Electronic Notes in Theoretical Informatics And Computer Science

ENTICS
https://entics.episciences.org

Volume 5
Proceedings of
MFPS 2025

Reversible Computations are Computations

Clément Aubert[a,1,2] Jean Krivine[b,3]

[a] *Augusta University*
Augusta, GA, USA

[b] *CNRS, Univ. Paris Cité, IRIF*
Paris, France

Abstract

Causality serves as an abstract notion of time for concurrent systems. A computation is causal, or simply valid, if each observation of a computation event is preceded by the observation of its causes. The present work establishes that this simple requirement is equally relevant when the occurrence of an event is invertible. We propose a conservative extension of causal models for concurrency that accommodates reversible computations. We first model reversible computations using a symmetric residuation operation in the general model of configuration structures. We show that stable configuration structures, which correspond to prime algebraic domains, remain stable under the action of this residuation. We then derive a semantics of reversible computations for prime event structures, which is shown to coincide with a switch operation that dualizes conflict and causality.

Keywords: concurrency, reversible computation, configuration structures, event structures, process calculi, non-interleaving semantics

1 Introduction

In physics, the concept of time being "reversible" stems from the fact that many fundamental physical laws are symmetric with respect to time. This means that the equations governing certain physical processes remain unchanged whether time moves forward or backward. For example, applying Newton's laws of motion to a celestial body does not reveal whether the body is moving forward or backward in time, as the equations validate the trajectory in both directions. The present work explores this intriguing idea in the context of models of concurrent systems, where time is abstracted to a weaker notion of causality or precedence. Specifically, it answers positively the following question: *Are standard models of concurrent systems also models of reversible ones?*

This is not a rhetorical question, as unlike time in physics, computations can, in practice, be reversed. Reversible computations have been studied in a variety of contexts [40], such as minimising entropy [10,39,1], resolving deadlocked transactions [36,14], and designing abstract machines and software capable of debugging concurrent execution traces [26,23]. With these objectives, abstract language models for reversible computing have been developed using process algebraic frameworks [13,21,33,11,19,2] or P/T nets [15,6,27].

[1] Email: caubert@augusta.edu
[2] This work has benefited from the support from the National Science Foundation under Grant No. 2242786.
[3] Email: jean.krivine@irif.fr

Rec'd Apr 3,2025; Pub'd Dec 15,2025
10.46298/entics.16677
Proceedings Available Online at
ENTICS MFPS 41 Proceedings

This paper proposes to study such reversible calculi from a denotational perspective, e.g., focusing on relating computation events rather than producing them.

Providing a denotational semantics of reversible concurrent programs has been studied over the last 10 years, in a line of research that considers backward events as first class citizens. It follows that the causal structure of computations induced by backward events is modeled by specific relations, such as reverse causality [35,41,27].

In this work, we aim to determine whether the concept of reversible computations necessitates disrupting the state-of-the art models of concurrency or can instead be seamlessly integrated. To achieve this, we examine event structure semantics for concurrent systems within the same framework established by Winskel and others since the 1980s [9,31,44].

We base the present work on a simple and general model of concurrent systems called configuration structures, where a process is denoted by the configurations (the collection of observable events) it can produce. This approach, which expresses the combination of observations that are allowed or disallowed as a partial order of configurations, was foundational to Winskel's pioneering work and has been extensively studied in subsequent research [5,42,43].

After introducing configuration structures and their semantics (Section 2), we show that reversible computations can be understood as a (partial) group action, which we call symmetric residuation, of configuration structures on themselves (Section 3). This generalizes the notion of residuals, a standard method of retrieving a transition system from a denotational model [8,9], while remaining purely within the simple partial order theory. Configuration structures, in their full generality, can model concurrent systems with arbitrary causal and conflict structures. While these structures may not always correspond to the computation of a traditional algebraic program like the calculus of communicating systems (CCS [29]), a well-known property of these structures is that those satisfying certain stability axioms correspond to prime algebraic domains. These domains provide a semantics for CCS-like calculi *via* prime event structures [44]. The first main result of this paper is to show that **stable configuration structures remain stable under symmetric residuation** (Theorem 5.8), which is a guarantee that symmetric residuation produces configuration structures that are also prime event structures. On these premises, the second main contribution of this paper (Theorem 8.3) is to show that **transporting symmetric residuation to (prime) event structures corresponds to a switch operation** on the graph that represents the causality and conflict relation. The term switch is intended to draw a connection with a class of well-known graph isomorphisms called Seidel switches [37] which is discussed in conclusion (Section 10).

Importantly, we show that the backward or forward orientation of an event exists only relatively to a given past state: like in our example of the celestial body in motion, our approach yields a causal semantics for reversible computation in which the notion of "backward moves" only appears when configuration structures are equipped with a distinguished configuration that we call *referential* (Definition 4.1). We show that these **pointed configuration structures correspond to a form of polarized event structures**, where negative events come and go under the action of our switching operation (Theorem 9.2), providing an adequate model for the explicit orientation of reversible process calculi.

2 Configuration structures

We recall the definition of general configuration structures as presented for instance in [43, Definition 1.1].

Definition 2.1 A *configuration structure* is a pair $\mathbb{C} = \langle E, X \rangle$ where E is a countable set of computation *events* (ranged over by lower case letters a through f) and $X \subseteq \mathcal{P}(E)$ is a set of *configurations* (ranged over by lower case letters x, y and z). A configuration structure is *rooted* when $\emptyset \in X$.

For such configurations, we use the direct notation $x \in \mathbb{C}$ when $x \in X$. In the remainder of the paper we only consider rooted configurations, and use $\mathcal{C}_E$ to denote the set of configuration structures over E.

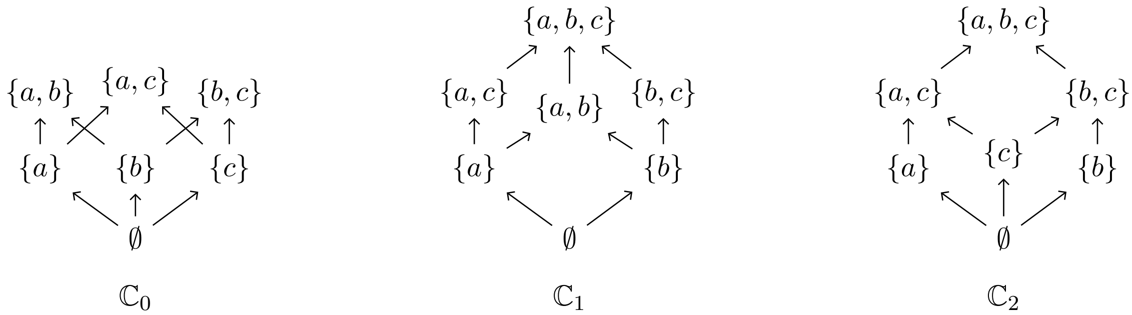
Configurations can be partially ordered by set inclusion. Thus, for a given configuration structure

$\mathbb{C} = \langle E, X\rangle$ and for any $Y \subseteq X$, we can use the standard order theoretic notations:

$$\downarrow_{\mathbb{C}} Y \coloneqq \{x \in \mathbb{C} \mid \exists y \in Y : x \subseteq y\}$$
$$\uparrow^{\mathbb{C}} Y \coloneqq \{x \in \mathbb{C} \mid \exists y \in Y : y \subseteq x\}$$

We also write $x = \bigsqcup^{\mathbb{C}} Y$ whenever Y has a least upper bound $x \in \mathbb{C}$. Similarly $x = \bigsqcap_{\mathbb{C}} Y$ denotes the greatest lower bound of Y when it exists in $\mathbb{C}$. Lastly, for any two configurations $x, y \in \mathbb{C}$, we write $x \frown_{\mathbb{C}} y$ whenever $\bigsqcup^{\mathbb{C}}\{x, y\}$ exists and say that x and y are *compatible in* $\mathbb{C}$. Two configuration structures are *equivalent* if there exists a bijection on events that preserves inclusion of configurations and that equates them.

Example 2.2 We give below three examples of configuration structures of $\mathcal{C}_{\{a,b,c\}}$, where an arrow denotes inclusion. Notice that $\{a\} \frown_{\mathbb{C}_0} \{b\}$, $\{b\} \frown_{\mathbb{C}_0} \{c\}$ and $\{a\} \frown_{\mathbb{C}_0} \{c\}$ but $\bigsqcup^{\mathbb{C}_0}\{a, b, c\}$ is not defined, which models a ternary conflict where at most two events can trigger. Configuration $\mathbb{C}_1$ models a form of disjunctive causality: the event c can be observed provided event a or event b (or both) appeared first. Lastly, configuration $\mathbb{C}_2$ models the fact that events a and b are not compatible unless c is triggered.



Configuration structures can be equipped with an operation of *residuation* [8], which describes which configurations remain reachable once a set of events has been triggered.

Definition 2.3 Let $\mathbb{C} \in \mathcal{C}_E$. For all finite $x \in \mathbb{C}$, we define the *residual* of $\mathbb{C}$ *after* x:

$$x \cdot \mathbb{C} \coloneqq \langle E, \{z \in \mathcal{P}(E) \mid \exists y \in \uparrow^{\mathbb{C}}\{x\} : z = y\backslash x\}\rangle$$

where $y\backslash x \coloneqq \{a \in y \mid a \notin x\}$ is the classical set difference.

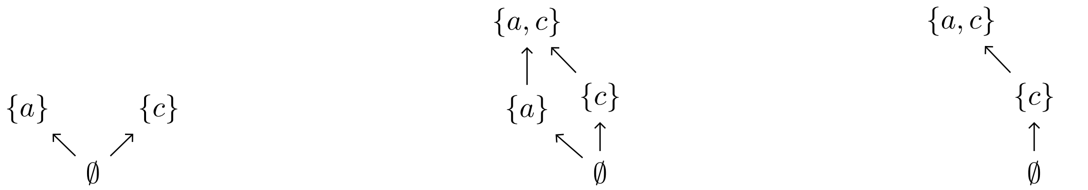
Since $(\mathcal{P}(E), \cup)$ is a commutative monoid, residuation can be understood as the action of a (partial) monoid on itself:

Proposition 2.4 (Monoid action) *The operator* $(\cdot) : \mathcal{P}_{\mathit{fin}}(E) \times \mathcal{C}_E \to \mathcal{C}_E$ *is a monoid action on configuration structures, i.e.,:*

- *for all configurations* x, y, *if* $y \in \mathbb{C}$ *and* $x \in y \cdot \mathbb{C}$, *then* $x \cdot (y \cdot \mathbb{C}) = (x \cup y) \cdot \mathbb{C}$.
- $\emptyset \cdot \mathbb{C} = \mathbb{C}$.

Proof. Each configuration $z \in x \cdot (y \cdot \mathbb{C})$ is of the form $z = (z'\backslash y)\backslash x = z'\backslash(x \cup y)$ for some $z' \in \mathbb{C}$. So $z \in x \cdot (y \cdot \mathbb{C})$ is equivalent to $z \in (x \cup y) \cdot \mathbb{C}$. The fact that $\emptyset \cdot \mathbb{C} = \mathbb{C}$ is a direct consequence of the definition of the residuation operation. □

Example 2.5 We give below the residuals of the configurations of Example 2.2 after $\{b\}$. Notice that each configuration residual can still trigger the events a and c, but they are incompatible in $\{b\} \cdot \mathbb{C}_0$, independent in $\{b\} \cdot \mathbb{C}_1$ and sequential in $\{b\} \cdot \mathbb{C}_2$.



$\{b\} \cdot \mathbb{C}_0$ $\{b\} \cdot \mathbb{C}_1$ $\{b\} \cdot \mathbb{C}_2$

For a Configuration structure $\mathbb{C} \coloneqq (E, X)$, consider $\mathcal{R}(\mathbb{C}) \coloneqq \{\mathbb{C}' \in \mathcal{C}_E \mid \exists x \in \mathbb{C} : \mathbb{C}' = x \cdot \mathbb{C}\}$ the *reachable* residuals of $\mathbb{C}$. Given an *initial* configuration structure $\mathbb{C}_0$, one can build the *labelled transition system* $\mathcal{T} \coloneqq (\mathbb{C}_0, \to, \mathcal{R}(\mathbb{C}_0))$ where $\to \subseteq \mathcal{R}(\mathbb{C}_0) \times \mathcal{P}(E) \times \mathcal{R}(\mathbb{C}_0)$ is the transition relation defined as $\mathbb{C} \to_x \mathbb{C}'$ if and only if $\mathbb{C}' = x \cdot \mathbb{C}$. This models program states as configuration structures $\mathbb{C}_0, \mathbb{C}_1, \dots$ and computations as sequences of transition between them. Notice that since $\to$ is engendered by a monoid action, the transition relation is closed by composition (see Proposition 2.4). It is also equipped with causal and conflict informations (which is sometimes called a truly concurrent semantics) since the transitions $\mathbb{C} \to_x \mathbb{C}'$ and $\mathbb{C} \to_y \mathbb{C}''$ are concurrent if $x \frown_{\mathbb{C}} y$ and conflicting otherwise.

3 Symmetric residuation

3.1 Building up intuitions

To model reversible computation we wish to keep interpreting program states as regular configurations structures: sets of events ordered by inclusion. It would be tempting to consider introducing negative events as first class citizens, but this would force one to consider configurations modulo some equivalence $\{a, a^-\} \sim \emptyset$, which would entail unnecessary complications as set union is no longer associative under this equivalence (think of $(\{a\} \cup \{a\}) \cup \{a^-\}$ and $\{a\} \cup (\{a\} \cup \{a^-\})$).

The alternative is to work with a different residuation operation which yields a group action instead of a monoid one. To illustrate this, consider a system that can perform a or b, but not both, and then terminates. This is modeled by the configuration structure $\mathbb{C} = \langle \{a, b\}, \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \rangle$. Classical residuation yields $\mathbb{C} \to_{\{a\}} \langle \{a, b\}, \{\emptyset\} \rangle$, as the event b is no longer visible in the future of $a \cdot \mathbb{C}$. Making this computations reversible requires a memory or special markers [19] in order to keep track of past events. Following this intuition, we will introduce a reversible residuation operator, which takes care of producing the "memory" of having consumed a and produce transitions of the form:

$$\langle \{a, b\}, \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \rangle \to_{\{a\}} \langle \{a, b\}, \{\emptyset, \{\underline{a}\}, \{\underline{a}, b\}\} \rangle$$

whose target state reads as "one can trigger event a (from the memory), and then one can trigger b". We will see that this transition can be derived using a symmetric version of the classical residuation, which uses symmetric set difference instead of set difference (Definition 3.1), but before diving into formal definitions, we can make a few more comments on the configuration structure $\langle \{a, b\}, \{\emptyset, \{\underline{a}\}, \{\underline{a}, b\}\} \rangle$. First, the underlined events are pure annotations, in order to keep track of the fact that the configuration $\{\underline{a}\}$ contains a past event. As all occurrences of a are underlined, there is no possible clash between a "forward" a and a "backward" one. This annotation must be understood in a similar way as tracking a redex in the λ-calculus: useful for proving convergence, but having no impact on the semantics (see for instance Ref [38], Section 2.4). A consequence is that symmetric residuation on configuration structures will have a constant support: in our example $E = \{a, b\}$ throughout computations, although underlined events will vary. Another important aspect is that symmetric residuation is conservative: erasing configurations with an underlined component corresponds to performing classical residuation. Lastly, we will see that symmetric residuation is a group action on configuration structure, with events being their own inverse. In our example, we will have:

$$\langle \{a, b\}, \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \rangle \to_{\{a\}} \langle \{a, b\}, \{\emptyset, \{\underline{a}\}, \{\underline{a}, b\}\} \rangle \to_{\{\underline{a}\}} \langle \{a, b\}, \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \rangle$$

which comes back to the initial structure.

3.2 Formally

In order to make these intuitions formal, we first introduce an operation of *symmetric residuation*, a generalization of the residuation of Definition 2.3, which takes care of producing the causal structure of reversible computations. We introduce explicit backward (the underlined events of our example in the previous section) and forward events in Section 4.

Recall that $\triangle$, the symmetric set difference, is defined as $x \triangle y \coloneqq (x \cup y) \backslash (x \cap y)$.

Property 1 *Any power set equipped with the symmetric difference forms an abelian group.*

In particular $(\mathcal{P}(E), \triangle)$ has an abelian group structure, where every event is its own inverse. Its action on configuration structures generates orbits that provide the mathematical foundations on which we will model reversible computations.

Definition 3.1 Let $\mathbb{C} \in \mathcal{C}_E$. For all finite $x \in \mathbb{C}$, we define the *symmetric residual of* $\mathbb{C}$ *after* x:

$$x \odot \mathbb{C} \coloneqq \langle E, \{z \in \mathcal{P}(E) \mid \exists y \in \mathbb{C} : z = y \triangle x\}\rangle.$$

Proposition 3.2 (Group action) *The operator* $(\odot) : \mathcal{P}_{\mathit{fin}}(E) \times \mathcal{C}_E \to \mathcal{C}_E$ *is a* group action *on configuration structures, i.e.:*

- *for all finite configurations x, y, if $x \in \mathbb{C}$ and $y \in x \odot \mathbb{C}$, then $x \odot (y \odot \mathbb{C}) = (x \triangle y) \odot \mathbb{C}$.*
- *$\emptyset \odot \mathbb{C} = \mathbb{C}$.*

Proof. We need to show that $x \triangle y$ is a configuration of $\mathbb{C}$. If $x \odot (y \odot \mathbb{C})$ is defined, then y is a configuration of $\mathbb{C}$ and x is a configuration of $y \odot \mathbb{C}$. Now all configurations of $y \odot \mathbb{C}$ must be of the form $y \triangle z$ for some configuration z of $\mathbb{C}$. Therefore $x = y \triangle z$ and thus, $x \triangle y = (y \triangle z) \triangle y = z \triangle (y \triangle y) = z$ using Proposition 1. It entails that $x \triangle y$ is indeed a configuration of $\mathbb{C}$.

The fact that $\emptyset \odot \mathbb{C} = \mathbb{C}$ is a consequence of $\emptyset$ being the neutral element for the group $(\mathcal{P}(E), \triangle)$. ☐

Notice that symmetric residuation $x \odot \mathbb{C}$ acts uniformly on all the configurations of $\mathbb{C}$. This is in contrast with the classical residuation $x \cdot \mathbb{C}$ (Definition 2.3), which discards configurations of $\mathbb{C}$ that are not above x. Yet both symmetric and classical residuations coincide on configurations above x.

Proposition 3.3 (Conservative extension) *Let $\mathbb{C} \in \mathcal{C}_E$. For all finite $x \in \mathbb{C}$, if $z \in x \cdot \mathbb{C}$ then $z \in x \odot \mathbb{C}$.*

Proof. Each configuration z in $x \cdot \mathbb{C}$ is such that $\exists y$, $x \subseteq y$ and $z = y \backslash x$. Since $x \subseteq y$, $y \backslash x = (y \cup x) \backslash (x \cap y) = x \triangle y$ and hence $z = x \triangle y \in x \odot \mathbb{C}$. ☐

Definition 3.4 The *orbit* $\mathit{Orb}(\mathbb{C})$ of a configuration structure $\mathbb{C} \in \mathcal{C}_E$ is defined as:

$$\mathit{Orb}(\mathbb{C}) \coloneqq \{\mathbb{C}' \in \mathcal{C}_E \mid \mathbb{C}' = x \odot \mathbb{C} \text{ for some finite } x \in \mathbb{C}\}.$$

The associated equivalence relation $\sim_\odot \; \subseteq \mathcal{C}_E \times \mathcal{C}_E$ is: $\mathbb{C} \sim_\odot \mathbb{C}'$ if and only if $\mathbb{C}' \in \mathit{Orb}(\mathbb{C})$.

Example 3.5 The configuration structures of Example 2.2 are equivalent by symmetric residuation, i.e., $\mathbb{C}_0 \sim_\odot \mathbb{C}_1 \sim_\odot \mathbb{C}_2$, as pictured in Figure 1, with e.g., $\odot_{\{a\}} : y \mapsto \{a\} \triangle y$.

Symmetric residuation does not act freely on configuration structures, i.e., not all the group fixed points are trivial. Note, for instance, that the configuration structure $(E, \mathcal{P}(E))$ is invariant under the action of $\odot$.

Example 3.6 The configuration structure $\mathbb{C}$ below remains invariant after symmetric residuation by $\{b\}$ (greyed out). Dashed arrows denote the function $\odot_{\{b\}} : y \mapsto \{b\} \triangle y$.

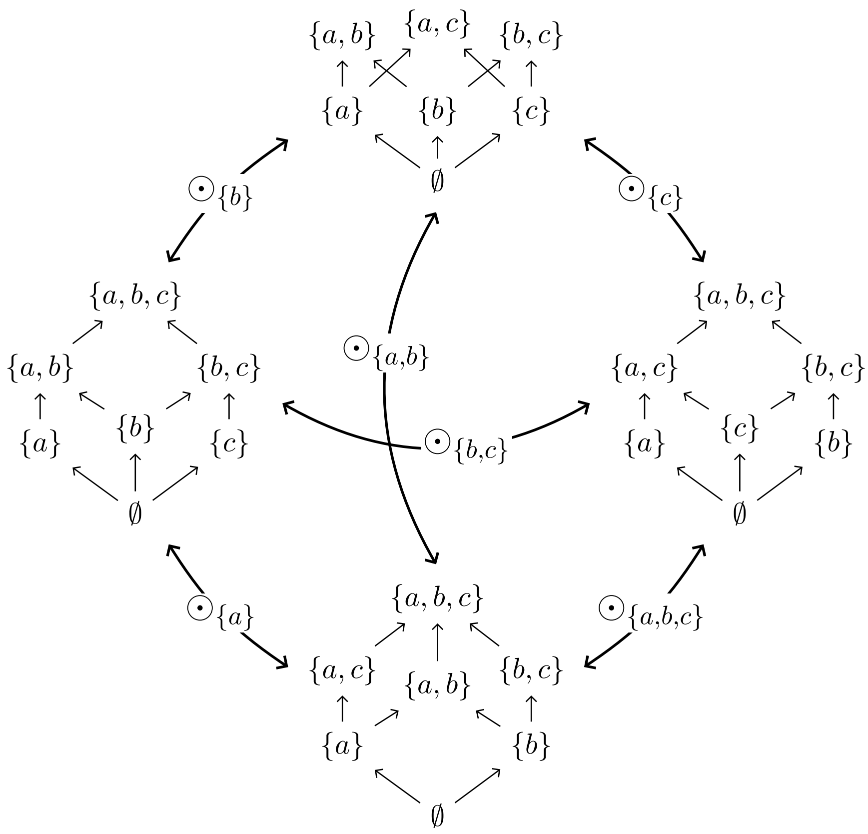
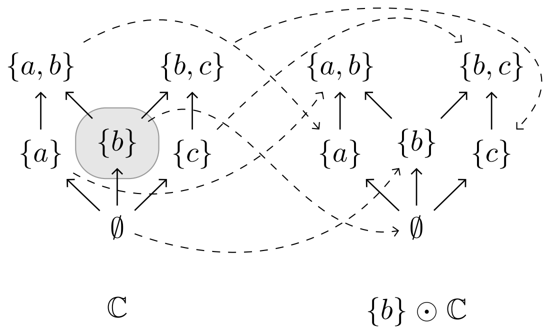


Fig. 1. Orbit of $\mathbb{C}_0$



4 Pointed configuration structures

Symmetric residuation yields the bare causal structure of reversible computations which is "direction insensible" in the sense that there is no way to tell whether an event is backward or forward in a computation. As a consequence, residuation by configurations that are neither enabling nor preventing any event from occurring are fixed points. This is, for instance, the case of configuration $\{b\}$ in Example 3.6.

Classical semantics of reversible computations distinguish backward and forward events [18] or transitions [28]. In operational semantics in particular, it may be desirable to observe the direction of a transition to prioritize "forward" executions, so that backtracking occurs only when necessary [20]. Debugging concur-

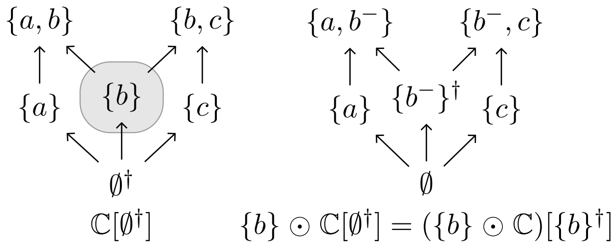


Fig. 2. Symmetric residuation applied to a pointed configuration structure. The negative signs label the events (here b) that belong to the referential configuration.

rent executions also has a natural orientation, where backtracking is possible only during trace analysis [23]. Bisimulations for reversible concurrent processes usually require matching forward and backward transitions with transitions of the same direction [16,30,34,3,24,4].

To comply with these semantics, we show here how symmetric residuation yields a natural notion of negative and positive configurations. Notice that adding a notion of direction of computation does not change the semantics we introduced so far, but rather annotates it with additional information. We do this by introducing *pointed* configuration structures, which are just plain configuration structures equipped with a distinguished configuration–called its *referential*–that points to the past. A similar idea was used to draw an operational correspondence between trajectories in a configuration structure and a reversible operator algebra using labels [3].

Definition 4.1 A *pointed configuration structure* $\mathbb{C}[x^\dagger]$ is a pair $\langle \mathbb{C}, x^\dagger \rangle$ where $\mathbb{C}$ is a configuration structure and $x^\dagger \in \mathbb{C}$ is a finite configuration, which we call its *referential*. A pointed configuration structure is *initial* when $x^\dagger = \emptyset$.

In the following, we simply write $\mathbb{C}[x^\dagger] \in \mathcal{C}_E$ if $\mathbb{C} \in \mathcal{C}_E$.

Definition 4.2 Let $\mathbb{C}[x^\dagger] \in \mathcal{C}_E$. For all finite $y \in \mathbb{C}$, we define $y \odot \mathbb{C}[x^\dagger] \coloneqq (y \odot \mathbb{C})[y \mathbin{\triangle} x^\dagger]$.

Symmetric residuation acts freely on pointed configuration structures, i.e., the orbit of a pointed configuration structure has as many elements as the structure has configurations.

Proposition 4.3 (Free action) *Let $\mathbb{C}[x^\dagger] \in \mathcal{C}_E$. For all finite $y, z \in \mathbb{C}$, $y \odot \mathbb{C}[x^\dagger] = z \odot \mathbb{C}[x^\dagger]$ if and only if $y = z$.*

Proof. The only if direction is trivial. For the if part, it suffices to remark that the referential in $y \odot \mathbb{C}[x^\dagger]$ must be equal to the referential in $z \odot \mathbb{C}[x^\dagger]$ and thus, $y \mathbin{\triangle} x^\dagger = z \mathbin{\triangle} x^\dagger$. Since $\odot_x : y \mapsto x \mathbin{\triangle} y$ is injective we can conclude that $y = z$. □

Definition 4.4 Let $\mathbb{C}[x^\dagger] \in \mathcal{C}_E$. For all $a \in E$, say that a is *negative in* $\mathbb{C}[x^\dagger]$ when $a \in x^\dagger$. It is *positive in* $\mathbb{C}[x^\dagger]$ otherwise. A configuration is *backward* (resp. *forward*) if all its events are *negative* (resp. *forward*).

Observe that referentials are finite configurations, and thus only finite configurations may be purely backwards. This echoes the *well-foundedness* axiom [25, Definition 3.1] requiring reversible systems to have a finite past.

Example 4.5 Once equipped with a referential, the configuration structure of Example 3.6 is no longer a fixed point, as illustrated in Figure 2, where negative signs denote negative events.

The referential of a pointed configuration structure provides a path back to its origin, which can be retrieved by performing the residuation $x^\dagger \odot \mathbb{C}[x^\dagger]$. In addition, Proposition 4.3 guarantees that this is the only way to go back to the initial structure.

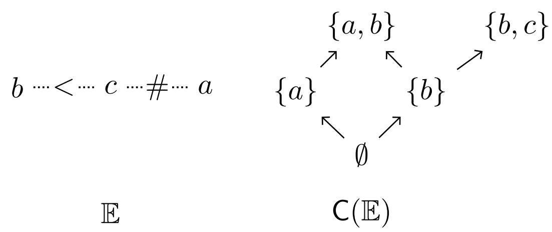


Fig. 3. An event structure and the set of its admissible configurations, forming a configuration structure.

5 Stable orbits and prime event structures

Configuration structures are *extensional* models for concurrency, which means that the partial order of configurations is *a priori* not reducible to relations on events themselves. For instance, in Example 2.2, the fact that $\{a, b\} \not\to_{\mathbb{C}_0} \{a, c\}$ is not a consequence of event b being in conflict with event c since $\{b, c\}$ is indeed a reachable configuration of $\mathbb{C}_0$. Prime event structures, which we introduce now, are models of concurrent systems whose set of admissible configurations is precisely engendered by a causality and a conflict relation on events. In particular, algebraic process calculi in the style of CCS [29], which are commonly used to model concurrent programs, can be interpreted as *prime event structures* [44,31,9,43].

Definition 5.1 A (prime) event structure is a tuple $\mathbb{E} = \langle E, <, \# \rangle$ where $< \subseteq E \times E$ is a partial order (e.g., transitive, anti-symmetric and reflexive) called the *causality* relation and $\# \subseteq E \times E$ an irreflexive symmetric *conflict* relation satisfying:

$$e \# e' < e'' \Rightarrow e \# e'' \qquad \text{(Principle of conflict heredity)}$$

The *configurations* of $\mathbb{E}$, written $\mathit{Conf}(\mathbb{E})$, are the subsets of E that are conflict-free and downward closed for the causality relation. Two prime event structures $(E, <, \#)$ and $(E', <', \#')$ are *isomorphic* if there is a bijection $\phi : E \to E'$ that preserves causality and conflict, i.e., $e < e'$ if and only if $\phi(e) <' \phi(e')$ and $e \# e'$ if and only if $\phi(e) \#' \phi(e')$.

A well-known result by Winskel [44] is that the configurations of a prime event structure coincide with *stable* configuration structures.

Definition 5.2 Let $\mathbb{C} \in \mathcal{C}_E$ be a configuration structure. Say that $\mathbb{C}$ is *stable* if it is:

Rooted: $\emptyset \in \mathbb{C}$

Connected: for all $x \in \mathbb{C}$, $x \neq \emptyset \Rightarrow \exists a \in x : x \backslash \{a\} \in \mathbb{C}$

Closed under bounded union: for all $x, y, z \in \mathbb{C}$, $x \cup y \subseteq z \Rightarrow x \cup y \in \mathbb{C}$

Closed under intersection: for all $x, y \in \mathbb{C}$, $x \cap y \in \mathbb{C}$

Coherent: for all $x, y, z \in \mathbb{C}$, if there exists $z', z'', z''' \in \mathbb{C}$ such that $x \cup y \subseteq z'$, $y \cup z \subseteq z''$ and $x \cup z \subseteq z'''$, then $x \cup y \cup z \in \mathbb{C}$.

For instance, in Example 2.2, none of the configurations are stable: $\mathbb{C}_0$ is missing $\{a, b, c\}$ to be coherent, $\mathbb{C}_1$ is missing $\{c\}$ to be closed under intersection, and $\mathbb{C}_2$ is missing $\{a, b\}$ to be closed under bounded union. As a consequence there are no prime event structures whose set of configurations is isomorphic to any of these structures, as indicated by the following theorem:

Theorem 5.3 (Winskel [44]) *Let $\mathbb{E} = \langle E, <, \# \rangle$ be a prime event structure. The configuration structure $\mathsf{C}(\mathbb{E}) \coloneqq \langle E, \mathit{Conf}(\mathbb{E}) \rangle$ is stable (see an illustration in Figure 3).*

In the following, we let $\mathcal{S}_E \subseteq \mathcal{C}_E$ denote stable configuration structures over E. Stable configuration structures are, in fact, concrete representations of *prime algebraic domains*, which we now introduce and which play an important role in connecting sets of configurations to event structures.

Definition 5.4 Let $\mathbb{P} = (X, \sqsubseteq)$ be a partial order. A set $Y \subseteq X$ is:

Consistent when $\bigsqcup^{\mathbb{P}} Y$ exists.

Pairwise consistent if $Y \neq \emptyset$ and all its pairs of distinct elements form a consistent set.

Directed when $Y \neq \emptyset$ and $\bigsqcup^{\mathbb{P}} Y \in Y$.

An element $x \in X$ is:

Compact if for every directed $Y \subseteq X$ such that $x \sqsubseteq \bigsqcup^{\mathbb{P}} Y$, there exists $y \in Y$ such that $x \sqsubseteq y$.

Complete prime if for every pairwise consistent $Y \subseteq X$ such that $x \sqsubseteq \bigsqcup^{\mathbb{P}} Y$, there exists $y \in Y$ such that $x \sqsubseteq y$.

Let $\mathsf{K}(\mathbb{P}) \subseteq X$ denote the *compact elements* of $\mathbb{P}$ and $\mathsf{Pr}(\mathbb{P}) \subseteq \mathsf{K}(\mathbb{P})$ the (complete) prime elements of $\mathbb{P}$. We use $p, q, \dots$ instead of $x, y, \dots$ to highlight prime elements. Let $\lfloor x \rfloor_{\mathbb{P}} \coloneqq \downarrow_{\mathbb{P}}\{x\} \cap \mathsf{Pr}(\mathbb{P})$ denote the complete prime elements that are below x in $\mathbb{P}$.

Definition 5.5 Let $\mathbb{P} = (X, \sqsubseteq)$ be a partial order. It is:

Finitary whenever for all $x \in \mathsf{K}(\mathbb{P})$, $\downarrow_{\mathbb{P}}\{x\}$ is finite.

Coherent whenever for all pairwise consistent set $Y \subseteq X$, $\bigsqcup^{\mathbb{P}} Y$ exists.

Prime algebraic if for all $x \in \mathbb{P}$, $x = \bigsqcup^{\mathbb{P}} \lfloor x \rfloor_{\mathbb{P}}$.

We call *(prime algebraic) domains*, the partial orders that are finitary, coherent and prime algebraic.

Theorem 5.6 (Winskel [44]) *If a configuration structure $\mathbb{C} = (E, X)$ is stable then $\mathbb{P} = (X, \subseteq)$ is a domain of configurations.*

Theorem 5.7 (Winskel [44]) *Let $\mathbb{C} \in \mathcal{S}_E$. Then $\mathsf{E}(\mathbb{C}) \coloneqq \langle \mathsf{Pr}(\mathbb{C}), <, \# \rangle$ where $p < q$ if $p \subseteq q$ and $p \# q$ if $p \not\uparrow_{\mathbb{C}} q$ is a prime event structure.*

A fundamental result is that combining the functors C and E of Theorems 5.3 and 5.7 actually defines an adjunction, which is also an equivalence between stable configurations and prime event structures [44]: for all stable $\mathbb{C}$ and all prime event structure $\mathbb{E}$: $\mathsf{C}(\mathsf{E}(\mathbb{C})) \sim \mathbb{C}$ and $\mathsf{E}(\mathsf{C}(\mathbb{E})) \sim \mathbb{E}$. We can utilize this equivalence to transport symmetric residuation to prime event structures, but in order to do this, we must first verify that symmetric residuation preserves stability, i.e., the properties of Definition 5.2. Observe that the orbit presented in Example 3.5 does not preserve prime algebraicity since the topmost configuration structure is prime algebraic (but not coherent), when the other elements of the orbit are not.

Theorem 5.8 (Stable orbits) *The group action $\odot$ preserves stability, and is therefore a group action on stable configuration structures.*

This property serves as a cornerstone result, enabling the definition of an event-structure semantics for reversible computations. This result, combined with the adjunction that makes prime event structures equivalent to stable configuration structures, allows one to conclude that symmetric residuation operates on prime event structures as well. Section 7 is devoted to tracking the effect of symmetric residuation on prime elements. In the meantime, we show that Winskel's construction on pointed stable configuration structures yields prime event structures that are equipped with a polarity map attributing a sign to events.

6 Polarized event structures

The construction of Theorem 5.7 maps configuration structures to prime event structures using their complete prime elements. We extend this construction to build *polarized event structures* out of pointed configuration structures. Polarized event structures are just plain prime event structures, in which events have a sign which characterizes whether they have a forward or backward contribution to the configurations.

Definition 6.1 A *polarized event structure* $\mathbb{E}[\pi]$ is a pair $\langle \mathbb{E}, \pi \rangle$ where $\mathbb{E} = (E, <, \#)$ is an event structure and $\pi : E \to \{-1, 1\}$, the *polarity of* $\mathbb{E}[\pi]$, satisfies $\mathit{Neg}(\mathbb{E}[\pi]) \coloneqq \{e \in E \mid \pi(e) < 0\} \in \mathit{Conf}(\mathbb{E})$. Two polarized event structures $\mathbb{E}[\pi]$ and $\mathbb{E}'[\pi']$ are isomorphic if $\mathbb{E}$ and $\mathbb{E}'$ are isomorphic, and the induced bijection $\phi : E \to E'$ satisfies $\pi(e) = \pi'(\phi(e))$.

That $Neg(\mathbb{E}[\pi])$ is a configuration of $\mathbb{E}[\pi]$ guarantees that past events have indeed been produced by a forward computation, a standard requirement in reversible semantics [25, Definition 3.1].

Proposition 6.2 *Let $\mathbb{C}[x^\dagger]$ be a pointed stable configuration structure and define $\mathsf{E}(\mathbb{C}[x^\dagger]) := \mathsf{E}(\mathbb{C})[\pi_{x^\dagger}]$ where for all $p \in \mathsf{Pr}(\mathbb{C})$, $\pi_{x^\dagger}(p) < 0$ if $p \subseteq x^\dagger$ and $\pi_{x^\dagger}(p) > 0$ otherwise. Then $\mathsf{E}(\mathbb{C}[x^\dagger])$ is a polarized event structure.*

Proof. Suffices to show that $N := Neg(\mathsf{E}(\mathbb{C})[\pi_{x^\dagger}])$ is a configuration of $\mathsf{E}(\mathbb{C})$. We have $N = \{p \in \mathsf{Pr}(\mathbb{C}) \mid p \subseteq x^\dagger\}$. Thus all the events of N have an upper bound in $\mathbb{C}$ and are therefore not in conflict. N is also downwards $<$-closed by construction and is therefore a configuration of $\mathsf{E}(\mathbb{C})$. ◻

7 Mapping prime elements through symmetric residuation

Symmetric residuation describes how a new configuration structure $x \odot \mathbb{C}$ is obtained from a configuration structure $\mathbb{C}$, and can take the form of a map $\odot_x : \mathbb{C} \to x \odot \mathbb{C}$ where, for all $y \in \mathbb{C}$, $\odot_x(y) = x \,\triangle\, y$ (see an illustration on Example 3.6). However, this does not map prime elements of its source to prime elements of its targets, take for instance $\odot_{\{b\}}(\{a\}) = \{a, b\}$ in Example 3.6. This would lead to a complex definition of the action of symmetric residuation on prime event structures, where events would be erased and created during computations.

For all stable configuration structure $\mathbb{C}$ and all finite configuration $x \in \mathbb{C}$, we establish now the existence of a bijective map $\sigma_x : \mathsf{Pr}(\mathbb{C}) \to \mathsf{Pr}(x \odot \mathbb{C})$ (Definition 7.6) which allows us to show that prime elements are indeed preserved by symmetric residuation, although their causal relationship is transformed (Theorem 7.9). We first need to state some useful properties of prime elements in stable configuration structures.

In prime algebraic partial orders, complete primes have a notoriously simpler characterization in terms of the unicity of their immediate predecessor (see Baldan *et al.* [5, Lemma 13]).

Proposition 7.1 (Unique predecessor) *Given a prime algebraic partial order $\mathbb{P}$, an element $x \in \mathbb{P}$ is a complete prime if and only if it has a unique immediate predecessor, i.e., there exists $y \in \mathbb{P}$ such that*

$$pred_{\mathbb{P}}(x) := max_{\mathbb{P}}(\downarrow_{\mathbb{P}}\{x\} \backslash \{x\}) = \{y\}$$

Proof. Suppose by contradiction that $p \in \mathbb{P}$ is prime but that it has two different immediate predecessors y and $y' \in \mathbb{P}$. Then, $p \sqsubseteq \bigsqcup\{y, y'\}$ and yet there is no $z \in \{y, y'\}$ such that $p \sqsubseteq z$, contradicting the prime completeness of p.

Conversely, suppose $y \in X$ has a unique immediate predecessor y'. Since $\mathbb{P}$ is prime algebraic we have $y = \bigsqcup^{\mathbb{P}} \lfloor y \rfloor_{\mathbb{P}}$. By contradiction, suppose that $y \notin \mathsf{Pr}(\mathbb{P})$, we would have $\bigsqcup^{\mathbb{P}} \lfloor y \rfloor_{\mathbb{P}} = \bigsqcup^{\mathbb{P}} \lfloor y' \rfloor_{\mathbb{P}} = y'$. This would contradict $y' \subsetneq y$ and therefore y must be a complete prime. ◻

Definition 7.2 Let $\mathbb{C} \in \mathcal{S}_E$ be stable configuration structure and $p \in \mathsf{Pr}(\mathbb{C})$. We define the *derivative* of p in $\mathbb{C}$ as $\delta_{\mathbb{C}}(p) := a$ whenever $p \backslash pred_{\mathbb{C}}(p) = \{a\}$.

Proposition 7.3 (Event introduction) *Let $\mathbb{C} \in \mathcal{S}_E$. For all $a \in E$ and $x \in \mathbb{C}$, $a \in x$ if only if there exists $p \in \mathsf{Pr}(\mathbb{C})$ such that $\delta_{\mathbb{C}}(p) = a$ and $p \subseteq x$. We say that p* introduces *a in $\mathbb{C}$.*

Proof. The only if part is trivial since $\delta_{\mathbb{C}}(p)$ implies $a \in p$ and together with $p \subseteq x$ it implies $a \in x$. For the other part, let $x \in \mathbb{C}$ such that $a \in x$. Since $\mathbb{C}$ is prime algebraic, $x = \bigsqcup^{\mathbb{C}} \lfloor x \rfloor_{\mathbb{C}}$ which is also the union of all prime elements below x (by stability of $\mathbb{C}$). This implies that the set $P_a := \{p \in \lfloor x \rfloor_{\mathbb{C}} \mid a \in p\}$ is not empty. Take the minimal elements $min(P_a)$ of P_a and suppose, by contradiction that it is not a singleton. We would have $p \in min(P_a)$ and $q \in min(P_a)$ with $a \in p \cap q =: y$ and $p \neq q$. Now y is a configuration of $\mathbb{C}$ (by stability) which is either prime and contains a or has prime containing a below it. This would contradict the minimality of $min(P_a)$. Let $min(P_a) = \{p\}$. By Proposition 7.1, p has a single immediate predecessor z such that $a \notin z$ and $\delta_{\mathbb{C}}(p) = a$. ◻

Proposition 7.4 (Unicity of event introducer) *Let $\mathbb{C} \in \mathcal{S}_E$. For all $a \in E$ such that $a \in x$ for some $x \in \mathbb{C}$, there is a unique $p \in \mathsf{Pr}(\mathbb{C})$ that introduces a.*

Proof. This is a consequence of $\mathbb{C}$ being closed under intersection: by contradiction, suppose there are two prime elements p and q that introduce a. We would have $a \in p \cap q$ and $p \cap q \subseteq p$, which would contradict $a \notin pred_{\mathbb{C}}(p)$ (since $p \backslash pred_{\mathbb{C}}(p) = \{a\}$). □

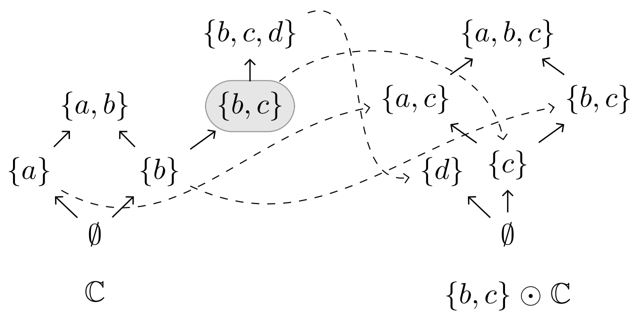
Corollary 7.5 (to Proposition 7.4) *Let $\mathbb{C} \in \mathcal{S}_E$. For all finite $x \in \mathbb{C}$ and $p \in \mathsf{Pr}(\mathbb{C})$, there is a unique $q \in \mathsf{Pr}(x \odot \mathbb{C})$ such that $\delta_{\mathbb{C}}(p) = \delta_{x \odot \mathbb{C}}(q)$.*

Proof. Using the fact that symmetric residuation does not introduce nor remove events and preserves stability (Theorem 5.8). □

Thanks to the above corollary, we can associate prime elements of $\mathbb{C}$ to prime elements of the symmetric residual of $\mathbb{C}$ after a finite configuration:

Definition 7.6 Let $\mathbb{C} \in \mathcal{S}_E$. For all finite $x \in \mathbb{C}$, define the *(symmetric) residuation map* $\sigma_x : \mathsf{Pr}(\mathbb{C}) \to \mathsf{Pr}(x \odot \mathbb{C})$ as $\sigma_x(p) = q$ if and only if $\delta_{\mathbb{C}}(p) = \delta_{x \odot \mathbb{C}}(q)$.

Example 7.7 Dashed arrows denote the residuation map $\sigma_{\{b,c\}} : \mathsf{Pr}(\mathbb{C}) \to \mathsf{Pr}(\{b,c\} \odot \mathbb{C})$.



Definition 7.8 We write $x \perp_{\mathbb{P}} y$ when two elements $x, y \in \mathbb{P}$ are *orthogonal*, i.e., compatible and incomparable: if $x \frown_{\mathbb{P}} y$ and $x \nsubseteq y$, $y \nsubseteq x$.

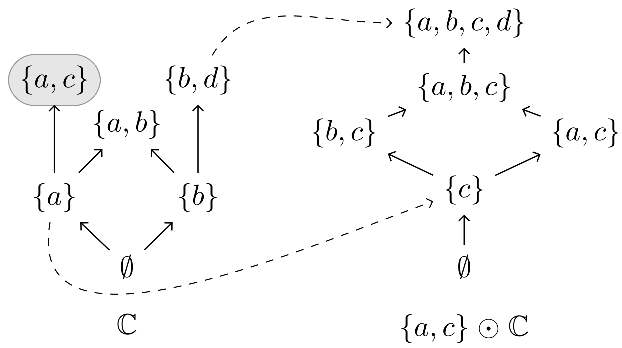
We can now characterize algebraically the effect of symmetric residuation on prime elements.

Theorem 7.9 (Effects of residuation on causal structure) *Let $\mathbb{C} \in \mathcal{S}_E$. For all finite $x \in \mathbb{C}$, let $\sigma_x : \mathsf{Pr}(\mathbb{C}) \to \mathsf{Pr}(x \odot \mathbb{C})$ be the residuation map of Definition 7.6. For all distinct $p, q \in \mathsf{Pr}(\mathbb{C})$, the following holds:*

$$
\begin{array}{llr}
\textit{If } p \perp_{\mathbb{C}} q \textit{ then} & \sigma_x(p) \perp_{x \odot \mathbb{C}} \sigma_x(q) & (1) \\
\textit{If } p \subseteq q \textit{ then} & q \subseteq x \implies \sigma_x(q) \subseteq \sigma_x(p) & (2) \\
\textit{If } p \subseteq q \textit{ then} & p \nsubseteq x \wedge q \nsubseteq x \implies \sigma_x(p) \subseteq \sigma_x(q) & (3) \\
\textit{If } p \subseteq q \textit{ then} & p \subseteq x \wedge q \nsubseteq x \implies \sigma_x(p) \not\frown_{x \odot \mathbb{C}} \sigma_x(q) & (4) \\
\textit{If } p \not\frown_{\mathbb{C}} q \textit{ then} & p \subseteq x \implies \sigma_x(p) \subseteq \sigma_x(q) & (5) \\
\textit{If } p \not\frown_{\mathbb{C}} q \textit{ then} & p \nsubseteq x \wedge q \nsubseteq x \implies \sigma_x(p) \not\frown_{x \odot \mathbb{C}} \sigma_x(q) & (6)
\end{array}
$$

Notice that for all $p', q' \in \mathsf{Pr}(\mathbb{C})$ exactly one of the above premise holds with either $p = p'$ and $q = q'$ or $p = q'$ and $q = p'$.

Example 7.10 We illustrate (5) below, taking $p = \{a\}$, $q = \{b, d\}$ and $x = \{a, c\}$, and representing the action of $\sigma_{\{a,c\}}$ on $\{a\}$ and $\{b, d\}$ with dashed lines:



We have $\{a\} \not\frown_{\mathbb{C}} \{b, d\}$, $\{a\} \subseteq \{a, c\}$ and indeed, $\sigma_x(\{a\}) = \{c\} \subseteq \sigma_{\{a,c\}}(\{b, d\}) = \{a, b, c, d\}$.

Proof. In the following we use the notation $x : a$ for $a \in x$, $x : \bar{b}$ for $b \notin x$ and use concatenations like $x : a\bar{b}$, for conjunction of these properties. Suppose p introduces a and q introduces b in $\mathbb{C}$. We verify (1) – (6) by case analysis.

(1) We must check that $p \perp_{\mathbb{C}} q$ implies $\sigma_x(p) \perp_{x\odot\mathbb{C}} \sigma_x(q)$. Since $p \perp_{\mathbb{C}} q$ by hypothesis, we have $p : a\bar{b}$ and $q : a\bar{b}$, otherwise by Proposition 7.3 we would have either $p \subseteq q$ or $q \subseteq p$ and they would not be orthogonal. We have three sub-cases to consider.

(1).1 Suppose $x : ab$. By Proposition 7.3 we have $p \subseteq x$ and $q \subseteq x$. It entails that $p \vartriangle x = x \backslash p$ and $q \vartriangle x = x \backslash q$ are configurations of $x \odot \mathbb{C}$. Since $p : a\bar{b}$, and $x : ab$, we have $x \backslash p : \bar{a}b$. By the same reasoning we have $x \backslash q : a\bar{b}$. Now, $x \backslash p$ and $x \backslash q$ are below x and incomparable. Since $x \backslash p : b$ we have $\sigma_x(q) \subseteq x \backslash p$ (by Proposition 7.3) and $\sigma_x(p) \subseteq x \backslash q$ by the same argument. It follows that $\sigma_x(p) \perp_{x\odot\mathbb{C}} \sigma_x(q)$.

(1).2 Suppose $x : \bar{a}b$ (the argument for $x : a\bar{b}$ is symmetrical). We have $p \cap x \in \mathbb{C}$ (by stability of $\mathbb{C}$) and $p \cup q \in \mathbb{C}$ (since $p \perp_{\mathbb{C}} q$). It entails that $x \vartriangle (p \cap x) = x \backslash p \in x \odot \mathbb{C}$ and $x \vartriangle (p \cup q) \in x \odot \mathbb{C}$. Now $x : \bar{a}b$ and $p : a\bar{b}$ implies $x \backslash p : \bar{a}b$. Also, $p \cup q : ab$ implies $x \vartriangle (p \cup q) : a\bar{b}$. It entails that $\sigma_x(q) \subseteq x \backslash p$ and $\sigma_x(p) \subseteq x \vartriangle (p \cup q)$ (by Proposition 7.3). Additionally $x : \bar{a}b$ and $p : a\bar{b}$ implies $x \vartriangle p : ab$ and is a configuration of $x \odot \mathbb{C}$. It follows that $\sigma_x(p) \subseteq x \vartriangle p$ and $\sigma_x(q) \subseteq x \vartriangle p$, so they are compatible. We prove they are incomparable by contradiction. Suppose $\sigma_x(p) \subseteq \sigma_x(q)$. This would entail $\sigma_x(p) \subseteq x \backslash p$ which would contradict $x \backslash p : \bar{a}$ (by Proposition 7.3). Still by contraction suppose $\sigma_x(q) \subseteq \sigma_x(p)$. This would entail $\sigma_x(q) \subseteq x \vartriangle (p \cup q)$ which would contradict $x \vartriangle (p \cup q) : \bar{b}$. We therefore have $\sigma_x(p) \perp_{x\odot\mathbb{C}} \sigma_x(q)$.

(1).3 Suppose $x : \bar{a}\bar{b}$. We have $x \vartriangle (p \cup q) : ab$, $x \vartriangle p : a\bar{b}$ and $x \vartriangle q : \bar{a}b$ are all configurations of $x \odot \mathbb{C}$. As a consequence $\sigma_x(p) \subseteq x \vartriangle p$, $\sigma_x(q) \subseteq x \vartriangle q$, $\sigma_x(p) \subseteq x \vartriangle (p \cup q)$ and $\sigma_x(q) \subseteq x \vartriangle (p \cup q)$. Therefore $\sigma_x(p)$ and $\sigma_x(q)$ are compatible. We prove they are incomparable by contradiction. Suppose $\sigma_x(p) \subseteq \sigma_x(q)$. It would entail that $\sigma_x(p) \subseteq x \vartriangle q$ which would contradict $x \vartriangle q : \bar{a}$. Still by contradiction suppose $\sigma_x(q) \subseteq \sigma_x(p)$. This would imply that $\sigma_x(q) \subseteq x \vartriangle p$ which would contradict $x \vartriangle p : \bar{b}$. Therefore $\sigma_x(p) \perp_{x\odot\mathbb{C}} \sigma_x(q)$.

(2) We verify $p \subseteq q$ and $q \subseteq x$ implies $\sigma_x(q) \subseteq \sigma_x(p)$. Combining hypothesis and Proposition 7.3, we have $p : a\bar{b}$, $q : ab$ and $x : ab$. By contradiction, we show that $\sigma_x(q) \subseteq \sigma_x(p)$. Suppose not, in which case we have $\sigma_x(p) : a\bar{b}$. So we have that $x \vartriangle \sigma_x(p) : \bar{a}b$ is a configuration of $\mathbb{C}$ in which b appears without a. This violates the fact that $q : ab$ introduces b in $\mathbb{C}$ and contains a. Therefore $\sigma_x(q) \subseteq \sigma_x(p)$.

(3) We verify $p \subseteq q$ and $p \not\subseteq x \wedge q \not\subseteq x$ implies $\sigma_x(p) \subseteq \sigma_x(q)$. Again by contradiction, suppose $\sigma_x(p) \not\subseteq \sigma_x(q)$. We would have $\sigma_x(q) : \bar{a}b$ and there would be a configuration $x \vartriangle \sigma_x(q) : \bar{a}b$ in $\mathbb{C}$, since by hypothesis $x : \bar{a}\bar{b}$. This would contradict the fact that any configuration containing b must be above $q : ab$ and hence should also contain a. Therefore $\sigma_x(p) \subseteq \sigma_x(q)$.

(4) We verify $p \subseteq q$, $p \subseteq x$ and $q \not\subseteq x$ implies $\sigma_x(p) \not\frown_{x\odot\mathbb{C}} \sigma_x(q)$. By contradiction, suppose there is a configuration $y \in x \odot \mathbb{C}$ such that $\sigma_x(p) \subseteq y$ and $\sigma_x(q) \subseteq y$. We would have $y : ab$ and there would be a configuration $x \vartriangle y \in \mathbb{C}$. Now by hypothesis we have $x : a\bar{b}$, therefore $x \vartriangle y : \bar{a}b$. This would contradict the hypothesis $q : ab$ and therefore $\sigma_x(p) \not\frown_{x\odot\mathbb{C}} \sigma_x(q)$.

(5) We verify $p \not\frown_{\mathbb{C}} q$ and $p \subseteq x$ implies $\sigma_x(p) \subseteq \sigma_x(q)$. By contradiction, suppose that $\sigma_x(p) \not\subseteq \sigma_x(q)$. We would have $\sigma_x(q) : \bar{a}b$ and $x \vartriangle \sigma_x(q) : ab$ would be a configuration of $\mathbb{C}$. This would contradict the hypothesis $p \not\frown_{\mathbb{C}} q$ and therefore $\sigma_x(p) \subseteq \sigma_x(q)$.

(6) We verify $p \not\frown_{\mathbb{C}} q$ and $p \not\subseteq x$ and $q \not\subseteq x$ implies $\sigma_x(p) \not\frown_{x \odot \mathbb{C}} \sigma_x(q)$. By contradiction, suppose there is a configuration $y \in x \odot \mathbb{C}$ such that $\sigma_x(p) \subseteq y$ and $\sigma_x(q) \subseteq y$. We would have $y : ab$ and $x \vartriangle y : ab$ would be a configuration of $\mathbb{C}$. This contradicts the fact that p and q are incompatible. Therefore, $\sigma_x(p) \not\frown_{x \odot \mathbb{C}} \sigma_x(q)$. □

8 A switch operation on prime event structures

Now that the effects of symmetric residuation are fully characterized, we proceed with the definition of symmetric residuation on prime event structures.

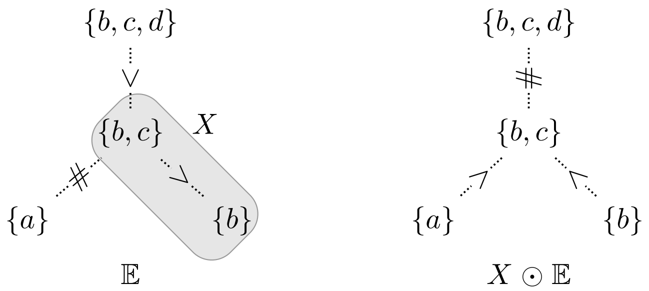
Definition 8.1 Let $\mathbb{E} = (E, <, \#)$ be a prime event structure. For all finite configuration $X \in \mathit{Conf}(\mathbb{E})$ we define $X \odot \mathbb{E} \coloneqq (E, <', \#')$ as:

$$a <' b \text{ if and only if } \begin{cases} a < b \text{ and } \{a,b\} \cap X = \emptyset \\ b < a \text{ and } \{a,b\} \subseteq X \\ a \# b \text{ and } a \in X \end{cases} \qquad a \#' b \quad \text{if and only if } \begin{cases} a \# b \text{ and } \{a,b\} \cap x = \emptyset \\ a < b, a \in X \text{ and } b \notin X \end{cases}$$

where clauses on the right of the curly brackets are taken disjunctively.

Note that the definition above does not imply that $X \odot \mathbb{E}$ is a prime event structure, as the $<'$ and $\#'$ might not satisfy the requirements listed in Definition 5.1. Theorem Theorem 8.3 will take care of both by ensuring that $X \odot \mathbb{E}$ is indeed a group action on prime event structure, and that it properly reflects the effect of symmetric residuation of the underlying configuration structure.

Example 8.2 A prime event structure $\mathbb{E}$ with its configuration $X = \{\{b\}, \{b,c\}\}$ grayed out, and $X \odot \mathbb{E}$:



Switching an event structure is relative to a finite configuration X and is reminiscent of the Seidel switch, a well known graph isomorphism which consists in fixing a set of vertices X and flipping edges and non-edges between X and its complement in the graph (the other edges remaining invariant) [37]. We discuss potential interesting connections in the conclusion of this paper.

For all configuration structures $\mathbb{C}$, recall that $\mathsf{E}(\mathbb{C})$ is Winskel's construction, mapping sets of configurations to a prime event structure, via the prime elements (see Theorem 5.7). The following theorem, where we borrow from Definition 5.4 that for all $x \in \mathbb{C}$, $\lfloor x \rfloor_{\mathbb{C}} \coloneqq \downarrow_{\mathbb{C}}\{x\} \cap \mathsf{Pr}(\mathbb{C})$, characterizes symmetric residual as a switch operation on prime event structures.

Theorem 8.3 (Adequacy) *Let $\mathbb{C} \in \mathcal{S}_E$ be a stable configuration structure. For all finite $x \in \mathbb{C}$, the prime event structure $\mathsf{E}(x \odot \mathbb{C})$ is isomorphic to $\lfloor x \rfloor_{\mathbb{C}} \odot \mathsf{E}(\mathbb{C})$.*

Proof. We show that the isomorphism of event structures is given by the bijection σ_x of Definition 7.6.

Consider

$$\begin{aligned}\mathbb{E}_0 &:= \mathsf{E}(\mathbb{C}) &&= (\mathsf{Pr}(\mathbb{C}), <_0, \#_0)\\ \mathbb{E}_1 &:= \lfloor x \rfloor_{\mathbb{C}} \odot \mathbb{E}_0 &&= (\mathsf{Pr}(\mathbb{C}), <_1, \#_1)\\ \mathbb{E}_2 &:= \mathsf{E}(x \odot \mathbb{C}) &&= (\mathsf{Pr}(x \odot \mathbb{C}), <_2, \#_2)\end{aligned}$$

We overload notations and write $a \perp b$ for two events in $(E, <, \#)$ whenever $a \not< b$, $b \not< a$ and $a \# b$. Since σ_x^{-1} is the residuation map of $x \odot (x \odot \mathbb{C}) = \mathbb{C}$ the proof is symmetric and we just need to show $p \mathcal{R}_1 q$ implies $\sigma_x(p) \mathcal{R}_2 \sigma_x(q)$, for $\mathcal{R}_i \in \{\perp_i, <_i, \#_i\}$:

$p \perp_1 q$ implies $\sigma_x(p) \perp_2 \sigma_x(q)$. We have $p \perp_1 q$ iff $p \perp_0 q$, by definition of the switch operation. Now $p \perp_0 q$ implies $\sigma_x(p) \perp_0 \sigma_x(q)$ by Theorem 7.9 (1).

$p <_1 q$ implies $\sigma_x(p) <_2 \sigma_x(q)$. There are three sub-cases according to Definition 8.1:

- $p <_0 q$ and $\{p, q\} \cap \lfloor x \rfloor_{\mathbb{C}} = \emptyset$. We have $p <_0 q$ implies $p \subseteq q$ by Theorem 5.7. Since $\mathbb{C}$ is prime algebraic, $\bigsqcup_{\mathbb{C}} \lfloor x \rfloor_{\mathbb{C}} = x$ and we have $p \not\subseteq x$ and $q \not\subseteq x$. By Theorem 7.9 (3), this implies $\sigma_x(p) \subseteq \sigma_x(q)$ and hence $\sigma_x(p) <_2 \sigma_x(q)$.
- $q <_0 p$ and $\{p, q\} \subseteq \lfloor x \rfloor_{\mathbb{C}}$. We have $q <_0 p$ implies $q \subseteq p$ by Theorem 5.7. Since $\mathbb{C}$ is prime algebraic, $\bigsqcup_{\mathbb{C}} \lfloor x \rfloor_{\mathbb{C}} = x$ and we have $p \subseteq x$ and $q \subseteq x$. By Theorem 7.9 (2), we have $\sigma_x(p) \subseteq \sigma_x(q)$ which entails $\sigma_x(p) <_2 \sigma_x(q)$ by definition of $<_2$ (see Theorem 5.7).
- $p \#_0 q$ and $p \in \lfloor x \rfloor_{\mathbb{C}}$. Using a similar argument as above, we deduce $p \subseteq x$. Additionally since $p \not\uparrow_{\mathbb{E}_0} q$, we can deduce $q \not\subseteq x$, otherwise x would be a bound for p and q. We can apply Theorem 7.9 (5) to obtain $\sigma_x(p) \subseteq \sigma_x(q)$, and as a consequence $\sigma_x(p) <_2 \sigma_x(q)$.

$p \#_1 q$ implies $\sigma_x(p) \#_2 \sigma_x(q)$. We have two sub-cases according to Definition 8.1:

- $p \#_0 q$ and $\{p, q\} \cap \lfloor x \rfloor_{\mathbb{C}} = \emptyset$. We have $p \not\uparrow_{\mathbb{C}} q$, $p \not\subseteq x$ and $q \not\subseteq x$. Applying Theorem 7.9 (6), we have $\sigma_x(p) \not\uparrow_{x \odot \mathbb{C}} \sigma_x(q)$ and thus $\sigma_x(p) \#_2 \sigma_x(q)$.
- $p <_0 q$, $p \in \lfloor x \rfloor_{\mathbb{C}}$ and $q \notin \lfloor x \rfloor_{\mathbb{C}}$. We have $p \subseteq q$, $p \subseteq x$ and $q \not\subseteq x$. By Theorem 7.9 (4), we get $\sigma_x(p) \not\uparrow_{x \odot \mathbb{C}} \sigma_x(q)$. Therefore $\sigma_x(p) \#_2 \sigma_x(q)$. □

Example 8.4 We show in Figure 4 an illustration of Theorem 8.3, connecting Examples 7.7 and 8.2.

Corollary 8.5 (To Theorem 8.3) *Let $\mathbb{C} \in \mathcal{S}_E$ be a stable configuration structure. For all finite $x \in \mathbb{C}$, $\lfloor x \rfloor_{\mathbb{C}} \odot \mathsf{E}(\mathbb{C})$ is a prime event structure.*

9 A switch operation on polarized event structure

The switch operation on a configuration X reverses causality within X and interchanges conflict and causality at its boundary (see Definition 8.1). Starting from pointed configuration structures, we derived a polarized event structure (Definition 6.1), where events are classified as either positive or negative based on their orientation relative to a given computational past. We now demonstrate that switching a polarized event structure on a configuration X also reverses the signs of the events within X: consuming positive events results in negative ones, while backtracking on negative events restores their positive counterparts.

Definition 9.1 Let $\mathbb{E}[\pi]$ be a polarized event structure. We define its *switching* as: $X \odot \mathbb{E}[\pi] := (X \odot \mathbb{E})[\pi']$ where for all $e \in E$,

$$\pi'(e) := \begin{cases} -\pi(e) & \text{if } e \in X \\ \pi(e) & \text{otherwise.} \end{cases}$$

Similarly to the switch operation on prime event structures, the above operation does not guarantee to produce a polarized event structure, as the negative elements of $X \odot \mathbb{E}[\pi]$ could fail to be downward closed for the causality relation, or exhibit conflict. The theorem below proves that it is not the case.

Theorem 9.2 (Polarity switch) *Let $\mathbb{C}[y^\dagger] \in \mathcal{S}_E$ be a stable pointed configuration structure. For all finite $x \in \mathbb{C}$, the polarized event structure $\lfloor x \rfloor_{\mathbb{C}} \odot \mathsf{E}(\mathbb{C}[y^\dagger])$ is isomorphic to $\mathsf{E}(x \odot \mathbb{C}[y^\dagger])$.*

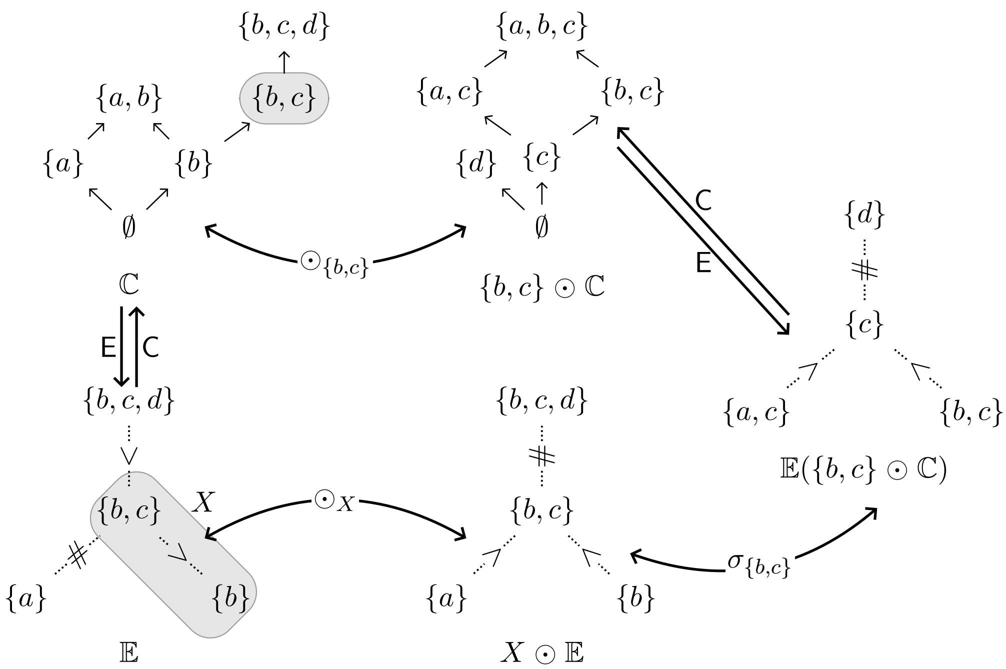


Fig. 4. Illustrating the correspondence between symmetric residuation and switch operations, with $X = \{\{b\}, \{b, c\}\}$.

Proof. Consider:

$$\mathbb{E}_0[\pi_0] \coloneqq \mathsf{E}(\mathbb{C}[y^\dagger]) \qquad \mathbb{E}_1[\pi_1] \coloneqq \lfloor x \rfloor_{\mathbb{C}} \odot \mathbb{E}_0[\pi_0] \qquad \mathbb{E}_2[\pi_2] \coloneqq \mathsf{E}(x \odot \mathbb{C}[y^\dagger])$$

To show that $\mathbb{E}_1[\pi_1]$ is isomorphic to $\mathbb{E}_2[\pi_2]$ we must (i) exhibit an isomorphism between $\mathbb{E}_1$ and $\mathbb{E}_2$ and (ii) show that it is polarity preserving. We already know from Theorem 8.3 that $\mathbb{E}_1 \sim_{\sigma_x} \mathbb{E}_2$, so we just need to verify (ii), i.e., $\pi_1(p) = \pi_2(\sigma_x(p))$ for all $p \in \mathsf{Pr}(\mathbb{C})$. According to Definition 9.1, we have two cases:

- If $p \in \lfloor x \rfloor_{\mathbb{C}}$ then $\pi_1(p) = -\pi_0(p)$. By Proposition 6.2, we have two sub-cases to consider:
  - Suppose $p \subseteq y^\dagger$ and hence $\pi_0(p) < 0$ and $\pi_1(p) > 0$. We must verify $\pi_2(\sigma_x(p)) > 0$. We have $p \subseteq x$ and $p \subseteq y^\dagger$ implies $a \notin x \vartriangle y^\dagger$ and therefore $\sigma_x(p) \nsubseteq x \vartriangle y^\dagger$ (by Proposition 7.3). Hence, $\pi_2(\sigma_x(p)) > 0$.
  - Suppose $p \nsubseteq y^\dagger$ and hence $\pi_0(p) > 0$ and $\pi_1(p) < 0$. We must verify $\pi_2(\sigma_x(p)) < 0$. We have $p \subseteq x$ and $p \nsubseteq y^\dagger$ implies $a \in x \vartriangle y^\dagger$ and therefore $\sigma_x(p) \subseteq x \vartriangle y^\dagger$ (by Proposition 7.3). Hence, $\pi_2(\sigma_x(p)) < 0$.
- If $p \notin \lfloor x \rfloor_{\mathbb{C}}$ then $\pi_1(p) = \pi_0(p)$. We consider again the two sub-cases of Proposition 6.2:
  - Suppose $p \subseteq y^\dagger$ and hence $\pi_0(p) < 0$ and $\pi_1(p) < 0$. We must verify $\pi_2(\sigma_x(p)) < 0$. We have $p \nsubseteq x$ and $p \subseteq y^\dagger$ implies $a \in x \vartriangle y^\dagger$ and therefore $\sigma_x(p) \subseteq x \vartriangle y^\dagger$ (by Proposition 7.3). Hence, $\pi_2(\sigma_x(p)) < 0$.
  - Suppose $p \nsubseteq y^\dagger$ and hence $\pi_0(p) > 0$ and $\pi_1(p) > 0$. We must verify $\pi_2(\sigma_x(p)) > 0$. We have $p \nsubseteq x$ and $p \nsubseteq y^\dagger$ implies $a \notin x \vartriangle y^\dagger$ and therefore $\sigma_x(p) \nsubseteq x \vartriangle y^\dagger$ (by Proposition 7.3). Hence, $\pi_2(\sigma_x(p)) > 0$. □

Corollary 9.3 (To Theorem 9.2) *Let $\mathbb{C} \in \mathcal{S}_E$ be a stable configuration structure and $y^\dagger \in \mathbb{C}$ be a distinguished finite configuration. For all finite $x \in \mathbb{C}$, $\lfloor x \rfloor_{\mathbb{C}} \odot \mathsf{E}(\mathbb{C}[y^\dagger])$ is a polarized event structure.*

10 Discussion

Reversibility is a fundamental property in concurrent systems with applications ranging from debugging to quantum computing, yet its integration with traditional operational semantics remains challenging

In the past two decades, significant research has focused on the design and formalization of reversible concurrent programs [13,33,20,11,23]. This research typically builds upon a classical, non-reversible operational semantics represented as a transition system, say $[\![P]\!] := (P, \mathcal{P}, \rightarrow)$, and turns it into a reversible one, say $[\![P]\!]_{\mathsf{rev}} := (P, \mathcal{R}, \rightarrow_{\mathsf{rev}})$, where $\mathcal{P}$ and $\mathcal{R}$ are the program states that are reachable from the initial state P, under the transition relations. Importantly, verifying that $[\![\cdot]\!]_{\mathsf{rev}}$ is sound involves proving (i) that $\mathcal{P}$ and $\mathcal{R}$ are essentially the same states, after removing additional decorations or encodings that are required to make the transition system reversible, and (ii) that backtracking is causally consistent, i.e., that it is not possible to backtrack on a transition without backtracking first on its consequences. Condition (i) ensures that the reversible semantics does not create new program states that were not reachable in the forward only semantics. Verification techniques to ensure (ii) have been narrowed down to requiring that $\rightarrow_{\mathsf{rev}}$ satisfies the fundamental *causal consistency* property, which states that all co-initial and co-final sequences of transitions, i.e., of the form $P \rightarrow^{a_1}_{\mathsf{rev}} \cdots \rightarrow^{a_n}_{\mathsf{rev}} Q$ and $P \rightarrow^{b_1}_{\mathsf{rev}} \cdots \rightarrow^{b_q}_{\mathsf{rev}} Q$, can be made *equal* by permuting concurrent steps and cancelling out consecutive inverse ones [13,22,25,2]. Importantly, proofs of causal consistency are syntax-intensive proofs which require characterizing causality, conflict and independence of transitions for each formalism whose reversible semantics is being established. We let the reader refer to a more axiomatic approach to reversibility [25] for a principled way of obtaining this result.

The work presented in this paper proposes approaching reversible computations in a syntax-free manner, assuming that a concurrent program is denoted by the set of its observable configurations. This makes practically no assumption on the nature of the underlying process, as configuration structures are just a set of observations, ordered by inclusion. We then showed that the reversible interpretation of a process can be made apparent by changing the traditional interpretation of the residuation of a configuration structure (Definition 2.3) *after* some events have occurred: instead of erasing those events, we showed that it becomes possible to observe inverse events using a *symmetric residuation operation* (Definition 3.1) that is shown to be a group action of configuration structures on themselves (Proposition 3.2).

In a second step, we strengthened the hypothesis on the causal structure of the underlying concurrent process by assuming it can be denoted by a prime event structure. As already well-established in the literature, this corresponds to requiring some stability axioms on its configuration structures (Definition 5.2). Note that stable denotational semantics for concurrent formalisms does not come for free, as multiple formalisms are known to naturally yield unstability [7,12,5]. At this point we showed that stable configuration structures are preserved by symmetric residuation (Theorem 5.8), which guarantees that the symmetric residual of a configuration structure is also a prime event structure:

$$\begin{array}{ccc} \mathbb{C} & \xrightarrow{\textit{Sym. res.}} & x \odot \mathbb{C} \\ \upharpoonleft\!\downharpoonright & & \upharpoonleft\!\downharpoonright \\ \mathbb{E} & \dashrightarrow & \mathbb{E}' \end{array}$$

We then showed that one can instantiate the dotted arrow of the above diagram using a switch operation on prime event structures (Theorem 8.3) which turns causality upside-down amongst events of a configuration, and flips conflict and causality relation at its boundary.

We believe this work lays a foundation for developing a general theory of reversibility in concurrent systems, while also opening up new avenues, some of which we discuss now.

First, notice that rewriting a configuration structure $\mathbb{C}$ into $x \odot \mathbb{C}$ is a global operation which transforms every configuration of $\mathbb{C}$ using symmetric set difference, whereas the switch operation is purely local, since the causality and conflict relations are only modified on pairs of events that appear in x. This indicates a promising technique to verify causal consistency of a reversible operational semantics: reusing the informal notations introduced above, one first needs to prove that $[\![P]\!]_{\mathsf{rev}}$ exhibit a stable causal semantics (its configuration structures are stable), which is a necessary and sufficient condition to establish that prime event structures can be used to denote computations of P. It then suffices to demonstrate that the operational semantics is "switch-like", i.e., a computation event $a : P \to_{\mathsf{rev}} Q$ produces the effect of a switch on the causal and conflict structures of P, which is a predicate on immediate predecessors and successors of a, and on events that are in conflict with a as well (see Definition 8.1). Requiring the reversible operational semantics to be stable and switch-like would then be an alternative characterization of being consistent.

Another intriguing perspective comes from the operation of symmetric residuation, that could be studied as a symmetry on partial order. The definition of the operation $x \odot \mathbb{C}$ requires a set structure and is therefore not directly a transformation of partial orders. However we have shown that the operation does act as a switch on prime event structures, which can be interpreted as particular kinds of 2 sorted directed graph. With this perspective, the switch introduced in this paper becomes an operation close to a Seidel switch [37], originally defined as a symmetry on graphs that consists in swapping edges and non edges that are incident on nodes in a set X, which is the parameter of the switch. We believe the actual relation between our event structure switch and the Seidel switch is worth investigating as Seidel switches on directed graphs is known to produce symmetries of partial orders [32] and have potential interpretations in quantum computing [17], which enjoys reversibility of operations prior to measurements.

References

[1] Abramsky, S., *A structural approach to reversible computation*, Theoretical Computer Science **347**, pages 441–464 (2005). https://doi.org/10.1016/j.tcs.2005.07.002

[2] Aubert, C., *The correctness of concurrencies in (reversible) concurrent calculi*, Journal of Logical and Algebraic Methods in Programming **136**, page 100924 (2024), ISSN 2352-2208. https://doi.org/10.1016/j.jlamp.2023.100924

[3] Aubert, C. and I. Cristescu, *How reversibility can solve traditional questions: The example of hereditary history-preserving bisimulation*, in: I. Konnov and L. Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1–4, 2020, Vienna, Austria*, volume 2017 of *Leibniz International Proceedings in Informatics*, pages 13:1–13:24, Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.7

[4] Aubert, C., I. Phillips and I. Ulidowski, *Bisimulations and reversibility*, in: *Components operationally: Reversibility and system engineering*, volume 16065 of *LNCS*, Springer International Publishing (2025). https://doi.org/10.1007/978-3-031-99717-4_3

[5] Baldan, P., A. Corradini and F. Gadducci, *Domains and event structures for fusions*, in: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005135

[6] Barylska, K., M. Koutny, Ł. Mikulski and M. Piątkowski, *Reversible computation vs. reversibility in petri nets*, Science of Computer Programming **151**, pages 48–60 (2018), ISSN 0167-6423. Special issue of the 8th Conference on Reversible Computation (RC2016). https://doi.org/10.1016/j.scico.2017.10.008

[7] Boreale, M. and D. Sangiorgi, *A fully abstract semantics for causality in the π-calculus*, Acta Inf. **35**, pages 353–400 (1998).

[8] Boudol, G. and I. Castellani, *On the semantics of concurrency: Partial orders and transition systems*, in: H. Ehrig, R. Kowalski, G. Levi and U. Montanari, editors, *TAPSOFT'87*, volume 249 of *Lecture Notes in Computer Science*, pages 123–137, Springer, Berlin, Heidelberg (1987), ISBN 3-540-17660-8. https://doi.org/10.1007/3-540-17660-8_52

[9] Boudol, G. and I. Castellani, *Permutation of transitions: An event structure semantics for CCS and SCCS*, in: J. W. de Bakker, W. P. de Roever and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop, Noordwijkerhout, The Netherlands, May 30 - June 3, 1988, Proceedings*, volume 354 of *Lecture Notes in Computer Science*, pages 411–427, Springer (1988), ISBN 3-540-51080-X. https://doi.org/10.1007/BFb0013028

[10] Buhrman, H., J. Tromp and P. M. B. Vitányi, *Time and space bounds for reversible simulation*, in: F. Orejas, P. G. Spirakis and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 1017–1027, Springer (2001), ISBN 3-540-42287-0. https://doi.org/10.1007/3-540-48224-5_82

[11] Cristescu, I., J. Krivine and D. Varacca, *A compositional semantics for the reversible p-calculus*, in: *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 388–397, IEEE Computer Society (2013).
https://doi.org/10.1109/LICS.2013.45

[12] Cristescu, I., J. Krivine and D. Varacca, *Rigid families for the reversible π-calculus*, in: S. J. Devitt and I. Lanese, editors, *Reversible Computation - 8th International Conference, RC 2016, Bologna, Italy, July 7-8, 2016, Proceedings*, volume 9720 of *Lecture Notes in Computer Science*, pages 3–19, Springer (2016), ISBN 978-3-319-40578-0.
https://doi.org/10.1007/978-3-319-40578-0_1

[13] Danos, V. and J. Krivine, *Reversible communicating systems*, in: P. Gardner and N. Yoshida, editors, *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings*, volume 3170 of *Lecture Notes in Computer Science*, pages 292–307, Springer (2004), ISBN 3-540-22940-X.
https://doi.org/10.1007/978-3-540-28644-8_19

[14] Danos, V. and J. Krivine, *Transactions in RCCS*, in: M. Abadi and L. de Alfaro, editors, *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, volume 3653 of *Lecture Notes in Computer Science*, pages 398–412, Springer (2005), ISBN 3-540-28309-9.
https://doi.org/10.1007/11539452_31

[15] Danos, V., J. Krivine and P. Sobociński, *General reversibility*, in: R. M. Amadio and I. Phillips, editors, *Proceedings of the 13th International Workshop on Expressiveness in Concurrency, EXPRESS 2006, Bonn, Germany, August 26, 2006*, volume 175 of *Electronic Notes in Theoretical Computer Science*, pages 75–86, Elsevier (2006).
https://doi.org/10.1016/J.ENTCS.2006.07.036

[16] De Nicola, R., U. Montanari and F. W. Vaandrager, *Back and forth bisimulations*, in: J. C. M. Baeten and J. W. Klop, editors, *CONCUR '90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 27-30, 1990, Proceedings*, volume 458 of *Lecture Notes in Computer Science*, pages 152–165, Springer (1990).
https://doi.org/10.1007/BFB0039058

[17] Dutta, S., B. Adhikari and S. Banerjee, *Seidel switching for weighted multi-digraphs and its quantum perspective*, ArXiv: Combinatorics (2016).
https://doi.org/10.48550/arXiv.1608.07830

[18] Graversen, E., I. Phillips and N. Yoshida, *Event structure semantics of (controlled) reversible CCS*, in: J. Kari and I. Ulidowski, editors, *Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 102–122, Springer (2018).
https://doi.org/10.1007/978-3-319-99498-7_7

[19] Lanese, I., D. Medić and C. A. Mezzina, *Static versus dynamic reversibility in CCS*, Acta Informatica (2019).
https://doi.org/10.1007/s00236-019-00346-6

[20] Lanese, I., C. A. Mezzina, A. Schmitt and J. Stefani, *Controlling reversibility in higher-order pi*, in: J. Katoen and B. König, editors, *CONCUR 2011 - Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings*, volume 6901 of *Lecture Notes in Computer Science*, pages 297–311, Springer, Berlin, Heidelberg (2011).
https://doi.org/10.1007/978-3-642-23217-6_20

[21] Lanese, I., C. A. Mezzina and J.-B. Stefani, *Reversing higher-order pi*, in: P. Gastin and F. Laroussinie, editors, *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 478–493, Springer (2010), ISBN 978-3-642-15374-7.
https://doi.org/10.1007/978-3-642-15375-4_33

[22] Lanese, I., C. A. Mezzina and F. Tiezzi, *Causal-consistent reversibility*, Bulletin of the European Association for Theoretical Computer Science **114**, page 17 (2014).

[23] Lanese, I., N. Nishida, A. Palacios and G. Vidal, *Cauder: A causal-consistent reversible debugger for erlang*, in: J. P. Gallagher and M. Sulzmann, editors, *Functional and Logic Programming - 14th International Symposium, FLOPS 2018, Nagoya, Japan, May 9-11, 2018, Proceedings*, volume 10818 of *Lecture Notes in Computer Science*, pages 247–263, Springer, Cham (2018), ISBN 978-3-319-90686-7.
https://doi.org/10.1007/978-3-319-90686-7_16

[24] Lanese, I. and I. Phillips, *Forward-reverse observational equivalences in CCSK*, in: S. Yamashita and T. Yokoyama, editors, *Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings*, volume 12805 of *Lecture Notes in Computer Science*, pages 126–143, Springer (2021), ISBN 978-3-030-79836-9.
https://doi.org/10.1007/978-3-030-79837-6_8

[25] Lanese, I., I. Phillips and I. Ulidowski, *An axiomatic theory for reversible computation*, ACM Transactions on Computational Logic **25**, pages 1–40 (2024).
https://doi.org/10.1145/3648474

[26] Lienhardt, M., I. Lanese, C. A. Mezzina and J. Stefani, *A reversible abstract machine and its space overhead*, in: H. Giese and G. Rosu, editors, *Formal Techniques for Distributed Systems - Joint 14th IFIP WG 6.1 International Conference, FMOODS 2012 and 32nd IFIP WG 6.1 International Conference, FORTE 2012, Stockholm, Sweden, June 13-16, 2012. Proceedings*, volume 7273 of *Lecture Notes in Computer Science*, pages 1–17, Springer, Berlin, Heidelberg (2012), ISBN 978-3-642-30793-5.
https://doi.org/10.1007/978-3-642-30793-5_1

[27] Melgratti, H., C. A. Mezzina and G. Michele Pinna, *A distributed operational view of reversible prime event structures*, in: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13 (2021).
https://doi.org/10.1109/LICS52264.2021.9470623

[28] Melgratti, H., C. A. Mezzina and G. M. Pinna, *A truly concurrent semantics for reversible ccs*, Logical Methods in Computer Science **Volume 20, Issue 4**, 20 (2024), ISSN 1860-5974.
https://doi.org/10.46298/lmcs-20(4:20)2024

[29] Milner, R., *Communication and Concurrency*, PHI Series in computer science, Prentice-Hall (1989), ISBN 978-0-13-115007-2.

[30] Montanari, U. and M. Pistore, *Minimal transition systems for history-preserving bisimulation*, in: R. Reischuk and M. Morvan, editors, *STACS 97, 14th Annual Symposium on Theoretical Aspects of Computer Science, Lübeck, Germany, February 27 - March 1, 1997, Proceedings*, volume 1200 of *Lecture Notes in Computer Science*, pages 413–425, Springer (1997).
https://doi.org/10.1007/BFB0023477

[31] Nielsen, M., G. D. Plotkin and G. Winskel, *Petri nets, event structures and domains, part I*, Theoretical Computer Science **13**, pages 85–108 (1981).
https://doi.org/10.1016/0304-3975(81)90112-2

[32] Pach, P. P., M. Pinsker, A. Pongrácz and C. Szabó, *A new operation on partially ordered sets*, Journal of Combinatorial Theory, Series A **120**, pages 1450–1462 (2013), ISSN 0097-3165.
https://doi.org/https://doi.org/10.1016/j.jcta.2013.04.003

[33] Phillips, I. and I. Ulidowski, *Reversing algebraic process calculi*, in: L. Aceto and A. Ingólfsdóttir, editors, *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings*, volume 3921 of *Lecture Notes in Computer Science*, pages 246–260, Springer (2006), ISBN 3-540-33045-3.
https://doi.org/10.1007/11690634_17

[34] Phillips, I. and I. Ulidowski, *A hierarchy of reverse bisimulations on stable configuration structures*, Mathematical Structures in Computer Science **22**, pages 333–372 (2012), ISSN 0960-1295.
https://doi.org/10.1017/S0960129511000429

[35] Phillips, I. and I. Ulidowski, *Reversibility and asymmetric conflict in event structures*, Journal of Logical and Algebraic Methods in Programming **84**, pages 781–805 (2015), ISSN 2352-2208. "Special Issue on Open Problems in Concurrency Theory".
https://doi.org/10.1016/J.JLAMP.2015.07.004

[36] Prasad, K., *Combinators and bisimulation proofs for restartable systems*, Ph.D. thesis, University of Edinburgh (1987).

[37] Seidel, J. J., *A survey of two-graphs*, Geometry and Combinatorics pages 146–176 (1991).
https://doi.org/10.1016/B978-0-12-189420-7.50018-9

[38] Selinger, P., *Lecture notes on the lambda calculus*, arXiv preprint arXiv:0804.3434 (2008).

[39] Shende, V. V., A. K. Prasad, I. L. Markov and J. P. Hayes, *Synthesis of reversible logic circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**, pages 710–722 (2003).
https://doi.org/10.1109/TCAD.2003.811448

[40] Ulidowski, I., I. Lanese, U. P. Schultz and C. Ferreira, editors, *Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action IC1405*, volume 12070 of *Lecture Notes in Computer Science*, Springer (2020), ISBN 978-3-030-47360-0.
https://doi.org/10.1007/978-3-030-47361-7

[41] Ulidowski, I., I. Phillips and S. Yuen, *Reversing event structures*, New Generation Computing **36**, pages 281–306 (2018).
https://doi.org/10.1007/S00354-018-0040-8

[42] van Glabbeek, R. J. and G. D. Plotkin, *Configuration structures*, in: *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, pages 199–209, IEEE Computer Society (1995). https://doi.org/10.1109/LICS.1995.523257

[43] van Glabbeek, R. J. and G. D. Plotkin, *Configuration structures, event structures and petri nets*, Theoretical Computer Science **410**, pages 4111–4159 (2009). https://doi.org/10.1016/j.tcs.2009.06.014

[44] Winskel, G., *Event structure semantics for CCS and related languages*, in: M. Nielsen and E. M. Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, volume 140 of *Lecture Notes in Computer Science*, pages 561–576, Springer (1982), ISBN 3-540-11576-5. https://doi.org/10.1007/BFb0012800