Comparative Transition System Semantics for Cause-Respecting Reversible Prime Event Structures

Nataliya Gribovskaya Irina Virbitskaite
A.P. Ershov Institute of Informatics Systems
the Siberian Branch of the Russian Academy of Sciences
6, Acad. Lavrentiev avenue, 630090, Novosibirsk, Russia
{natamosk,virbitskaite}@gmail.com

Reversible computing is a new paradigm that has emerged recently and extends the traditional forwards-only computing mode with the ability to execute in backwards, so that computation can run in reverse as easily as in forward. Two approaches to developing transition system (automaton-like) semantics for event structure models are distinguished in the literature. In the first case, states are considered as configurations (sets of already executed events), and transitions between states are built by starting from the initial configuration and repeatedly adding executable events. In the second approach, states are understood as residuals (model fragments that have not yet been executed), and transitions are constructed by starting from the given event structure as the initial state and deleting already executed (and conflicting) parts thereof during execution. The present paper focuses on an investigation of how the two approaches are interrelated for the model of prime event structures extended with cause-respecting reversibility. The bisimilarity of the resulting transition systems is proved, taking into account step semantics of the model under consideration.

1 Introduction

Reversible computations, extensively studied during in recent years, is an unconventional form of computations that can be performed in the forward direction as easily as in the reverse direction. Any sequence of actions executed by the system can subsequently be canceled for some reason (for example, in case of an error), which allows the system to restore previous consistent states, as if these canceled actions were not executed at all. Reversible computing is attracting interest for its applications in many fields including program analysis and debugging [20], programming abstractions for reliable systems [11, 23], modelling biochemical reactions [18], hardware design and quantum computing [12], and etc.

Despite the fact that reversing computations in concurrent/distributed systems has many promising applications, it also involves many technical and conceptual challenges. One of the most essential issues that arise concerns the techniques that should be applied when moving backwards. Several different styles of the undoing of computation have been identified recently. The most prominent of these are backtracking [26], causal reversibility [25, 26], and out-of-causal-order reversibility [26, 28], that differ in the order of executing actions in backward direction. Backtracking is generally understood as the ability to execute past actions in the exact reverse order in which they were executed. Causal reversibility in concurrent systems means that actions that cause others can only be undone after the caused actions are undone first, and that actions which are independent of each other can be reversed in an arbitrary order. Out-of-causal reversibility, a form of reversal most characteristic of biochemical systems, does not preserve causes. The interplay between reversibility and concurrency has been widely studied in various models: parallel rewriting systems [1], cellular automata [16], process calculi [11, 19], Petri nets [6, 13, 26], event structures [24, 27, 30], membrane systems [29], and etc.

Zs. Gazdag, Sz. Iván, G. Kovásznai (Eds.): 16th International Conference on Automata and Formal Languages (AFL 2023)
EPTCS 386, 2023, pp. 112–126, doi:10.4204/EPTCS.386.10

Event structures are a well-established model of concurrency. They were originally proposed by Winskel in his PhD dissertation [32] and were considered as an intermediate abstraction between Scott domains (i.e., a denotational model) and Petri nets (i.e., an operational model). Basically, event structures are collections of possible events, some of which are conflicting (i.e., the execution of an event forbids the execution of other events), while others are causally dependent (i.e., an event cannot be executed if it has not been preceded by other ones), and events that are neither in causal dependency nor in conflict are treated as concurrent. Events are often labelled with actions, to represent different occurrences of the same action. Prime Event Structures (written PESs) are the earliest and simplest form of event structures, where causality is a partial order and conflict between events is inherited by their causal successors. The association of transition system (automaton-like) models with event structures has proved to contribute to studying and solving various problems in the analysis and verification of concurrent systems. It is distinguished two methods of providing transition system semantics for event structures: a configuration-based and a residual-based method. In the first case (see [2, 3, 31, 15, 17, 32, 33] among others), states are understood as sets of events, called configurations, and state transitions are built by starting with the initial configuration and enlarging configurations by already executed events. In the second more 'structural' method (see [5, 9, 10, 17, 21] among others), states are understood as event structures, and transitions are built by starting with the given event structure as an initial state and removing already executed (and conflicting) parts thereof in the course of execution. In the literature, configuration-based transition systems seem to be predominantly used as the semantics of event structures, and residual-based transition systems are actively used in providing operational semantics of process calculi and in demonstrating the consistency of operational and denotational semantics. The two kinds of transition systems have occasionally been treated alongside each other (see [17] as an example), but their general relationship has not been studied for a wide range of existing models. In a seminal paper, viz. [22], bisimulations between configuration-based and residual-based transition systems have been proved to exist for prime event structures [33]. The result of [22] has been extended in [7] to more complex event structure models with asymmetric conflict. The paper [8] demonstrated that when using non-executable events, the removal operators defined in [22, 7] to obtain residuals can be tightened in such a way that isomorphisms, rather than just bisimulations, between the two types of transition systems belonging to a single event structure can be obtained, for a full spectrum of semantics (interleaving, step, pomset, multiset).

Reversible event structures extend event structures to represent reversible computational processes, capable of undoing executed actions by allowing configurations to evolve by eliminating events. In [27, 30], Phillips et. al. determined causal and out-of-causal reversible forms of prime, asymmetric and general event structures and showed the correspondence between their configurations and traditional ones when there are no reversible events. In [4], Aubert and Cristescu have provided a true concurrent semantics of a reversible extension of CCS, RCCS (without auto-concurrency, auto-conflict, or recursion), in terms of configuration structures. In [14], Graversen et. al. have developed a category of reversible bundle event structures with symmetric conflict and used the causal subcategory to model semantics of another reversible extension of CCS, CCSK. They also modified CCSK to control reversibility with a rollback primitive, and gave, by exploiting the capacity for out-of-causal reversibility, semantics of this kind of CCSK in terms of reversible bundle event structures with asymmetric conflict. Constructions associating causal reversible prime event structures to reversible occurrence nets and vice versa have been proposed within causal reversibility in [25], as well as within out-of-causal reversibility in [24].

The aim of this paper is to identify two (configuration-based and residual-based) types of transition system semantics for cause-respecting reversible prime event structures and to understand how these types relate to each other, which can assist in the construction of algebraic calculi to describe and verify

reversible concurrent processes.

This paper is structured as follows. In Section 2, we start with recalling the syntax of prime and reversible prime event structures and their (step) semantics in terms of configurations and traces. In Section 3, we define a removal operator, which is useful for constructing model residuals, and demonstrate the correctness of the operator. In Section 4, we develop two types of transition system semantics for cause-respecting reversible prime event structures and establish bisimulation results between the semantics. In Section 5, we provide some concluding remarks. The proofs of the propositions presented here can be found at www.iis.nsk.su/virb/proofs-AFL-2023.

2 Reversing in Prime Event Structures

In this section, we first recall the notion of prime event structures (PESs) [32] labeled over the set $L = \{a, b, c, \ldots\}$ of actions, and then formulate the concept of reversible prime event structures (RPESs) [27] and consider their (step) semantics and properties.

The behavior of concurrent systems is formally modelled by event structure models where units of the behavior are represented by events. There are different ways to relate events. In prime event structures (PESs), the dependency between events, called causality, is given by a partial order, and the incompatibility is determined by a conflict relation. Two events which are neither in causal dependency nor in conflict are considered independent (concurrent).

Definition 1. *A (labeled)* prime event structure (PES) *(over the set L of actions) is a tuple* $\mathscr{E} = (E, <, \sharp, l, C_0)$*, where*

- E *is a countable set of* events*;*
- $< \subseteq E \times E$ *is an irreflexive partial order (the* causality relation*) satisfying the* principle of finite causes*:* $\forall e \in E \circ \lfloor e \rfloor = \{e' \in E \mid e' < e\}$ *is finite;*
- $\sharp \subseteq E \times E$ *is an irreflexive and symmetric relation (the* conflict relation*) satisfying the* principle of hereditary conflict*:* $\forall e, e', e'' \in E \circ e < e'$ *and* $e \sharp e''$ *then* $e' \sharp e''$*;*
- $l : E \to L$ *is a* labeling function*;*
- $C_0 = \emptyset$ *is the* initial configuration[(1)]*.*

So, the PES is a simple event-based model of concurrent and nondeterministic computations where events labeled over the set L of actions are considered as atomic, indivisible and instantaneous action occurrences, some of which can only be executed after another (i.e. there is a causal dependency represented by a partial order $\leq$ between the events) and some of which might not be executed together (i.e. there is a binary conflict $\sharp$ between the events). In addition, the principle of finite causes and the principle of conflict inheritance are required.

The PES progresses by executing events, thus moving from one state to another, starting from the initial state, which is an empty set. A state called a configuration is a set of events that have occurred. A subset of events $X \subseteq E$ is *left-closed under* $<$ iff for all $e \in X$ it holds that $\lfloor e \rfloor \subseteq X$; is *conflict-free* iff for all $e, e' \in X$ it holds that $\neg(e \sharp e')$, and we denote it with $CF(X)$. A subset $C \subseteq E$ is a *configuration* of $\mathscr{E}$ iff C is finite, left-closed under $<$ and conflict-free.

Reversible prime event structures (RPESs) [27, 30] are based on a weaker form of PESs because conflict inheritance may not hold when adding reversibility to PESs. Also, in RPESs, some events are

(1) We add the initial configuration as an empty set to the classical PES definition, but this does not affect the behavior of the structure in any way, because the PES progresses by moving from one configuration to another and starting from an empty set.

categorised as reversible, and two relations are added: the reverse causality relation and the prevention relation. The first one is a dependency relation in the backward direction: to reverse an event in the current configuration there must be other events on which the event reversibly depends. The second relation, on the contrary, identifies those events whose presence in the current configuration prevents the event being reversed.

Definition 2. *A (labeled)* reversible prime event structure (RPES) *(over L) is a tuple* $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$*, where*

- E *is a countable set of* events*;*
- $\sharp \subseteq E \times E$ *is an irreflexive and symmetric relation (the* conflict relation*);*
- $< \subseteq E \times E$ *is an irreflexive partial order (the* causality relation*) satisfying:* $\lfloor e \rfloor$ *is finite and conflict-free, for every* $e \in E$*;*
- $l : E \to L$ *is a* labeling function*;*
- $F \subseteq E$ *are* reversible events *being denoted by the set* $\underline{F} = \{\underline{e} \mid e \in F\}$ *such that* $\underline{F} \cap E = \emptyset$*;*
- $\prec \subseteq E \times \underline{F}$ *is the* reverse causality relation *satisfying:* $a \prec \underline{a}$ *and* $\{e \in E \mid e \prec \underline{a}\}$ *is finite and conflict-free, for every* $a \in F$*;*
- $\rhd \subseteq E \times \underline{F}$ *is the* prevention relation *such that* $\rhd \cap \prec = \emptyset$*;*
- $\ll$ *is the transitive* sustained causation relation*:* $a \ll b$ *is defined to mean that* $a < b$ *and if* $a \in F$ *then* $b \rhd \underline{a}$*.* $\sharp$ *is* hereditary w.r.t. the sustained causation $\ll$*: if* $a \sharp b \ll c$ *then* $a \sharp c$*;*
- $C_0 \subseteq E$ *is the* initial configuration *which is finite, left-closed under* $<$ *and conflict-free.*

It is straightforward to check that any PES is also an RPES with $F = \emptyset$ and $C_0 = \emptyset$. Then, any concept defined for RPESs applies to PESs as well.

Example 1. Consider the structure $\mathscr{E}_0 = (E_0, <_0, \sharp_0, l_0, F_0, \prec_0, \rhd_0, C_0^0)$, where $E_0 = \{a,b,c,d,e\}$; $<_0 = \{(b,d),(c,e)\}$; $\sharp_0 = \{(a,b),(b,a),(b,c),(c,b)\}$; l_0 is the identical function; $F_0 = \{b,c\}$; $\prec_0 = \{(b,\underline{b}),(c,\underline{c})\}$; $\rhd_0 = \emptyset$; $C_0^0 = \emptyset$. It is easy to make sure that the components of the structure $\mathscr{E}_0$ meet the requirements of the corresponding items of Definition 2. In particular, we see that $\prec_0 = \{(b,\underline{b}),(c,\underline{c})\}$ and $(b,\underline{b}),(c,\underline{c}) \notin \rhd_0$. Notice that $\sharp_0$ is not hereditary w.r.t. $<_0$ because $a \sharp_0 b <_0 d$ and $\neg(a \sharp_0 d)$, $b \sharp_0 c <_0 e$ and $\neg(b \sharp_0 e)$, $c \sharp_0 b <_0 d$ and $\neg(c \sharp_0 d)$. From Definition 2, we know that x and y are in the sustained causation relation iff x causes y, and x cannot be reversed as long as y is present. In $\mathscr{E}_0$, the pairs (b,d) and (c,e) are in the causality relation $<_0$, and the prevention relation $\rhd_0$ is empty. Therefore, the sustained causation relation $\ll_0$ is empty. It is easy to see that $\sharp_0$ is hereditary w.r.t. $\ll_0$. So, the structure $\mathscr{E}_0$ is indeed an RPES. ◇

The RPES progresses by executing events and/or by undoing previously executed events, thus moving from one configuration to another. The act of moving is a computation step. Reachable configurations are subsets of events which can be reached from the initial configuration by executing computation steps. A sequence of computation steps is a trace of the RPES.

Definition 3. *Given an RPES* $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$*, and* $C \subseteq E$ *such that* $CF(C)$*,*

- *for* $A \subseteq E$ *and* $B \subseteq F$*, we say that* $A \cup \underline{B}$ *is* enabled at C *if*
  - *a)* $A \cap C = \emptyset$*,* $B \subseteq C$*,* $CF(C \cup A)$*;*
  - *b)* $\forall e \in A, \forall e' \in E$ *:* *if* $e' < e$ *then* $e' \in (C \setminus B)$*;*
  - *c)* $\forall e \in B, \forall e' \in E$ *:* *if* $e' \prec \underline{e}$ *then* $e' \in (C \setminus (B \setminus \{e\}))$*;*

d) $\forall e \in B, \forall e' \in E$: *if* $e' \rhd \underline{e}$ *then* $e' \notin (C \cup A)$.

If $A \cup \underline{B}$ *is enabled at* C *then* $C \xrightarrow{A \cup \underline{B}} C' = (C \setminus B) \cup A$. *We shall write* $l(A \cup \underline{B}) = M$ *iff* M *is a multiset over the set* L *of actions, defined as follows:* $M(a) = \mid \{e \in (A \cup B) \mid l(e) = a\} \mid$ *for all* $a \in L$.

- C *is a* forwards reachable configuration of $\mathscr{E}$ *(from* C_0*) iff for all* $i = 1, \ldots, n$ $(n \geq 0)$, *there exists a finite set* $A_i \subseteq E$ *such that* $C_{i-1} \xrightarrow{A_i \cup \underline{\emptyset}_i} C_i$ *and* $C_n = C$.
- C *is a* (reachable) configuration of $\mathscr{E}$ *(from* C_0*) iff for all* $i = 1, \ldots, n$ $(n \geq 0)$, *there exist finite sets* $A_i \subseteq E$ *and* $B_i \subseteq F$ *such that* $C_{i-1} \xrightarrow{A_i \cup \underline{B}_i} C_i$ *and* $C_n = C$. *In this case,* $t = (A_1 \cup \underline{B}_1) \ldots (A_n \cup \underline{B}_n)$ $(n \geq 0)$ *is a trace of* $\mathscr{E}$ *and* $last(t) = C_n$. *The set of (reachable) configurations of* $\mathscr{E}$ *is denoted by* $Conf(\mathscr{E})$, *and the set of traces of* $\mathscr{E}$ *— by* $Traces(\mathscr{E})$. *Clearly, any configuration* $C \in Conf(\mathscr{E})$ *is conflict-free, and any prefix of any trace* $t \in Traces(\mathscr{E})$ *belongs to* $Traces(\mathscr{E})$.
- *Two traces* $t = (A_1 \cup \underline{B}_1) \ldots (A_n \cup \underline{B}_n)$ $(n \geq 0)$ *and* $t' = (A'_1 \cup \underline{B'}_1) \ldots (A'_m \cup \underline{B'}_m)$ $(m \geq 0)$ *of* $\mathscr{E}$ *are called to be* equivalent w.r.t. $\sim$ *(denoted* $t \sim t'$*) iff* $last(t) = last(t')$.

The last two items of Definition 3 lead to the following auxiliary

Lemma 1. *Given an RPES* $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$, *it holds:*

(i) $\{last(t) \mid t \in Traces(\mathscr{E})\} = Conf(\mathscr{E})$;

(ii) for any $t \in Traces(\mathscr{E})$, *if* $t(A \cup \underline{B}) \in Traces(\mathscr{E})$ *then* $last(t) \xrightarrow{A \cup \underline{B}} last(t(A \cup \underline{B}))$;

(iii) for any $t, t' \in Traces(\mathscr{E})$, *if* $last(t) \xrightarrow{A \cup \underline{B}} last(t')$ *then* $t(A \cup \underline{B}) \in Traces(\mathscr{E})$ *and* $t(A \cup \underline{B}) \sim t'$.

Example 2. First, recall the RPES $\mathscr{E}_0 = (E_0, <_0, \sharp_0, l_0, F_0, \prec_0, \rhd_0, C_0^0)$ (see Example 1) with the components: $E_0 = \{a,b,c,d,e\}$; $<_0 = \{(b,d),(c,e)\}$; $\sharp_0 = \{(a,b),(b,a),(b,c),(c,b)\}$; l_0 is the identical function; $F_0 = \{b,c\}$; $\prec_0 = \{(b,\underline{b}),(c,\underline{c})\}$; $\rhd_0 = \emptyset$; $C_0^0 = \emptyset$. We shall check if the sequence $t = (\{b\} \cup \emptyset)(\{d\} \cup \emptyset)(\emptyset \cup \{\underline{b}\})(\{c\} \cup \emptyset)(\{e\} \cup \emptyset)(\emptyset \cup \{\underline{c}\})$ is a trace of $\mathscr{E}_0$, using Definition 3. First, we need to show that $((A_1 = \{b\}) \cup (\underline{B}_1 = \emptyset))$ is enabled at C_0^0. Item a) is true because $(A_1 = \{b\}) \cap (C_0^0 = \emptyset) = \emptyset$, $B_1 = \emptyset \subseteq C_0^0$, and $(\emptyset \cup \{b\})$ is conflict-free. Item b) is correct, since the event b has no causes, i.e. there is no $e' \in E_0$ such that $e' <_0 b$. As $B_1 = \emptyset$, items c) and d) are met. Then, we have $C_0^0 \xrightarrow{\{b\} \cup \emptyset} C_1^0 = \{b\}$. Second, verify if $((A_2 = \{d\}) \cup (\underline{B}_2 = \emptyset))$ is enabled at C_1^0. We see that $(A_2 = \{d\}) \cap (C_1^0 = \{b\}) = \emptyset$, $B_2 = \emptyset \subseteq C_1^0$, and $\{b,d\}$ is conflict-free. Hence, item a) is correct. Item b) is met because d has the only cause b belonging to $C_1^0 \setminus B_2$. Due to $B_2 = \emptyset$, items c) and d) are true. So, we get $C_1^0 \xrightarrow{\{d\} \cup \emptyset} C_2^0 = \{b,d\}$. Third, make sure that $((A_3 = \emptyset) \cup (\underline{B}_3 = \{\underline{b}\}))$ is enabled at C_2^0. Items a) is fulfilled thanks to $A_3 = \emptyset$, $B_3 = \{b\} \subseteq C_2^0$, and C_2^0 is conflict-free. Clearly, item b) is true. Item c) holds because the only reverse cause for the event b is the event itself, which is in $\{b,d\} = (C_2^0 \setminus (B_3 \setminus \{b\}))$. As $\rhd_0 = \emptyset$, item d) is correct. Hence, we obtain $C_2^0 \xrightarrow{\emptyset \cup \{\underline{b}\}} C_3^0 = \{d\}$. Fourth, demonstrate that $((A_4 = \{c\}) \cup (\underline{B}_4 = \emptyset))$ is enabled at C_3^0. We see that $(A_4 = \{c\}) \cap (C_3^0 = \{d\}) = \emptyset$, $B_4 = \emptyset \subseteq C_3^0$, and $C_3^0 \cup A_4 = \{c,d\}$ is conflict-free. This means that item a) is correct. Item b) is met thanks to the fact that c has no causes. Because of $B_4 = \emptyset$, items c) and d) are met. Therefore, $C_3^0 \xrightarrow{\{c\} \cup \emptyset} C_4^0 = \{c,d\}$ is true. Fifth, check that $((A_5 = \{e\}) \cup (\underline{B}_5 = \emptyset))$ is enabled at C_4^0. Since $(A_5 = \{e\}) \cap (C_4^0 = \{c,d\}) = \emptyset$, $B_5 = \emptyset \subseteq C_4^0$, $(C_4^0 \cup A_5) = \{c,d,e\}$ is conflict-free, item a) is correct. As e has the only cause c belonging $C_4^0 \setminus B_5$, item b) is met. Due to $B_5 = \emptyset$, items c) and d) are true. Hence, we get $C_4^0 \xrightarrow{\{e\} \cup \emptyset} C_5^0 = \{c,d,e\}$. Finally, we examine if $((A_6 = \emptyset) \cup (\underline{B}_6 = \{\underline{c}\}))$ is enabled at C_5^0. Item a) is fulfilled thanks to $A_6 = \emptyset$, $B_6 = \{c\} \subseteq C_5^0$, and C_5^0 is conflict-free. Obviously, item b) is true. Item c) holds because the only reverse cause for the event c is the event itself, which is in

$\{c,d,e\} = (C_5^0 \setminus (B_6 \setminus \{c\}))$. Because of $\rhd_0 = \emptyset$, item d) is correct. So, we obtain $C_5^0 \stackrel{\emptyset\cup\{\underline{c}\}}{\longrightarrow} C_6^0 = \{d,e\}$. Thus, t is indeed a trace of $\mathscr{E}_0$.

Reasoning analogously, we get the following configurations of $\mathscr{E}_0$: $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$, $\{b,d\}$, $\{c,e\}$, $\{c,d\}$, $\{b,e\}$, $\{d,e\}$, $\{b,d,e\}$, $\{c,d,e\}$. Since the event a is independent with each of the events c, d, e, we get the additional configurations: $\{a,c\}$, $\{a,d\}$, $\{a,e\}$, $\{a,c,e\}$, $\{a,c,d\}$, $\{a,d,e\}$, $\{a,c,d,e\}$. Since the pair (a,b) ((b,c)) is in the conflict relation $\sharp_0$, the events a and b (b and c) cannot occur together in any configuration. Therefore, all the configurations of $\mathscr{E}_0$ are listed above. Some of the maximal traces of $\mathscr{E}_0$ are: $(t_1t_2)^*(\{a\}\cup\emptyset)t_2t_4t_2$, $(t_1t_2)^*t_4(t_1t_2)^*(\{a,c\}\cup\emptyset)t_5$, $(t_1t_2)^*t_3(t_1t_2)^*t_4(t_1t_2)^*(\{a,c\}\cup\emptyset)t_5$, $(t_1t_2)^*t_3(t_1t_2)^*(\{c\}\cup\emptyset)(\{a,e\}\cup\emptyset)t_5$, where $t_1 = (((\{b\}\cup\emptyset)(\emptyset\cup\{\underline{b}\}))^*$, $t_2 = (((\{c\}\cup\emptyset)(\emptyset\cup\{\underline{c}\}))^*$, $t_3 = (\{b\}\cup\emptyset)(\{d\}\cup\emptyset)(\emptyset\cup\{\underline{b}\})$, $t_4 = (\{c\}\cup\emptyset)(\{e\}\cup\emptyset)(\emptyset\cup\{\underline{c}\})$, $t_5 = (\emptyset\cup\{\underline{c}\})(\{c\}\cup\emptyset)$.

Second, consider the structure $\mathscr{E}_1 = (E_1, <_1, \sharp_1, l_1, F_1, \prec_1, \rhd_1, C_0^1)$, where $E_1 = \{a,b\}$; $<_1 = \{(a,b)\}$; $\sharp_1 = \emptyset$; l_1 is the identical function; $F_1 = \{a\}$; $\prec_1 = \{(a,\underline{a})\}$; $\rhd_1 = \emptyset$; $C_0^1 = \emptyset$. It is easy to see that $\mathscr{E}_1$ is an RPES. As the only pair (a,b) is in the causality relation $<_1$, i.e., the event a has no cause and it causes the event b, the event a can occur first and only after that b can happen. Then, we obtain the forward steps: $\emptyset \stackrel{(\{a\}\cup\emptyset)}{\rightarrow} \{a\} \stackrel{(\{b\}\cup\emptyset)}{\rightarrow} \{a,b\}$. The intended meaning of $a \prec_1 \underline{a}$ is that the event a can be undone if it has occurred in a configuration. In this regard, the reverse step $\{a\} \stackrel{(\emptyset\cup\{\underline{a}\})}{\rightarrow} \emptyset$ is possible, thanks to $(b,\underline{a}) \notin \prec_1$ and $\rhd_1 = \emptyset$. Moreover, the event a can be undone in the configuration $\{a,b\}$ even though the event b is present because $(b,\underline{a}) \notin \rhd_1$. This means that we can move backwards from $\{a,b\}$ to $\{b\}$ by executing the step $(\emptyset\cup\{\underline{a}\})$. Therefore, the configurations of $\mathscr{E}_1$ are $\emptyset$, $\{a\}$, $\{b\}$, $\{a,b\}$, and the traces of $\mathscr{E}_1$ are all prefixes of the trace $(((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a\}\cup\emptyset)(\{b\}\cup\emptyset)((\emptyset\cup\{\underline{a}\})(\{a\}\cup\emptyset))^*(\emptyset\cup\{\underline{a}\})$.

Third, examine the structure RPES $\mathscr{E}_2 = (E_2, <_2, \sharp_2, l_2, F_2, \prec_2, \rhd_2, C_0^2)$, where $E_2 = \{a,b\}$; $<_2 = \emptyset$; $\sharp_2 = \emptyset$; l_2 is the identical function; $F_2 = \{a\}$; $\prec_2 = \{(a,\underline{a})\}$; $\rhd_2 = \{(b,\underline{a})\}$; $C_0^2 = \emptyset$. It is not difficult to check that $\mathscr{E}_2$ is an RPES. As the causality relation $<_2$ and the conflict relation $\sharp_2$ are empty, the events a and b are independent, and, therefore, they can take place in any order. This leads to the following forward steps: $\emptyset \stackrel{(\{a\}\cup\emptyset)}{\rightarrow} \{a\} \stackrel{(\{b\}\cup\emptyset)}{\rightarrow} \{a,b\}$ and $\emptyset \stackrel{(\{b\}\cup\emptyset)}{\rightarrow} \{b\} \stackrel{(\{a\}\cup\emptyset)}{\rightarrow} \{a,b\}$. Since $b \rhd_2 \underline{a}$, we conclude that b prevents the undoing of a, i.e. a cannot be undone if b is present. So, we can go back from $\{a\}$ to $\emptyset$ by executing the step $(\emptyset\cup\{\underline{a}\})$ and cannot move backwards from $\{a,b\}$. The configurations of $\mathscr{E}_2$ are $\emptyset$, $\{a\}$, $\{b\}$, $\{a,b\}$, and the traces of $\mathscr{E}_2$ are all prefixes of the traces $(((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a\}\cup\emptyset)(\{b\}\cup\emptyset)$, $(((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a,b\}\cup\emptyset)$, $(((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{b\}\cup\emptyset)(\{a\}\cup\emptyset)$.

It is not difficult to verify the truth of Lemma 1 for all the RPESs discussed above. ◇

RPESs are able to model such a peculiarity of reversible computation as causal-consistent reversibility which relates reversibility with causality: an event can be undone provided that all of its effects have been undone. This allows the system to get back to a past state, which could only be reached by forward computation. This notion of reversibility is natural in reliable concurrent systems since when an error occurs the system tries to go back to a past consistent state.

Definition 4. *An RPES $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$ is called*

- cause-respecting *if for any $e,e' \in E$, if $e < e'$ then $e \ll e'$;*
- causal *if for any $e \in E$ and $u \in F$ it holds: $e \prec \underline{u}$ iff $e = u$, and $e \rhd \underline{u}$ iff $u < e$.*

Informally, in the cause-respecting and causal RPES, causes can be only undone if their effects are not present in the current configuration. Clearly, if the RPES is causal, then it is cause-respecting as well.

Example 3. First, recall the RPES $\mathscr{E}_0$ (see Examples 1 and 2) with the components: $E_0 = \{a,b,c,d,e\}$; $<_0 = \{(b,d),(c,e)\}$; $\sharp_0 = \{(a,b),(b,a),(b,c),(c,b)\}$; l_0 is the identical function; $F_0 = \{b,c\}$; $\prec_0 =$

$\{(b,\underline{b}),(c,\underline{c})\}$; $\rhd_0=\emptyset$; $C_0^0=\emptyset$. We know from Example 1 that the sustained causation relation $\ll_0$ is empty, because the causality relation $<_0$ contains the pairs (b,d) and (c,e) and the prevention relation $\rhd_0$ is empty. Since $\ll_0 \neq <_0$, we have that this RPES is neither cause-respecting nor causal.

Second, consider the RPES $\mathscr{E}_1$ (see Example 2) with the components: $E_1=\{a,b\}$; $<_1=\{(a,b)\}$; $\sharp_1=\emptyset$; l_1 is the identical function; $F_1=\{a\}$; $\prec_1=\{(a,\underline{a})\}$; $\rhd_1=\emptyset$; $C_0^1=\emptyset$. It is easy to see that $\ll_1=\emptyset$, since $<_1=\{(a,b)\}$ and $(b,\underline{a})\notin\rhd_1$. Then, we obtain $<_1\neq\ll_1$. So, this RPES is neither cause-respecting nor causal.

Third, examine the RPES $\mathscr{E}_2=(E_2,<_2,\sharp_2,l_2,F_2,\prec_2,\rhd_2,C_0^2)$ (see Example 2) with the components: $E_2=\{a,b\}$; $<_2=\emptyset$; $\sharp_2=\emptyset$; l_2 is the identical function; $F_2=\{a\}$; $\prec_2=\{(a,\underline{a})\}$; $\rhd_2=\{(b,\underline{a})\}$; $C_0^2=\emptyset$. The RPES is cause-respecting, because the causality relation $<_2$ is empty, and, hence, for the only reversible event a of $\mathscr{E}_3$, the set of its effects is empty, which implies $<_2=\ll_2=\emptyset$. On the other hand, $\mathscr{E}_2$ is not causal, because there are the events a and b such that $b\rhd_2\underline{a}$ and $a\not<_2 b$.

Fourth, treat the RPES $\mathscr{E}_3=(E_3,<_3,\sharp_3,l_3,F_3,\prec_3,\rhd_3,C_0^3)$, where $E_3=\{a,b,c,d\}$; $<_3=\{(b,d),(c,d)\}$; $\sharp_3=\{(a,c),(c,a),(a,d),(d,a)\}$; l_3 is the identical function; $F_3=\{b\}$; $\prec_3=\{(a,\underline{b}),(b,\underline{b})\}$; $\rhd_3=\{(d,\underline{b})\}$ and $C_0^3=\{b\}$. Since for the only reversible event b, the set of its effects is equal to $\{d\}$ and $d\rhd_3\underline{b}$ is true, we conclude that the RPES is cause-respecting, whereas it is not causal because $(a,\underline{b})\in\prec_3$ and $a\neq b$.

Finally, consider the RPES $\mathscr{E}_4=(E_4,<_4,\sharp_4,l_4,F_4,\prec_4,\rhd_4,C_0^4)$, where $E_4=\{a,b,c,d\}$; $<_4=\{(c,d)\}$; $\sharp_4=\{(a,c),(c,a),(a,d),(d,a)\}$; l_4 is the identical function; $F_4=\{c,b\}$; $\prec_4=\{(b,\underline{b}),(c,\underline{c})\}$; $\rhd_4=\{(d,\underline{c})\}$, $C_0^4=\{b,c\}$. The RPES is causal and therefore cause-respecting. This is because $<_4=\{(c,d)\}$ and $\rhd_4=\{(d,\underline{c})\}$, and the reverse cause for the undoing of the only reversible event is the event itself, since we have $F_4=\{b,c\}$ and $\prec_4=\{(b,\underline{b}),(c,\underline{c})\}$. $\diamond$

Any cause-respecting RPES with the empty initial configuration can be presented as a PES. On the other hand, any PES can be converted into a causal and therefore cause-respecting RPES with the empty initial configuration, once we specify which events are to be reversible. The following facts are slight modifications of Propositions 3.36 and 3.37 from [27].

Proposition 1.

(i) If $\mathscr{E}=(E,<,\sharp,l,F,\prec,\rhd,\emptyset)$ is a cause-respecting RPES then $\phi(\mathscr{E})=(E,<,\sharp,l,\emptyset)$ is a PES.

(ii) If $\mathscr{E}=(E,<,\sharp,l,\emptyset)$ is a PES and $F\subseteq E$ then $\varphi(\mathscr{E},F)=(E,<,\sharp,l,F,\prec,\rhd,\emptyset)$ is a causal RPES, where $e\prec\underline{e}$ for any $e\in F$, and $e\rhd\underline{e'}$ for any $e\in E$ and $e'\in F$ such that $e'<e$. Moreover, $\phi(\varphi(\mathscr{E},F))=\mathscr{E}$.

The following lemma states specific features of the configurations of the cause-respecting RPES, which are left-closed w.r.t. causality and forwards reachable. Thanks to Definitions 2 and 3, the truth of item (i) follows from Proposition 3.38(1) [27], and the truth of item (ii) — from Proposition 3.40(2) [27].

Lemma 2. *Given a cause-respecting $\mathscr{E}$ and its configuration $C\in Conf(\mathscr{E})$, it holds:*

(i) C is left-closed under $<$;

(ii) if C is reachable, then C is forwards reachable.

The below example explains the above lemma.

Example 4. Recall the non-cause-respecting RPES $\mathscr{E}_0$ (with $<_0=\{(b,d),(c,e)\}$) from Examples 1–3. We know that $\{d\}$, $\{e\}$, $\{b,e\}$, $\{c,d\}$, $\{d,e\}$, $\{b,d,e\}$, $\{c,d,e\}$, $\{a,d\}$, $\{a,e\}$, $\{a,c,d\}$, $\{a,d,e\}$, $\{a,c,e,d\}$ are configurations of $\mathscr{E}_0$. Clearly, these configurations are not left-closed under $<_0$. Also,

we can reach the configurations only by using a combination of forward and reverse steps, i.e. the configurations are reachable but not forwards reachable.

Consider the non-cause-respecting RPES $\mathscr{E}_1$ (with $<_1=\{(a,b)\}$) from Examples 2–3. The configurations of $\mathscr{E}_1$ are $\emptyset$, $\{a\}$, $\{b\}$, $\{a,b\}$. We see that the configuration $\{b\}$ is not left-closed under $<_1$. In addition, the configuration $\{b\}$ can only be reached with a combination of forward and reverse steps, but this is not possible when doing only forward steps.

It is easy to check that in the cause-respecting RPES $\mathscr{E}_2$ from Examples 2–3, all its configurations are left-closed under its causality relation and, moreover, forwards reachable. $\diamond$

3 Residuals

The removal operator, the concept of which is based on deleting already executed configurations (traces) and events that conflict with the events presenting in the configurations (traces), is necessary for residual semantics.

Introduce the definition of the removal operator for RPESs by using their traces.

Definition 5. *For an RPES* $\mathscr{E}=(E,<,\sharp,l,F,\prec,\rhd,C_0)$ *and its trace* $t=(A_1\cup\underline{B}_1)\ \ldots\ (A_n\cup\underline{B}_n)\in Traces(\mathscr{E})$ *(*$n\geq 0$*), the residual* $\mathscr{E}\setminus t$ *of* $\mathscr{E}$ *after* t *under the removal operator* $\setminus$ *is defined by induction on* $0\leq i\leq n$ *as follows:*

$i=0$. $\mathscr{E}\setminus(t_0=\varepsilon)=\mathscr{E}$.

$i>0$. $\mathscr{E}\setminus t_i=(E^i,\ <^i=<^{i-1}\cap(E^i\times E^i),\ \sharp^i=\sharp^{i-1}\cap(E^i\times E^i),\ l^i=l^{i-1}\mid_{E^i},\ F^i,\ \prec^i=\prec^{i-1}\cap(E^i\times\underline{F}^i),$
$\rhd^i=\rhd^{i-1}\cap\ (E^i\times\underline{F}^i),\ C_0^i)$, *with*

- $E^i=E^{i-1}\setminus(\widetilde{A}_i\cup\sharp^{i-1}(\widetilde{A}_i))$, *where*
  $\widetilde{A}_i=(A_i\setminus F^{i-1})\ \cup\ (\lfloor(A_i\setminus F^{i-1})\rfloor\cap F^{i-1}=\{\widetilde{a}\in F^{i-1}\mid\exists a\in A_i\setminus F^{i-1}\colon\widetilde{a}<^{i-1}a\})$,
  $\sharp^{i-1}(\widetilde{A}_i)=\{a\in E^{i-1}\mid\exists\widetilde{a}\in\widetilde{A}_i\ :\ a\,\sharp^{i-1}\,\widetilde{a}\}$;
- $F^i=(F^{i-1}\cap E^i)\setminus\big(\hat{A}_i\cup\hat{\hat{A}}_i\big)$, *where*
  $\hat{A}_i=\{e\in F^{i-1}\mid\exists a\in\sharp^{i-1}(\widetilde{A}_i)\colon a\prec^{i-1}\underline{e}\}$,
  $\hat{\hat{A}}_i=\{e\in F^{i-1}\mid\exists a\in\widetilde{A}_i\colon a\rhd^{i-1}\underline{e}\}$;
- $C_0^i=((C_0^{i-1}\setminus B_i)\cup A_i)\cap E^i$.

$\mathscr{E}\setminus t=\mathscr{E}\setminus t_n$.

The intuitive interpretation of the above definition is as follows. In the process of constructing the residual of the RPES after a trace, all the irreversible events occurred in the current computation step, their reversible causes and conflicting events thereof are removed, yielding a reduction of all the relations, the labelling function and the initial configuration in the residual. This is due to the fact that all these removed events will never be able to occur in any subsequent step. In addition, reversible events become irreversible, whenever at least one of their reverse causes and/or at least one of the events preventing their undoing are eliminated because the reversible events can never be undone afterwards. At the same time, the other reversible events presented in the current step are retained, since they can be reversed in next steps.

It should be emphasized that for any trace t of the RPES $\varphi(\mathscr{E},\emptyset)^{(2)}$, where $\mathscr{E}$ is a PES, the residual $\varphi(\mathscr{E},\emptyset)\setminus t$ coincides with the residual $\mathscr{E}\ \setminus'\ last(t)$, where $\setminus'$ is the removal operator defined in [22]$^{(3)}$.

(2) See Proposition 1(ii).

(3) In [22], for the PES $\mathscr{E}=(E,<,\sharp,l)$ and its configuration $C\in Conf(\mathscr{E})$, the residual $\mathscr{E}\ \setminus'\ C$ is defined as follows: $\mathscr{E}\ \setminus'\ C=(E'=E\setminus(C\cup\sharp(C)),\leq\cap(E'\times E'),\sharp\cap(E'\times E'),l\mid_{E'})$, where $\sharp(C)$ denotes the events conflicting with the events in C.

We illustrate the application of the above removal operator with

Example 5. Consider the RPES $\mathscr{E}_2 = (E_2, <_2, \sharp_2, l_2, F_2, \prec_2, \rhd_2, C_0^2)$ (see Examples 2–4) with the components: $E_2 = \{a,b\}$; $<_2 = \emptyset$; $\sharp_2 = \emptyset$; l_2 is the identical function; $F_2 = \{a\}$; $\prec_2 = \{(a,\underline{a})\}$; $\rhd_2 = \{(b,\underline{a})\}$; $C_0^2 = \emptyset$. From Example 2 we know that the traces of $\mathscr{E}_2$ are $((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*$, $((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a\}\cup\emptyset)$, $((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a\}\cup\emptyset)(\{b\}\cup\emptyset)$, $((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a,b\}\cup\emptyset)$, $((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{b\}\cup\emptyset)$, $((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{b\}\cup\emptyset)(\{a\}\cup\emptyset)$.

Applying the removal operator to the RPES $\mathscr{E}_2$ and its traces, we obtain the following structures:

- $\tilde{\mathscr{E}}_2 = \mathscr{E}_2 \setminus (A_1 = \{a\} \cup \underline{B}_1 = \emptyset) = (\tilde{E} = E_2, \tilde{<} = <_2, \tilde{\sharp} = \sharp_2, \tilde{l} = l_2, \tilde{F} = F_2, \tilde{\prec} = \prec_2, \tilde{\rhd} = \rhd_2, \tilde{C}_0 = \{a\})$, because $(\widetilde{A}_1 \cup \sharp_2(\widetilde{A}_1)) = \emptyset$, due to $a \in F_2$, and $\tilde{C}_0 = ((C_0^2 = \emptyset) \cup (A_1 = \{a\})) \cap (\tilde{E} = \{a,b\}) = \{a\}$;
- $\hat{\mathscr{E}}_2 = \mathscr{E}_2 \setminus (A_1 = \{a\} \cup \underline{B}_1 = \emptyset)(A_2 = \emptyset \cup \underline{B}_2 = \{\underline{a}\}) = \mathscr{E}_2$, since $(\widetilde{A}_2 \cup \dot{\sharp}(\widetilde{A}_2)) = \emptyset$, thanks to $a \in \tilde{F}$, and $((\tilde{C}_0 = \{a\}) \setminus B_2 = \{a\}) \cap \hat{E}_2 = \emptyset$;
- $\check{\mathscr{E}}_2 = \mathscr{E}_2 \setminus (A_1 = \{a\} \cup \underline{B}_1 = \emptyset)(A_2 = \{b\} \cup \underline{B}_2 = \emptyset) = (\check{E} = \{a\}, \check{<} = \emptyset, \check{\sharp} = \emptyset; \check{l} = l_2|_{\{a\}}; \check{F} = \emptyset; \check{\prec} = \emptyset; \check{\rhd} = \emptyset, \check{C}_0 = \{a\})$, because $\widetilde{A}_2 = \{b\}$, due to $b \in A_2 \setminus \tilde{F}$, $a \notin \check{F}$, due to $(b,\underline{a}) \in \tilde{\rhd}$, and $\check{C}_0 = ((\tilde{C}_0 = \{a\}) \cup (A_2 = \{b\})) \cap (\check{E} = \{a\}) = \{a\}$;
- $\check{\mathscr{E}}_2 = \mathscr{E}_2 \setminus (A_1 = \{a,b\} \cup \underline{B}_1 = \emptyset) = \check{\mathscr{E}}_2$, since $\widetilde{A}_1 = \{b\}$, due to $b \in A_1 \setminus F_2$, $a \notin \check{F}$, due to $(b,\underline{a}) \in \rhd_2$, and $\check{C}_0 = ((C_2^0 = \emptyset) \cup (A_1 = \{a,b\})) \cap (\check{E} = \{a\}) = \{a\} = \check{C}_0$;
- $\dot{\mathscr{E}}_2 = \mathscr{E}_2 \setminus (A_1 = \{b\} \cup \underline{B}_1 = \emptyset) = (\dot{E} = \{a\}, \dot{<} = \emptyset, \dot{\sharp} = \emptyset, \dot{l} = l_2|_{\{a\}}, \dot{F} = \emptyset, \dot{\prec} = \emptyset, \dot{\rhd} = \emptyset, \dot{C}_0 = \emptyset)$, because $\widetilde{A}_1 = \{b\}$, due to $b \in A_1 \setminus F_2$, $a \notin \dot{F}$, due to $(b,\underline{a}) \in \rhd_2$, and $\dot{C}_0 = ((C_2^0 = \emptyset) \cup (A_1 = \{b\})) \cap (\dot{E} = \{a\}) = \emptyset$;
- $\ddot{\mathscr{E}}_2 = \mathscr{E}_2 \setminus (A_1 = \{b\} \cup \underline{B}_1 = \emptyset)(A_2 = \{a\} \cup \underline{B}_2 = \emptyset) = (\ddot{E} = \emptyset, \ddot{<} = \emptyset, \ddot{\sharp} = \emptyset, \ddot{l} = \emptyset, \ddot{F} = \emptyset, \ddot{\prec} = \emptyset, \ddot{\rhd} = \emptyset, \ddot{C}_0 = \emptyset)$, since $\widetilde{A}_2 = \{a\}$, due to $a \in A_2 \setminus \dot{F}$, and $\ddot{C}_0 = ((\dot{C}_0 = \emptyset) \cup (A_2 = \{a\})) \cap (\dot{E} = \emptyset) = \emptyset$.

Notice that the removal operator produces the same residuals after the different traces. For example, it is easy to see that:

$\mathscr{E}_2 \setminus (\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}) = \mathscr{E}_2 \setminus ((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*$,
$\mathscr{E}_2 \setminus (\{a\}\cup\emptyset) = \mathscr{E}_2 \setminus ((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a\}\cup\emptyset)$,
$\mathscr{E}_2 \setminus (\{a\}\cup\emptyset)(\{b\}\cup\emptyset) = \mathscr{E}_2 \setminus ((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a\}\cup\emptyset)(\{b\}\cup\emptyset)$,
$\mathscr{E}_2 \setminus (\{a,b\}\cup\emptyset) = \mathscr{E}_2 \setminus ((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{a,b\}\cup\emptyset)$,
$\mathscr{E}_2 \setminus (\{b\}\cup\emptyset) = \mathscr{E}_2 \setminus ((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{b\}\cup\emptyset)$,
$\mathscr{E}_2 \setminus (\{b\}\cup\emptyset)(\{a\}\cup\emptyset) = \mathscr{E}_2 \setminus ((\{a\}\cup\emptyset)(\emptyset\cup\{\underline{a}\}))^*(\{b\}\cup\emptyset)(\{a\}\cup\emptyset)$. ◇

Below are some technical facts specific to the removal operator for RPESs.

Lemma 3. *Given a cause-respecting RPES $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$, a trace $t = (A_1 \cup \underline{B_1}) \dots (A_n \cup \underline{B_n})$ $(C_0 \overset{A_1\cup B_1}{\rightarrow} C_1 \dots C_{n-1} \overset{A_n\cup B_n}{\rightarrow} C_n)$ $(n \geq 0)$ of $\mathscr{E}$, and $\mathscr{E} \setminus t = (E^n, <^n, \sharp^n, l^n, F^n, \prec^n, \rhd^n, C_0^n)$, it holds:*

(i) $E^j \subseteq E^i$, $F^j \subseteq F^i$, $l^j \subseteq l^i$, $\nabla^j \subseteq \nabla^i$ $(\nabla \in \{<, \sharp, \prec, \rhd\})$, *for any* $0 \leq i \leq j \leq n$;

(ii) $\mathscr{E} \setminus t_i$ *is a cause-respecting RPES, for any* $0 \leq i \leq n$;

(iii) $B_i \subseteq F^{i-1}$, *for any* $1 \leq i \leq n$;

(iv) $A_i \subseteq E^{i-1}$, *for any* $1 \leq i \leq n$;

(v) $\widetilde{A}_i \subseteq C_n$, *for any* $1 \leq i \leq n$;

(vi) $C_0^n = C_n \cap E^n$.

The following two statements demonstrate compositional properties of the residual operator for cause-respecting RPESs.

Proposition 2. *Given a cause-respecting RPES $\mathscr{E}$ with a trace $t \in Traces(\mathscr{E})$ and its residual $\mathscr{E}' = \mathscr{E} \setminus t$ with a trace $t' \in Traces(\mathscr{E}')$, it holds that $tt' \in Traces(\mathscr{E})$, and, moreover, $\mathscr{E} \setminus tt' = \mathscr{E}' \setminus t'$.*

So, it turned out that the concatenation of any trace t of the cause-respecting RPES $\mathscr{E}$ and any trace t' of the residual $\mathscr{E} \setminus t$ is a trace of $\mathscr{E}$, and, moreover, the residuals $\mathscr{E} \setminus tt'$ and $\mathscr{E} \setminus t \setminus t'$ coincide.

Example 6. First, consider the non-cause-respecting $\mathscr{E}_0 = (E_0, <_0, \sharp_0, l_0, F_0, \prec_0, \rhd_0, C_0^0)$ from Examples 1–4, where $E_0 = \{a,b,c,d,e\}$; $<_0 = \{(b,d),(c,e)\}$; $\sharp_0 = \{(a,b),(b,a),(b,c),(c,b)\}$; l_0 is the identical function; $F_0 = \{b,c\}$; $\prec_0 = \{(b,\underline{b}),(c,\underline{c})\}$; $\rhd_0 = \emptyset$; $C_0^0 = \emptyset$. As was demonstrated in Example 2, the sequences $(\{b\} \cup \emptyset)$, $(\{b\} \cup \emptyset)(\{d\} \cup \emptyset)$ are traces of $\mathscr{E}_0$. Construct the following residuals of $\mathscr{E}_0$:

- $\dot{\mathscr{E}}_0 = \mathscr{E}_0 \setminus (A_1 = \{b\} \cup \underline{B}_1 = \emptyset) = (\dot{E}_0 = E_0, \dot{<}_0 = <_0, \dot{\sharp}_0 = \sharp_0, \dot{l}_0 = l_0, \dot{F}_0 = F_0, \dot{\prec}_0 = \prec_0, \dot{\rhd}_0 = \rhd_0, \dot{C}_0 = \{b\})$, because $(\widetilde{A}_1 \cup \sharp_0(\widetilde{A}_1)) = \emptyset$, due to $b \in F_0$, and, moreover, $\dot{C}_0 = ((C_0^0 = \emptyset) \cup (A_1 = \{b\})) \cap (\dot{E}_0 = \{a,b,c,d,e\}) = \{b\}$;
- $\ddot{\mathscr{E}}_0 = \mathscr{E}_0 \setminus (A_1 = \{b\} \cup \underline{B}_1 = \emptyset)(A_2 = \{d\} \cup \underline{B}_2 = \emptyset) = \ddot{E}_0 = \{e\}$, $\ddot{<}_0 = \emptyset$, $\ddot{\sharp}_0 = \emptyset$, $\ddot{l}_0 = \dot{l}_0|_{\{e\}}$, $\ddot{F}_0 = \emptyset$, $\ddot{\prec}_0 = \emptyset$, $\ddot{\rhd}_0 = \emptyset$, $\ddot{C}_0 = \emptyset$), because $\widetilde{A}_2 = \{b,d\}$ thanks to $d \in A_2 \setminus \dot{F}_0$, $(b,d) \in \dot{<}_0$ and $b \in \dot{F}_0$, and $\dot{\sharp}_0(\widetilde{A}_2) = \{a,c\}$, due to $(a,b),(b,c) \in \dot{\sharp}_0$, and, moreover, $\dot{C}_0 = ((\dot{C}_0 = \{b\}) \cup (A_2 = \{d\})) \cap (\dot{E}_0 = \{e\}) = \emptyset$.

It is easy to see that $(\{e\} \cup \emptyset)$ is a trace of $\ddot{\mathscr{E}}_0$, whereas the sequence $(\{b\} \cup \emptyset)(\{d\} \cup \emptyset)(\{e\} \cup \emptyset)$ is not a trace of $\mathscr{E}_0$.

Using Examples 2–5, it is not difficult to make sure that Proposition 2 holds for the cause-respecting RPES $\mathscr{E}_2$. ◇

It is stated below that any suffix t' of any trace tt' of the cause-respecting RPES $\mathscr{E}$ is a trace of the residual $\mathscr{E} \setminus t$.

Proposition 3. *Given a cause-respecting RPES $\mathscr{E}$ with traces $t', t't'' \in Traces(\mathscr{E})$, $t'' \in Traces(\mathscr{E} \setminus t')$ holds.*

Example 7. Examine the non-cause-respecting RPES $\mathscr{E}_1$ from Examples 2–4, with the components: $E_1 = \{a,b\}$; $<_1 = \{(a,b)\}$; $\sharp_1 = \emptyset$; l_1 is the identical function; $F_1 = \{a\}$; $\prec_1 = \{(a,\underline{a})\}$; $\rhd_1 = \emptyset$; $C_0^1 = \emptyset$. We know that $t' = (\{a\} \cup \emptyset)(\{b\} \cup \emptyset)$ and $t = (\{a\} \cup \emptyset)(\{b\} \cup \emptyset)(\emptyset \cup \{\underline{a}\})(\{a\} \cup \emptyset)$ are traces of $\mathscr{E}_1$. Let $t'' = (\emptyset \cup \{\underline{a}\})(\{a\} \cup \emptyset)$. Using Definition 5, we obtain the RPES $\mathscr{E}_1 \setminus t' = (E_1' = \emptyset, <_1' = \emptyset, \sharp_1' = \emptyset, l_1' = \emptyset, F_1' = \emptyset, \prec_1' = \emptyset, \rhd_1' = \emptyset, C_0'^1 = \emptyset)$. It is clear that $Traces(\mathscr{E}_1 \setminus t') = \emptyset$. Therefore, we get $t'' \notin Traces(\mathscr{E}_1 \setminus t')$.

Using Examples 2–5, it is not difficult to check that Proposition 3 holds for the cause-respecting RPES $\mathscr{E}_2$. ◇

4 Transition System Semantics for Cause-Respecting RPESs

In this section, we first give some basic definitions concerning labeled transition systems. Then, we define the mappings $TC(\mathscr{E})$ and $TR(\mathscr{E})$, which associate two distinct kinds of transition systems – one whose states are configurations and one whose states are residuals – with the RPES $\mathscr{E}$ labeled over the set L of actions.

A transition system $T = (S, \rightarrow, i)$ labeled over a set $\mathscr{L}$ of labels consists of a set of states S, a transition relation $\rightarrow \subseteq S \times \mathscr{L} \times S$, and an initial state $i \in S$. Two transition systems labeled over $\mathscr{L}$ are

isomorphic if their states can be mapped one-to-one to each other, preserving transitions and initial states. We call a relation $R \subseteq S \times S'$ a *bisimulation* between transition systems $T = (S, \rightarrow, i)$ and $T' = (S', \rightarrow', i')$ over $\mathscr{L}$ iff $(i, i') \in R$, and for all $(s, s') \in R$ and $l \in \mathscr{L}$: if $(s, l, s_1) \in \rightarrow$ then $(s', l, s'_1) \in \rightarrow'$ and $(s_1, s'_1) \in R$, for some $s'_1 \in S'$; and if $(s', l, s'_1) \in \rightarrow'$ then $(s, l, s_1) \in \rightarrow$ and $(s_1, s'_1) \in R$, for some $s_1 \in S$. Two transition systems over $\mathscr{L}$ are *bisimilar* if there is a bisimulation between them.

For a fixed set L of actions in RPESs, define the set $\mathbb{L} := \mathbb{N}_0^L$ (the set of multisets over L, or functions from L to the non-negative integers). The set $\mathbb{L}$ will be used as the set of labels in transition systems.

We are ready to define transition systems (labeled over $\mathbb{L}$) with configurations as states.

Definition 6. *For an RPES $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$ over L,*

$$TC(\mathscr{E}) \text{ is a transition system } (Conf(\mathscr{E}), \multimap, C_0) \text{ over } \mathbb{L},$$

where $C \xrightarrow{M} C'$ iff $C \overset{(A \cup \underline{B})}{\rightarrow} C'$ in $\mathscr{E}$ and $M = l(A \cup \underline{B})$[4].

Let us explain the above definition with

Example 8. Consider the cause-respecting RPES $\mathscr{E}_2$ from Examples 2–5. In Example 2, we can see that $C_0^2 = \emptyset$ and $Conf(\mathscr{E}_2) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$. Using Definition 6, we obtain $\multimap = \{(\emptyset, (\{a\} \cup \emptyset), \{a\}), (\{a\}, (\emptyset \cup \{a\}), \emptyset), (\{a\}, (\{b\} \cup \emptyset), \{a,b\}), (\emptyset, (\{b\} \cup \emptyset), \{b\}), (\{b\}, (\{a\} \cup \emptyset), \{a,b\}), (\emptyset, (\{a,b\} \cup \emptyset), \{a,b\})\}$. A graphical representation of the configuration transition system $TC(\mathscr{E}_2)$ is shown in Fig. 1. $\diamond$

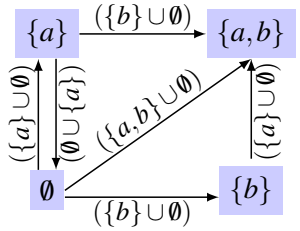

Figure 1: The configuration transition system $TC(\mathscr{E}_2)$

We next propose the definition of labeled transition systems over $\mathbb{L}$ with RPESs as states.

Definition 7. *For an RPES $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$ over L,*

$$TR(\mathscr{E}) \text{ is a transition system } (Reach(\mathscr{E}), \rightharpoonup, \mathscr{E}) \text{ over } \mathbb{L},$$

where $\mathscr{F} \overset{M}{\rightharpoonup} \mathscr{F}'$ iff $\mathscr{F}' = \mathscr{F} \setminus (A \cup \underline{B})$ and $M = l(A \cup \underline{B})$, and $Reach(\mathscr{E}) = \{\mathscr{F} \mid \exists \mathscr{E}_0, \ldots, \mathscr{E}_k \ (k \geq 0)$ s.t. $\mathscr{E}_0 = \mathscr{E} \setminus \varepsilon$, $\mathscr{E}_k = \mathscr{F}$, and $\mathscr{E}_i \overset{l(A \cup \underline{B})}{\rightharpoonup} \mathscr{E}_{i+1} \ (0 \leq i < k)\}$.

We illustrate the above definition with

Example 9. Consider the RPES $\mathscr{E}_2$ from Examples 2–5. Using Definitions 5 and 7, we construct the residual transition system $TR(\mathscr{E}_2)$ which is depicted in Fig. 2. It is easy to check that the configuration transition system $TC(\mathscr{E}_2)$ (see Fig. 1) and the residual transition system $TR(\mathscr{E}_2)$ are bisimilar but not isomorphic. $\diamond$

We establish the relationships between the states and transitions of the configuration-based and residual-based transition systems of the RPES.

[4] See Definition 3.

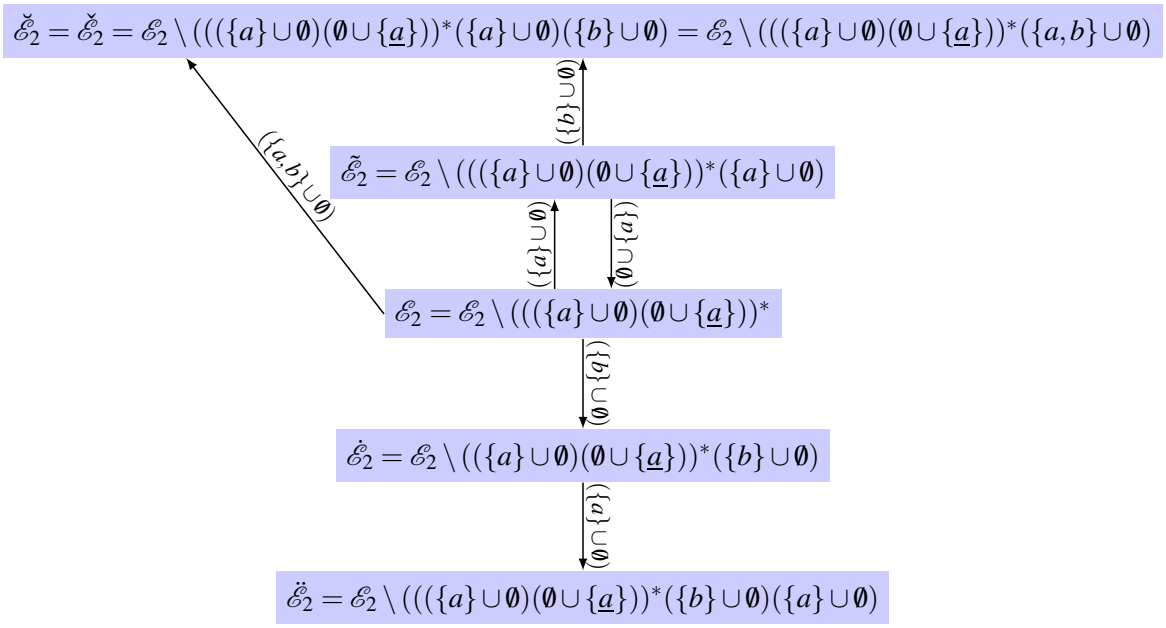

Figure 2: The residual transition system $TR(\mathscr{E}_2)$

Proposition 4. *Given a cause-respecting RPES $\mathscr{E} = (E, <, \sharp, l, F, \prec, \rhd, C_0)$ over L,*

(i) for any $last(t) \in Conf(\mathscr{E})$, $\mathscr{E} \setminus t \in Reach(\mathscr{E})$;

(ii) for any $\mathscr{E}' \in Reach(\mathscr{E})$, there is $last(t) \in Conf(\mathscr{E})$ such that $\mathscr{E}' = \mathscr{E} \setminus t$;

(iii) for any $last(t)$, $last(t') \in Conf(\mathscr{E})$, if $last(t) \overset{l(A \cup \underline{B})}{\rightharpoonup} last(t')$ then $\mathscr{E} \setminus t \overset{l(A \cup \underline{B})}{\rightharpoonup} \mathscr{E} \setminus t(A \cup \underline{B})$ and $last(t(A \cup \underline{B})) = last(t')$;

(iv) for any $\mathscr{E}', \mathscr{E}'' \in Reach(\mathscr{E})$, if $\mathscr{E}' \overset{l(A \cup \underline{B})}{\rightharpoonup} \mathscr{E}''$ then, for any $last(t) \in Conf(\mathscr{E})$ such that $\mathscr{E}' = \mathscr{E} \setminus t$, there is $last(t') \in Conf(\mathscr{E})$ such that $\mathscr{E}'' = \mathscr{E} \setminus t'$ and $last(t) \overset{l(A \cup \underline{B})}{\rightharpoonup} last(t')$.

Theorem 1. *Given a cause-respecting RPES $\mathscr{E}$ over L, $TC(\mathscr{E})$ and $TR(\mathscr{E})$ are bisimilar and in general not isomorphic.*

Proof. From Example 9 we know that, for the cause-respecting RPES $\mathscr{E}_2$, $TC(\mathscr{E}_2)$ and $TR(\mathscr{E}_2)$ are not isomorphic.

We shall check that $TC(\mathscr{E})$ and $TR(\mathscr{E})$ are bisimilar for an arbitrary cause-respecting RPES $\mathscr{E} = (E, <, \sharp, L, l, F, \prec, \rhd, C_0)$. Due to Lemma 1(i) and Propositions 4(i), we can define a relation $R \subseteq Conf(\mathscr{E}) \times Reach(\mathscr{E})$ as follows: $R = \{(last(t), \mathscr{E} \setminus t) \mid t \in Traces(\mathscr{E})\}$.

We need to show that R is a bisimulation between $TC(\mathscr{E})$ and $TR(\mathscr{E})$. Clearly, we have that $\varepsilon \in Traces(\mathscr{E})$, and, moreover, $C_0 = last(\varepsilon) \in Conf(\mathscr{E})$ and $\mathscr{E} = \mathscr{E} \setminus \varepsilon \in Reach(\mathscr{E})$. So, $(C_0, \mathscr{E}) \in R$ holds. Take an arbitrary $(last(t), \mathscr{E} \setminus t) \in R$. Suppose that $last(t) \overset{l(A \cup \underline{B})}{\rightharpoonup} C'$ in $TC(\mathscr{E})$ for some $C' \in Conf(\mathscr{E})$. By Lemma 1(i), there is $t' \in Traces(\mathscr{E})$ such that $C' = last(t')$. According to Proposition 4(iii), it is true that $\mathscr{E} \setminus t \overset{l(A \cup \underline{B})}{\rightharpoonup} \mathscr{E} \setminus t(A \cup \underline{B})$ and $last(t(A \cup \underline{B})) = last(t')$. Thanks to Lemma 1(i), we have $t(A \cup \underline{B}) \in Traces(\mathscr{E})$. Hence, $(C' = last(t(A \cup \underline{B})), \mathscr{E} \setminus t(A \cup \underline{B})) \in R$ holds. In the opposite direction, assume that

$\mathscr{E} \setminus t \stackrel{l(A \cup \underline{B})}{\longrightarrow} \mathscr{E}'$ in $TR(\mathscr{E})$ for some $\mathscr{E}' \in Reach(\mathscr{E})$. Due to Propositions 4(iv), for $last(t) \in Conf(\mathscr{E})$, there is $last(t') \in Conf(\mathscr{E})$ such that $\mathscr{E}' = \mathscr{E} \setminus t'$ and $last(t) \stackrel{l(A \cup \underline{B})}{\longrightarrow} last(t')$. Due to Lemma 1(i), $t' \in Traces(\mathscr{E})$ is true. This implies that $(last(t'), \mathscr{E} \setminus t' = \mathscr{E}') \in R$ holds. Hence, R is indeed a bisimulation. □

5 Concluding Remarks

In this paper, we dealt with two different – configuration-based and residual-based – ways of giving (step) transition system semantics for cause-respecting reversible prime event structures which encompass prime event structures. For this purpose, we firstly defined (step) semantics from [27], which is based on configurations/traces obtained by starting with the initial configuration and by executing events and/or undoing previously executed events, and, secondly, developed a removal operator which is useful for constructing residuals (model fragments) by retaining an appropriate amount of structure during the execution of the model. We also stated some correctness criteria for the removal operator. The meaning of the correctness properties is that the obtained residuals do not allow configurations/traces that are disallowed by the original structure. Also, in some sense, this signifies some compositionality properties of the removal operator. It turned out that in the context of PESs, the removal operator developed here produces the same residuals as the removal operator proposed in [22]. As our main result, we have obtained a (step) bisimulation between configuration-based and residual-based transition systems of the models under consideration. The configuration-based method discussed here can be useful in analyzing the state space of reversible concurrent systems whose behavior is represented as RPESs, and the proposed residual-based method can be suitable for specification and visualization of changes in the structures of reversible concurrent processes during their simulation in tools. Due to the good compositionality properties of the residual-based transition systems of RPESs and their complementarity and consistency with the configuration-based ones, it is hoped that the results obtained here may be helpful in demonstrating the correspondence between operational and denotational semantics of algebraic calculi of reversible concurrent processes, similar to how the results from [5, 9, 17] have found their application in traditional (irreversible) process algebras.

As for future work, we plan to broaden the list of studied models by adding flow/bundle/general event structures with symmetric and asymmetric conflict. Work on extending our approach to out-of-causal reversible prime event structures is under way and has yielded promising intermediate results. Another future line of our research is to generalize the model of reversible prime event structures with non-executable (impossible) events (for example, by dropping the transitivity/acyclicity of causality, as well as the principles of finite causes) in order to obtain isomorphisms between the two types of transition systems of the models, as was done for the corresponding extension of PESs in the paper [8]. There, the authors have been able to argue that non-executable events are useful in comparative semantics, facilitating the elimination of non-fundamental inconsistencies between models. Furthermore, isomorphisms between the transition system semantics are expected to allow one to relate those constructed on configurations and those derived from denotational semantics of process calculi in a tight way.

References

[1] Bogdan Aman & Gabriel Ciobanu (2018): *Controlled Reversibility in Reaction Systems*. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa & Claudio Zandron, editors: *Membrane Computing*, Springer International Publishing, Cham, pp. 40–53, doi:10.1007/978-3-319-73359-3_3.

[2] Youssef Arbach, David Karcher, Kirstin Peters & Uwe Nestmann (2015): *Dynamic Causality in Event Structures*. In Susanne Graf & Mahesh Viswanathan, editors: *Formal Techniques for Distributed Objects, Components, and Systems*, Springer International Publishing, Cham, pp. 83–97, doi:10.1007/978-3-319-19195-9_6.

[3] Abel Armas-Cervantes, Paolo Baldan & Luciano Garcia-Banuelos (2016): *Reduction of event structures under history preserving bisimulation*. *Journal of Logical and Algebraic Methods in Programming* 85(6), pp. 1110–1130, doi:10.1016/j.jlamp.2015.10.004.

[4] Clement Aubert & Ioana Cristescu (2017): *Contextual equivalences in configuration structures and reversibility*. *Journal of Logical and Algebraic Methods in Programming* 86(1), pp. 77–106, doi:10.1016/j.jlamp.2016.08.004.

[5] Christel Baier & Mila Majster-Cederbaum (1994): *The connection between an event structure semantics and an operational semantics forTCSP*. *Acta Informatica* 31(1), doi:10.1007/BF01178923.

[6] Kamila Barylska, Anna Gogolinska, Lukasz Mikulski, Anna Philippou, Marcin Piatkowski & Kyriaki Psara (2022): *Formal Translation from Reversing Petri Nets to Coloured Petri Nets*. In Claudio Antares Mezzina & Krzysztof Podlaski, editors: *Reversible Computation*, Springer International Publishing, Cham, pp. 172–186, doi:10.1007/978-3-031-09005-9_12.

[7] Eike Best, Nataliya Gribovskaya & Irina Virbitskaite (2017): *Configuration- and Residual-Based Transition Systems for Event Structures with Asymmetric Conflict*. In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey & Tiziana Margaria, editors: *SOFSEM 2017: Theory and Practice of Computer Science*, Springer International Publishing, Cham, pp. 132–146, doi:10.1007/978-3-319-51963-0_11.

[8] Eike Best, Nataliya Gribovskaya & Irina Virbitskaite (2018): *From Event-Oriented Models to Transition Systems*. In Victor Khomenko & Olivier H. Roux, editors: *Application and Theory of Petri Nets and Concurrency*, Springer International Publishing, Cham, pp. 117–139, doi:10.1007/978-3-319-91268-4_7.

[9] Gérard Boudol (1990): *Flow event structures and flow nets*. In Irène Guessarian, editor: *Semantics of Systems of Concurrent Processes*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 62–95, doi:10.1007/3-540-53479-2_4.

[10] Silvia Crafa, Daniele Varacca & Nobuko Yoshida (2012): *Event Structure Semantics of Parallel Extrusion in the π-Calculus*. In Lars Birkedal, editor: *Foundations of Software Science and Computational Structures*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 225–239, doi:10.1007/978-3-642-28729-9_15.

[11] Vincent Danos & Jean Krivine (2005): *Transactions in RCCS*. In Martín Abadi & Luca de Alfaro, editors: *CONCUR 2005 – Concurrency Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 398–412, doi:10.1007/11539452_31.

[12] Alexis De Vos, Stijn De Baerdemacker & Yvan Van Rentergem (2018): *Synthesis of Quantum Circuits vs. Synthesis of Classical Reversible Circuits*, first edition. Springer Cham, doi:10.1007/978-3-031-79895-5.

[13] David de Frutos Escrig, Maciej Koutny & Łukasz Mikulski (2019): *Reversing Steps in Petri Nets*. In Susanna Donatelli & Stefan Haar, editors: *Application and Theory of Petri Nets and Concurrency*, Springer International Publishing, Cham, pp. 171–191, doi:10.1007/978-3-030-21571-2_11.

[14] Eva Graversen, Iain Phillips & Nobuko Yoshida (2021): *Event structure semantics of (controlled) reversible CCS*. *Journal of Logical and Algebraic Methods in Programming* 121, p. 100686, doi:10.1016/j.jlamp.2021.100686.

[15] P.W. Hoogers, H.C.M. Kleijn & P.S. Thiagarajan (1996): *An event structure semantics for general Petri nets*. *Theoretical Computer Science* 153(1), pp. 129–170, doi:10.1016/0304-3975(95)00120-4.

[16] Jarkko Kari (2018): *Reversible Cellular Automata: From Fundamental Classical Results to Recent Developments*. *New Generation Computing* 36(3), pp. 145–172, doi:10.1007/s00354-018-0034-6.

[17] Joost-Pieter Katoen (1996): *Quantitative and Qualitative Extensions of Event Structures*. Ph.D. thesis, University of Twente, Netherlands.

[18] Stefan Kuhn, Bogdan Aman, Gabriel Ciobanu, Anna Philippou, Kyriaki Psara & Irek Ulidowski (2020): *Reversibility in Chemical Reactions*, pp. 151–176. Springer International Publishing, Cham, doi:10.1007/978-3-030-47361-7_7.

[19] Ivan Lanese, Claudio Antares Mezzina & Jean-Bernard Stefani (2016): *Reversibility in the higher-order π-calculus*. *Theoretical Computer Science* 625, pp. 25–84, doi:10.1016/j.tcs.2016.02.019.

[20] Ivan Lanese, Adrián Palacios & Germán Vidal (2019): *Causal-Consistent Replay Debugging for Message Passing Programs*. In Jorge A. Pérez & Nobuko Yoshida, editors: *Formal Techniques for Distributed Objects, Components, and Systems*, Springer International Publishing, Cham, pp. 167–184, doi:10.1007/978-3-030-21759-4_10.

[21] Rom Langerak (1991): *Bundle event structures: a non-interleaving semantics for LOTOS*. In Michel Diaz & Roland Groz, editors: *Formal Description Techniques V*, IFIP transactions C, Communication systems, North Holland, Netherlands, pp. 331–346. 5th International Conference on Formal Description Techniques for Distributed Systems and Communications Protocols, FORTE 1992.

[22] Mila Majster-Cederbaum & Markus Roggenbach (1998): *Transition systems from event structures revisited*. *Information Processing Letters* 67(3), pp. 119–124, doi:10.1016/S0020-0190(98)00105-7.

[23] Doriana Medić, Claudio Antares Mezzina, Iain Phillips & Nobuko Yoshida (2020): *Towards a Formal Account for Software Transactional Memory*. In Ivan Lanese & Mariusz Rawski, editors: *Reversible Computation*, Springer International Publishing, Cham, pp. 255–263, doi:10.1007/978-3-030-52482-1_16.

[24] Hernan Melgratti, Claudio Antares Mezzina & G. Michele Pinna (2021): *A distributed operational view of Reversible Prime Event Structures*. In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–13, doi:10.1109/LICS52264.2021.9470623.

[25] Hernan Melgratti, Claudio Antares Mezzina, Iain Phillips, G. Michele Pinna & Irek Ulidowski (2020): *Reversible Occurrence Nets and Causal Reversible Prime Event Structures*. In Ivan Lanese & Mariusz Rawski, editors: *Reversible Computation*, Springer International Publishing, Cham, pp. 35–53, doi:10.1007/978-3-030-52482-1_2.

[26] Anna Philippou & Kyriaki Psara (2020): *Reversible Computation in Cyclic Petri Nets*. *CoRR* abs/2010.04000, doi:10.48550/arXiv.2010.04000.

[27] Iain Phillips & Irek Ulidowski (2015): *Reversibility and asymmetric conflict in event structures*. *Journal of Logical and Algebraic Methods in Programming* 84(6), pp. 781–805, doi:10.1016/j.jlamp.2015.07.004.

[28] Iain Phillips, Irek Ulidowski & Shoji Yuen (2013): *A Reversible Process Calculus and the Modelling of the ERK Signalling Pathway*. In Robert Glück & Tetsuo Yokoyama, editors: *Reversible Computation*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 218–232, doi:10.1007/978-3-642-36315-3_18.

[29] Giovanni Michele Pinna (2017): *Reversing steps in membrane systems computations*. In M. Gheorghe, G. Rozenberg, A. Salomaa & C. Zandron, editors: *Membrane Computing*, 10725, Springer International Publishing, Cham, pp. 245–261, doi:10.1007/978-3-319-73359-3_16.

[30] Irek Ulidowski, Iain Phillips & Shoji Yuen (2018): *Reversing event structures*. *New Generation Computing* 36(3), pp. 281–306, doi:10.1007/s00354-018-0040-8.

[31] R.J. van Glabbeek & G.D. Plotkin (2009): *Configuration structures, event structures and Petri nets*. *Theoretical Computer Science* 410(41), pp. 4111–4159, doi:10.1016/j.tcs.2009.06.014.

[32] Glynn Winskel (1980): *Events in computation*. Ph.D. thesis, University of Edinburgh.

[33] Glynn Winskel (1989): *An introduction to event structures*. In J. W. de Bakker, W. P. de Roever & G. Rozenberg, editors: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 364–397, doi:10.1007/BFb0013026.