $b^\bullet = t''$. As the token in the place in ${}^\bullet t \cap {}^\bullet t'$ is used by t, the condition b cannot be marked and this implies that $t \# t''$, and $\mathcal{Z}(C)$ is indeed a CN.

We now show that $\mathsf{St}(C) = \mathsf{St}(\mathcal{Z}(C))$.

- $\mathsf{St}(C) \subseteq \mathsf{St}(\mathcal{Z}(C))$. We prove by induction on the length of σ that $\sigma \in \mathcal{R}^C_{\mathsf{m}}$ implies that there exists $\sigma' \in \mathcal{R}^{\mathcal{Z}(C)}_{\mathsf{c}}$ such that $X_\sigma = X_{\sigma'}$. The base case ($len(\sigma) = 0$) is immediate since $0 \in \mathsf{St}(C)$ and $0 \in \mathsf{St}(\mathcal{Z}(C))$ as well. For the inductive case, we assume that $len(\sigma) = n$ and $\sigma \in \mathcal{R}^C_{\mathsf{m}}$ implies there exists $\sigma' \in \mathcal{R}^{\mathcal{Z}(C)}_{\mathsf{c}}$ such that $X_\sigma = X_{\sigma'}$. Assume now that $\sigma\,[t\rangle\, m'$. Now as $lead(\sigma)\,[t\rangle$ it means that $\forall s \in {}^\circ t.\ lead(\sigma)(s) = 0$ thus all the conditions (t', t) are marked for all $t' \in {}^\circ t^\bullet$ and this implies that $lead(\sigma')\,[t\rangle$.
- $\mathsf{St}(\mathcal{Z}(C)) \subseteq \mathsf{St}(C)$ is proved similarly.

Hence $C \equiv \mathcal{Z}(C)$. □

The net $\mathcal{Z}(C)$ in not necessarily conflict-saturated, as C could be not conflict-saturated.

Example 8.12. In Figure 14 we depict a PCN C (on the left) and the associated ON (on the right), where the causal dependency of c on b is represented by the place (b, c).

By combining Proposition 8.9 and Proposition 8.11 we obtain the following result.

Theorem 8.13. *Let C be a causal net. Then $C = \mathcal{O}(\mathcal{Z}(C))$. Let O be an occurrence net. Then $O \equiv \mathcal{Z}(\mathcal{O}(O))$.*

Despite being able to directly obtain a PES from a ON and vice versa [33], the mappings introduced in Sections 6.1 and 6.2 provide an alternative procedure. This procedure relies on the intermediate representation of PESS in terms of CNS, as stated below.

Theorem 8.14. *Let P be a PES. Then $\mathcal{Z}(\mathcal{A}(\mathsf{P}))$ is an ON and $\mathsf{Conf}_{PES}(\mathsf{P}) = \mathsf{Conf}_{ON}(\mathcal{Z}(\mathcal{A}(\mathsf{P})))$. Let O be an ON. Then $\mathcal{Q}(\mathcal{O}(O))$ is a PES and $\mathsf{Conf}_{ON}(O) = \mathsf{Conf}_{PES}(\mathcal{Q}(\mathcal{O}(O)))$.*

8.4 Reversible occurrence nets

A causal-consistent reversible version of ONs has be proposed in [18, 20]. This model associates each reversible event of an occurrence net with a *reversing* transition, similar to our concept of RCNs. As already noticed in [18], this approach encounters limitations when dealing with the full generality of RPEs. The issue arises from the tight correspondence between the configurations of an occurrence net and their associated reachable markings. For an ON $O = \langle B, E, F, \mathrm{c}\rangle$ and a configuration X, the associated reachable marking m_X is computed as $(\mathrm{c} \cup [\![X^\bullet]\!]) \setminus [\![{}^\bullet X]\!]$. This equation makes clear that certain *effects*—i.e., the produced tokens— resulting from the execution of events may remain *observable* in the reachable marking. Specifically, if $e \in X$ and $e^\bullet \cap {}^\bullet X \neq \emptyset$, then $e^\bullet \not\subseteq m_X$. This scenario is exemplified in the configuration $\{b, c\}$ of the ON depicted in Figure 14b, where the condition (b, c) is *used* by the firing of c and ceases to hold after c is executed. Assuming that b is an out-of-causal-order reversible event, we introduce the reversing transition $\underline{b}$ that consumes from s_4 and (b, c). Consequently, it becomes evident that $\underline{b}$ is not enabled at m_X. Consequently, the reversal of b would require the reversal of c, thereby enforcing a causally-ordered reversibility model.

9 APPLICATIONS

Out-of-causal order behaviours. Out-of-causal order reversibility was initially developed to model biochemical reactions. However, in this section, we broaden its scope to explore applications beyond biochemistry. We showcase its application in standard programming languages, demonstrating its utility in scenarios such as implementing the *sagas pattern* [9] for managing long-running transactions. The saga pattern has been employed in various contexts since its introduction as a mechanism for managing database transactions. It has found applications in workflow composition languages and service-oriented composition languages. More recently, with the rise of microservices, the saga pattern has proven to be a practical solution for implementing distributed long-running transactions. The concept of long-running transactions typically relies on a weaker notion of atomicity based on compensations [9]. A saga breaks down the steps of a long-running transaction into a series of short, local transactions. In a distributed setting, a local transaction serves as the unit of work performed by a saga participant. Each step may be accompanied by a compensating transaction—a transaction designed to reverse, to the extent possible, the effects of its corresponding local transaction. The core principle guiding saga execution is that the steps are carried out in the order specified by the saga. However, if any step encounters a failure, the compensations for the already executed steps are triggered in an order consistent with the reversal of those executed steps. Moreover, the saga pattern ensures that either all operations complete successfully, or the corresponding compensation transactions are executed to undo the work previously completed. It is essential to emphasise that while perfect reversal is attainable in certain situations, in several others, reversals are partial. Consequently, the execution of a saga is not atomic in the conventional sense, as partial effects of its execution may become observable when a saga is rolled back.

Consider a business process for handling orders, composed of the following three services:

- o: Placement order service that registers information about the order.
- p: Payment service responsible for charging the customer's credit card.
- f: Fulfillment of the order.

Services o and p are designed to execute in parallel, while the activation of f depends on the completion of the first two actions. The fulfillment service can either succeed or fail. In the former case, the process concludes successfully; in the latter case, the process is aborted. When fulfillment fails, compensation is required for the preceding steps.

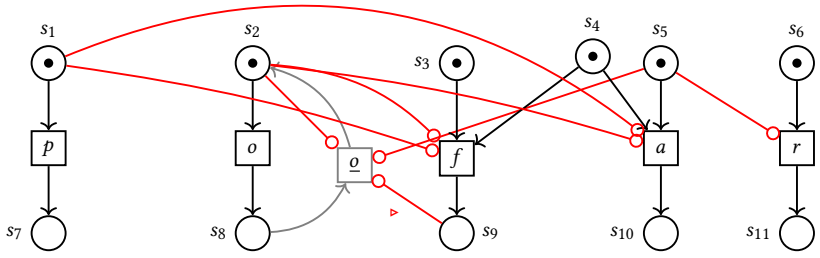


Fig. 15. The RCN associated to the RPES that models a saga $\mathsf{P_S}$.

While o can be perfectly undone, i.e., there exists a reversing transaction $\underline{o}$ that completely rolls back the effects of performing o (e.g., resetting the information in the database), the service p is not reversible but compensatable. In case of need, a transaction r for refunding the charge on the credit card should be invoked. For simplicity, we assume that neither o nor p fails.

The above saga, can be represented as an RPES $\mathsf{P_S} = (E, U, <, \#, \prec, \triangleright)$ defined as follows:

$$E = \{o, p, f, a, r\} \qquad U = \{o\} \qquad < = \{(o, f), (p, f), (o, a), (p, a), (a, r)\}$$
$$\# = \{(f, a), (a, f)\} \qquad \prec = \{(o, \underline{o}), (a, \underline{o})\} \qquad \triangleright = \{(f, \underline{o})\}$$

The observable events in E can be interpreted as follows:

- o: Successful completion of the placement order service.
- p: Successful completion of the payment service.
- f: Successful completion of the order fulfillment service.
- a: Abortion of the process due to the failure of the fulfillment service.
- r: Completion of the credit card refunding.

As described earlier, the sole reversible action is the placement of an order (o). The causal dependencies in $<$ dictate that the fulfillment event (f) and the abortion (a) can only occur after o and p ocurr. Moreover, the refunding r can only happen after abortion a occurs. The conflict relation states that the process either completes successfully (f) or aborts (a). The prevention relation $\triangleright$ states that the placement of an order cannot be undone if the order has been fulfilled. The reverse causality dictates that the placement of an order can be reversed only if the order has been placed (o) and the process has been aborted (a).

The RPES described above can be associated with the RCN depicted in Figure 15 using the constructions provided in the previous sections. We now comment on the structure of the net. The successful completion of o and p enabling f is represented by the inhibitor arcs connecting f to s_1 and s_2. Similarly, the arcs connecting the abortion event a to s_1 and s_2 signify that the process can abort only after the execution of o and p. The fact that the process completes either with the successful fulfillment f or the abortion is captured by the events f and a being in conflict due to the place in s_4. The refunding event r is only executed in case of abortion, as indicated by the arc connecting s_5 with r, ensuring that the compensating event r is enabled only when fulfillment aborts (consuming the token in s_5). Analogously, the occurrence of a enables the reversing of o. Conversely, when f completes successfully, the undoing of o is prevented by the presence of the token in s_9 generated by f.

The implementation of the sagas pattern often requires ingenuity [8, 29]. A saga's implementation involves coordinating the steps of the saga. The coordination logic selects and instructs a participant to execute a local transaction. Upon completion, the sequencing coordination selects and invokes the next saga participant, continuing until all steps of the saga are executed. In the context of microservices, it can be seen as a sequence of local transactions where each update publishes an event, triggering the next update in the sequence [29]. As illustrated by the previous example, the mappings from RPESes to RCN in the previous section make explicit the messages that should be generated to activate the execution of transitions. These constructions provide a helpful blueprint for implementing the sagas pattern, ensuring that the necessary steps are coordinated, and compensations are performed when needed.

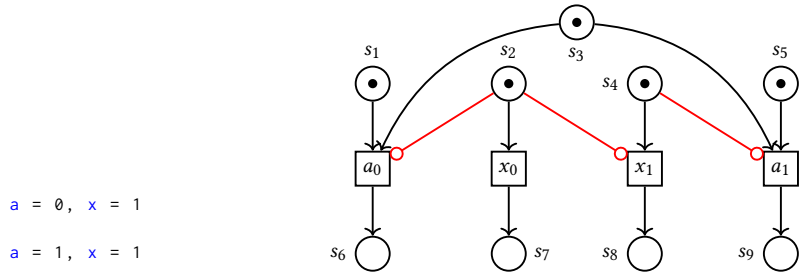
Reversible debugging in shared memory models. One successful application of reversibility is causal-consistent reversible debugging [10, 15]. Reversible debuggers extend classical ones by allowing the user to go back in the execution, thereby making it easier to find the source of a bug. Causal-consistent reversible debuggers further improve on reversible ones by leveraging causal information while undoing computations. While this technique has been applied to message-passing concurrent systems (e.g., actor-like languages), it has not been applied to shared-memory-based concurrency so far. One of the main challenges is accommodating asymmetric conflicts. Asymmetric conflicts arise when two independent actions, both of which can occur in a computation, cannot happen in any order. In other words, it may happen that one action prevents the other, but not vice versa. Therefore, both actions can coexist in a computation only if executed in a particular order.

```
1   int x = 0; //global shared variable
2   void *tOne(void *vargp)
3   {
4     int a = x; //thread local variable
5   }
6
7   int main()
8   {
9     pthread_t thread_id1;
10    pthread_create(&thread_id1, NULL, tOne, NULL);
11    x++;
12    return 1;
13  }
```

Fig. 16. Snippet of C code

Asymmetric conflicts are typical in shared memory scenarios. Let us consider the snippet of C code listed in Figure 16. This fragment of code illustrates a scenario where a thread read the current value of a shared global variable, storing it in a local variable. A global variable `x` is declared and initialised to `0` (line 1). Then, a thread function (`tOne`) is defined. The thread function declares a local variable (`a`), initialized with the current value of the global variable `x`. The main function creates a thread, assigning it to execute the `tOne` function. Then, the global variable `x` is incremented. The order of execution of the thread and the main program is not guaranteed, leading to non-deterministic behaviour and potential race conditions. That is, depending on how the scheduler executes the thread, its local variable can be initialised with either 0 or 1 (assuming sequential consistency). Therefore, any valid and complete execution of the program lead the system to one of the configurations (i.e., state variables) in Figure 17a.

The program's behaviour can be effectively described using a PES. Specifically, one approach is to describe computations in terms of the values assigned to variables. Accordingly, events may be selected to represent these value



(a) Configurations (b) N_V

Fig. 17. An CN model of the C Code

assignments. For clarity, we denote events such as x_0 and x_1 to signify instances where variable x is assigned the values 0 and 1, respectively. Analogously, events like a_0 and a_1 are employed for the variable a.

Consequently, the program's behaviour can be defined using the PES $\mathsf{P}_V = (E, <, \#)$, where

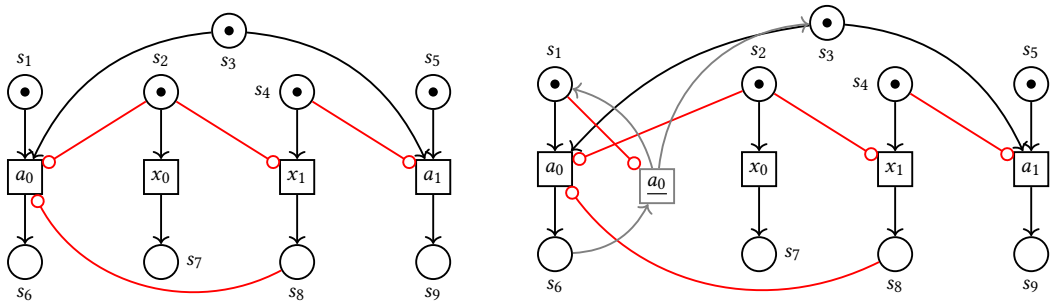
$$E = \{a_0, a_1, x_0, x_1\} \qquad < = \{(x_0, x_1), (x_0, a_0), (x_1, a_1)\} \qquad \# = \{(a_0, a_1), (a_1, a_0)\}$$

The causality relation establishes that x_0 is the minimal event since the assignment of 0 to x is the first one to occur in the program. The relation $x_0 < x_1$ signifies that the assignment of 1 due to the increment in the program follows the initial assignment. The remaining pairs in the definition of the causality relation capture the dependency of the value assigned to the local variable on the value of the global variable. For example, (x_0, a_0) indicates that the assignment to a occurs after reading the value 0 from x, while (x_1, a_1) denotes the case where the value read from x is 1.

Furthermore, it is noteworthy that both x_0 and x_1 coexist in every complete execution, as x is initially assigned 0 and subsequently 1. This is not the case for the local variable a, if a is assigned 0 in one execution, it cannot be assigned 1 in the same execution (and vice versa). This fact is captured by the conflict relation, which states that the different assignments to the local variable are in conflict, i.e., $a_0 \# a_1$.

The network depicted in Figure 17b is obtained by applying the encoding outlined in the previous sections. While the proposed model accurately captures maximal configurations, it is noteworthy that it exhibits a weakness in constraining the order in which certain assignments can take place. For instance, the state $\{x_0, x_1\}$ allows the firing of a_0. This suggests that the model does not explicitly account for the fact that the value assigned to a has been read before the assignment of 1 to x. While this behaviour may be acceptable in certain contexts, its justification becomes less evident when utilising the model for debugging.

Consider a scenario where the program has reached its maximal configuration $\{x_0, x_1, a_0\}$ with reverse debugging activated. Given that a_0 is a maximal event, a reverse of a_0 would generate tokens in s_1 and s_3, consequently enabling both a_1 and a_0. Notice that, if computing forward, the execution of a_0 should be precluded, as x_1 is already present in the configuration, and thus, the variable x holds the value 1. The model must explicitly state that the occurrence of event a_0 should be precluded when the variable x has value 1. We can enhance the model by introducing inhibitor arcs to account for these dependencies. In our *r*CN, we use inhibitor arcs between the postset of a transition and a reversing transition, signifying the prevention relation. This mechanism can be extended to handle asymmetric conflicts, such as those between a_0 and x_1. Specifically, we can broaden the prevention relation to include forward events (which is not the case of RPES) , e.g., $a_0 \triangleright x_1$, indicating that a_0 is prevented by the presence of x_1. This relationship can be



(a) N_V extended with the prevention relation between forward transitions

(b) Reversible version of the extended model

Fig. 18. A reversible model of the C code

effectively modelled with an inhibitor arc linking the postset of x_1 to the transition a_0, as illustrated in Figure 18a. Note the inhibitor arc connecting a_0 with s_8. This suggest that we can extend our results to (reversible) Asymmetric Event Structures.

Then, an accurrate reversible model of the code is straightforwardly obtained by adding the reversing transitions in a standard fashion, as depicted in Figure 18b.

10 CONCLUSIONS

The main contribution of this paper is the characterisation of a class of Petri nets with inhibitor arcs, dubbed *reversible causal nets* (RCN), which provides an operationally counterpart for the reversibility model behind reversible prime event structures (RPES) [26]. The key observation is that three out of the four (primitive) relations that RPESes define over events can be captured by inhibitor arcs: both *forward and reverse causality* can be modelled as inhibitor arcs that connect an event (either forward or reversing) to (some of the conditions in) the presets of its immediate causes, while *prevention* is translated to arcs that connect an event to (some of the conditions in) the postsets of the preventing events. The remaining *conflict relation* corresponds, as usual, to overlapped presets.

Figure 19 briefly recaps our results with respect to the state of the art. The correspondence between occurrence nets and prime event structures is due to Winskel's seminal work [33]. We have introduced a new class of Petri nets, called *causal nets*, and showed their correspondence with prime event structures (Theorem 6.9). We have then presented the reversible variant of causal nets (RCNS), and showed that they are an operational counterpart of the reversible prime event structures of [26] (Theorem 7.19). Finally, we showed that causal nets are tightly connected to occurrence nets (Theorem 8.13), and give a revisitation of Winskel's correspondence via Theorem 8.14. To give a complete picture of known results, we depict the extension of occurrence nets into reversible occurrence nets proposed in [18]; where the latter ones correspond to a proper subclass of *r*PES that just accounts for causally-consistent reversibility (CRPES). We plan to implement our framework into Agda so to automatically check the correctness of our results and of the encodings.

In section 7.5 we hinted what subclass of RCNS obeys to causal consistent reversibility. We leave as future work to formally prove from the axioms provided in [14] that such subclass is causal consistent with respect to reversibility.

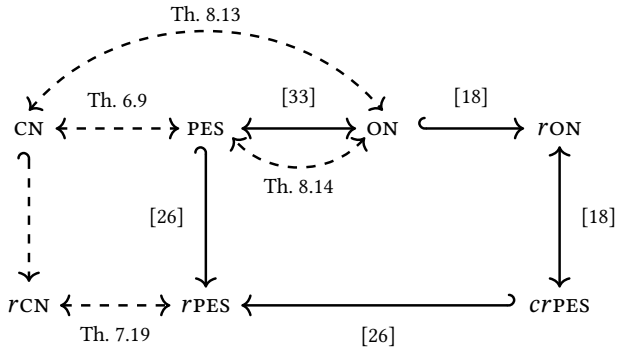


Fig. 19. Recap of results. Solid arrows represent known results/works in the literature while dashed arrows stand for results in this paper. Double-headed arrows are used for correspondences while curly tailed arrows are for inclusions and reversible extensions.

Besides contributing to the well-established line of work initiated in [23, 33] that links classes of nets with classes of event structures, our work explains the reversing mechanisms of RPESes by relying on a well-studied concurrent model. In this respect, our contribution constitutes a step forward with respect to [18], which just deals with the subclass of *causal-reversible* RPESes which, e.g., do not provide out-of-causal-order reversibility. Remarkably, our reversible causal nets do not appeal to original elements to achieve reversibility as, e.g., the *bonds* of [25]. As a matter of fact, causal nets do not rely either on the information (i.e., colours) carried on by tokens (as done in [3, 20]).

The characterisation of RPESes as restrictions over the structure of the nets enables the study of less constrained models. For instance, an undoable event in an RPES can be undone just by performing one reversing event. This excludes the possibility of defining alternative conditions for reversing an event. We can overcome this limitation in the operational model just by dropping the requirement of non-overlapping backward transitions (i.e., $[\![{}^\bullet\underline{T}]\!] = {}^\bullet\underline{T}$). This kind of analysis seems an interesting line to explore in future work.

Causal nets and their reversible versions play a role analogous to occurrence nets (in their connection with PESes). Consequently, we may ask ourselves about the class of (reversible) nets whose unfoldings are (reversible) causal nets, or, alternatively, which are the suitable categories that would allow us, if possible, to extend Winskel's chain of coreflection to reversible event structures.

ACKNOWLEDGMENT

This work has been partially supported by the BehAPI project funded by the EU H2020 RISE under the Marie Sklodowska-Curie action (No: 778233), by the Italian PRIN 2020 project NiRvAna – Noninterference and Reversibility Analysis in Private Blockchains, the Italian PRIN 2022 project DeKLA – Developing Kleene Logics and their Applications, the INdAM-GNCS E53C22001930001 project RISICO – Reversibilità in Sistemi Concorrenti: Analisi Quantitative e Funzionali, and the European Union - NextGenerationEU SEcurity and RIghts in the CyberSpace (SERICS) Research and Innovation Program PE00000014, projects STRIDE and SWOP.

We would like to thank the anonymous reviewers for their useful suggestions and criticisms that helped us to improve the paper. Many thanks also to Marco Dessalvi for implementing part of our findings.

REFERENCES

[1] Bogdan Aman, Gabriel Ciobanu, Robert Glück, Robin Kaarsgaard, Jarkko Kari, Martin Kutrib, Ivan Lanese, Claudio Antares Mezzina, Lukasz Mikulski, Rajagopal Nagarajan, Iain C. C. Phillips, G. Michele Pinna, Luca Prigioniero, Irek Ulidowski, and Germán Vidal. 2020. Foundations of Reversible Computation. In *Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action IC1405*. Lecture Notes in Computer Science, Vol. 12070. Springer, 1–40. https://doi.org/10.1007/978-3-030-47361-7_1

[2] Paolo Baldan, Nadia Busi, Andrea Corradini, and G. Michele Pinna. 2004. Domain and Event Structure Semantics for Petri Nets with Read and Inhibitor Arcs. *Theoretical Computer Science* 323, 1-3 (2004), 129–189.

[3] Kamila Barylska, Anna Gogolinska, Lukasz Mikulski, Anna Philippou, Marcin Piatkowski, and Kyriaki Psara. 2018. Reversing Computations Modelled by Coloured Petri Nets. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2018 Satellite event of the conferences: 39th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2018 and 18th International Conference on Application of Concurrency to System Design ACSD (CEUR Workshop Proceedings, Vol. 2115)*. CEUR-WS.org, 91–111. http://ceur-ws.org/Vol-2115

[4] Gérard Boudol. 1990. Flow Event Structures and Flow Nets. In *Semantics of Systems of Concurrent Processes (Lecture Notes in Computer Science, Vol. 469)*, Irène Guessarian (Ed.). Springer, 62–95. https://doi.org/10.1007/3-540-53479-2_4

[5] Giovanni Casu and G. Michele Pinna. 2014. Flow Unfolding of Multi-clock Nets. In *Petri Nets 2014 (Lecture Notes in Computer Science, Vol. 8489)*, Gianfranco Ciardo and Ekkart Kindler (Eds.). Springer, 170–189. https://doi.org/10.1007/978-3-319-07734-5_10

[6] Giovanni Casu and G. Michele Pinna. 2017. Petri nets and dynamic causality for service-oriented computations. In *Proceedings of SAC 2017*, Ahmed Seffah, Birgit Penzenstadler, Carina Alves, and Xin Peng (Eds.). ACM, 1326–1333. https://doi.org/10.1145/3019612.3019806

[7] Vincent Danos and Jean Krivine. 2005. Transactions in RCCS. In *Concurrency Theory, 16th International Conference, CONCUR 2005 (Lecture Notes in Computer Science, Vol. 3653)*. Springer, 398–412. https://doi.org/10.1007/11539452_31

[8] Neal Ford, Mark Richards, Pramod Sadalage, and Zhamak Dehghani. 2021. *Software Architecture: The Hard Parts*. " O'Reilly Media, Inc.".

[9] Hector Garcia-Molina and Kenneth Salem. 1987. Sagas. In *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference*, Umeshwar Dayal and Irving L. Traiger (Eds.). ACM Press, 249–259. https://doi.org/10.1145/38713.38742

[10] Elena Giachino, Ivan Lanese, and Claudio Antares Mezzina. 2014. Causal-Consistent Reversible Debugging. In *Fundamental Approaches to Software Engineering - 17th International Conference, FASE 2014 (Lecture Notes in Computer Science, Vol. 8411)*. Springer, 370–384. https://doi.org/10.1007/978-3-642-54804-8_26

[11] Stefan Kuhn, Bogdan Aman, Gabriel Ciobanu, Anna Philippou, Kyriaki Psara, and Irek Ulidowski. 2020. Reversibility in Chemical Reactions. In *Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action IC1405*. Lecture Notes in Computer Science, Vol. 12070. Springer, 151–176. https://doi.org/10.1007/978-3-030-47361-7_7

[12] Ivan Lanese, Michael Lienhardt, Claudio Antares Mezzina, Alan Schmitt, and Jean-Bernard Stefani. 2013. Concurrent Flexible Reversibility. In *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013 (Lecture Notes in Computer Science, Vol. 7792)*. Springer, 370–390. https://doi.org/10.1007/978-3-642-37036-6_21

[13] Ivan Lanese, Adrián Palacios, and Germán Vidal. 2019. Causal-Consistent Replay Debugging for Message Passing Programs. In *Formal Techniques for Distributed Objects, Components, and Systems - 39th IFIP WG 6.1 International Conference, FORTE 2019 (Lecture Notes in Computer Science, Vol. 11535)*. Springer, 167–184. https://doi.org/10.1007/978-3-030-21759-4_10

[14] Ivan Lanese, Iain C. C. Phillips, and Irek Ulidowski. 2024. An Axiomatic Theory for Reversible Computation. *ACM Trans. Comput. Log.* 25, 2 (2024), 11:1–11:40. https://doi.org/10.1145/3648474

[15] Ivan Lanese, Ulrik Pagh Schultz, and Irek Ulidowski. 2022. Reversible Computing in Debugging of Erlang Programs. *IT Prof.* 24, 1 (2022), 74–80. https://doi.org/10.1109/MITP.2021.3117920

[16] Rom Langerak. 1993. Bundle Event Structures: A Non-interleaving Semantics for LOTOS. In *FORTE '92 Conference Proceedings (IFIP Transactions, Vol. C-10)*, Michel Diaz and Roland Groz (Eds.). North-Holland, 331–346.

[17] Doriana Medic, Claudio Antares Mezzina, Iain Phillips, and Nobuko Yoshida. 2020. Towards a Formal Account for Software Transactional Memory. In *Reversible Computation - 12th International Conference, RC 2020 (Lecture Notes in Computer Science, Vol. 12227)*. Springer, 255–263. https://doi.org/10.1007/978-3-030-52482-1_16

[18] Hernán C. Melgratti, Claudio Antares Mezzina, Iain Phillips, G. Michele Pinna, and Irek Ulidowski. 2020. Reversible Occurrence Nets and Causal Reversible Prime Event Structures. In *Reversible Computation - 12th International Conference, RC 2020 (Lecture Notes in Computer Science, Vol. 12227)*. Springer, 35–53. https://doi.org/10.1007/978-3-030-52482-1

[19] Hernán C. Melgratti, Claudio Antares Mezzina, and G. Michele Pinna. 2021. A distributed operational view of Reversible Prime Event Structures. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. IEEE, 1–13. https://doi.org/10.1109/LICS52264.2021.9470623

[20] Hernán C. Melgratti, Claudio Antares Mezzina, and Irek Ulidowski. 2020. Reversing Place Transition Nets. *Log. Methods Comput. Sci.* 16, 4 (2020). https://lmcs.episciences.org/6843

[21] Claudio Antares Mezzina, Rudolf Schlatte, Robert Glück, Tue Haulund, James Hoey, Martin Holm Cservenka, Ivan Lanese, Torben Æ. Mogensen, Harun Siljak, Ulrik Pagh Schultz, and Irek Ulidowski. 2020. Software and Reversible Systems: A Survey of Recent Activities. In *Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action IC1405*. Lecture Notes in Computer Science, Vol. 12070. Springer, 41–59. https://doi.org/10.1007/978-3-030-47361-7_1

[22] Ugo Montanari and Francesca Rossi. 1995. Contextual Nets. *Acta Informatica* 32, 6 (1995), 545–596. https://doi.org/10.1007/BF01178907

[23] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. 1981. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science* 13 (1981), 85–108. https://doi.org/10.1016/0304-3975(81)90112-2

[24] Michael A. Nielsen and Isaac L. Chuang. 2016. *Quantum Computation and Quantum Information (10th Anniversary edition).* Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

[25] Anna Philippou and Kyriaki Psara. 2018. Reversible Computation in Petri Nets. In *Reversible Computation - 10th International Conference, RC 2018 (Lecture Notes in Computer Science, Vol. 11106)*, Jarkko Kari and Irek Ulidowski (Eds.). Springer, 84–101. https://doi.org/10.1007/978-3-319-99498-7_6

[26] Iain Phillips and Irek Ulidowski. 2015. Reversibility and asymmetric conflict in event structures. *J. Log. Algebraic Methods Program.* 84, 6 (2015), 781–805. https://doi.org/10.1016/J.JLAMP.2015.07.004

[27] Iain Phillips, Irek Ulidowski, and Shoji Yuen. 2013. A Reversible Process Calculus and the Modelling of the ERK Signalling Pathway. In *Reversible Computation, 4th International Workshop, RC 2012. Revised Papers (Lecture Notes in Computer Science, Vol. 7581).* Springer, 218–232. https://doi.org/10.1007/978-3-642-36315-3_18

[28] G. Michele Pinna. 2011. How much is worth to remember? A taxonomy based on Petri Nets Unfoldings. In *PETRI NETS 2011 (Lecture Notes in Computer Science, Vol. 6709)*, Lars Michael Kristensen and Laure Petrucci (Eds.). 109–128. https://doi.org/10.1007/978-3-642-21834-7_7

[29] Chris Richardson. 2018. *Microservices patterns: with examples in Java.* Simon and Schuster.

[30] Rob J. van Glabbeek and Gordon D. Plotkin. 1995. Configuration Structures. In *10th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. IEEE Computer Society Press, 199–209. https://doi.org/10.1109/LICS.1995.523257

[31] Rob J. van Glabbeek and Gordon D. Plotkin. 2004. Event Structures for Resolvable Conflict.. In *MFCS'04 Conference Proceedings (Lecture Notes in Computer Science, Vol. 3153)*, Jirí Fiala, Václav Koubek, and Jan Kratochvíl (Eds.). Springer, 550–561. https://doi.org/10.1007/978-3-540-28629-5_42

[32] Rob J. van Glabbeek and Gordon D. Plotkin. 2009. Configuration structures, event structures and Petri nets. *Theoretical Computer Science* 410, 41 (2009), 4111–4159. https://doi.org/10.1016/j.tcs.2009.06.014

[33] Glynn Winskel. 1986. Event Structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency (Lecture Notes in Computer Science, Vol. 255).* Springer, 325–392. https://doi.org/10.1007/3-540-17906-2_31