Concurrency and Reversibility

Irek Ulidowski[1], Iain Phillips[2], and Shoji Yuen[3]

[1] Department of Computer Science, University of Leicester, England
[2] Department of Computing, Imperial College London, England
[3] Graduate School of Information Science, Nagoya University, Japan

Abstract. Reversible computation has attracted increasing interest in recent years, with applications in hardware, software and biochemistry. In this paper we show how to model reversibility in concurrent computation as realised abstractly in terms of event structures. Two different forms of event structures are presented and it is shown how to extend them with reversibility.

1 Introduction

Reversing computation in concurrent and distributed systems has many promising applications as well as technical and conceptual challenges. Several different forms of undoing of computation have been identified recently. *Backtracking* and reversing of computation that preserves *causal order* were considered in, for example, [8, 16, 13, 4, 14, 5, 9] with applications including recovery-oriented systems and reversible debugging. Reversing *out of causal order*, however, which is a very common mode of operation in, for example, biochemical systems has not been studied widely. The first attempt was made by Phillips, Ulidowski and Yuen [21] where an extension of the reversible process calculus CCSK with the execution control operator was proposed. This was followed by a study of a form of reversible event structure [22] based on a generalisation of Winskel's enabling relation [26]. Phillips and Ulidowski proposed then in [19] reversible event structures that focused on analysing conflict and causation as first-class notions in the setting of reversible computation.

The last decade has produced a good understanding of how causal reversibility can be described in the settings of operational semantics and process calculi, and how to model reversibility logically and in terms of behavioural equivalences. Research on reversing process calculi can be traced back perhaps to Berry and Boudol's Chemical Abstract Machine [3]. Danos and Krivine reversed CCS in [6, 7], and Phillips and Ulidowski proposed a general method for reversing process calculi in [16, 17]. Reversible structures that compute forwards and backwards asynchronously were developed by Cardelli and Laneve [4]. Mechanisms for controlling reversibility based on a rollback construct were devised by Lanese, Mezzina, Schmitt and Stefani [12] for a reversible higher-order π calculus [13], and an alternative mechanism based on the execution control operator was proposed in [21]. Event Identifier Logic (EIL), which extends Hennessy-Milner logic [11] with reverse modalities, was introduced in [20]. EIL corresponds to hereditary history-preserving bisimulation equivalence [2] within a particular true-concurrency model

S. Yamashita and S. Minato (Eds.): RC 2014, LNCS 8507, pp. 1–14, 2014.

of stable configuration structures [10]. Moreover, natural sublogics of EIL correspond to coarser equivalences, several of them defined in terms of reversible events, sets of concurrent reversible events or pomsets of reversible events. These equivalences and other behavioural equivalences based in the reversible setting were studied for the first time in [18].

In this paper we show how to understand and model reversibility in concurrent computation as realised abstractly in terms of event structures. In Section 2 we introduce the notions of events, configurations, computation and configuration systems. Then in Sections 3 and 4, we recall two different forms of event structures and show how to extend them with reversibility. Numerous examples are used to illustrate our approach. The last section contains conclusions and lists some future challenges.

2 Events and Configurations

We represent the behaviour of systems and processes in the setting of event structures where units of behaviour are modelled by *events*. Since we aim to cover a wide range of systems and processes, events will represent activities such as incrementing the value of a variable, sending a message, as well as entering a room, putting a coin into a vending machine, or creating a bond between two molecules. Events have names and we assume that no two different events have the same name. We shall use a, b, c, d, e, f to denote events. A system or a process is then represented as an *event structure* which is a set of events and a number of relations on events. Event structures were defined by Winskel [26] following earlier work by Nielsen, Plotkin and Winskel [15]. They were further developed in, for example, [24, 23, 27] and [25]. There are many ways in which events can be related, and this determines how events are performed or undone. For example, a number of events can *cause* each other thus occurring in a sequence. Also, events can be *independent* from each other, or some events may be in *conflict* with other events. Alternatively, an *enabling* relation on events is used.

Event structures compute (or execute) by either performing events or undoing events, thus moving from one state to another state. A state is simply a set of events that have occurred and have not been undone yet, and is called a *configuration*. The act of moving from a configuration to another configuration is a computation step and is represented by a transition relation: $C \rightarrow C'$ means that configuration C evolves to configuration C' by performing and/or undoing some events. A sequence of computation steps is called an execution (or computation). For example, the execution $\varnothing \rightarrow \{a\} \rightarrow \{a, b\}$ says that, initially, no event has occurred, then event a takes place, and finally event b occurs. We note that any initial subsequence of an execution is also an execution. Events can also be undone. We take the view that undoing an event e means that e is removed from the current configuration, and it is as if e had never occurred, apart possibly from indirect effects, such as e having caused another event f before e was reversed. When we undo e in configuration $\{e, f\}$ we regress to $\{f\}$: this is often written as $\{e, f\} \dashrightarrow \{f\}$ instead of $\{e, f\} \rightarrow \{f\}$ to indicate that

an event is undone and to match the notation used in figures. The computation of event structures is thus represented by a *configuration system*. Configuration systems are closely related to *configuration structures*, which have a notion of *configuration* and a notion of concurrent or *step* transition. These were introduced by van Glabbeek and Goltz in [10] and later generalised by van Glabbeek and Plotkin in [23]. Let $\mathfrak{P}(E)$ denote the powerset of a set E. A configuration structure is a pair $\mathcal{C} = (E, \mathsf{C})$ where E is a set of events and $\mathsf{C} \subseteq \mathfrak{P}(E)$ is a set of configurations. For configurations X, Y, we let $X \to Y$ if $X \subseteq Y$ and for every Z, if $X \subseteq Z \subseteq Y$ then Z is a configuration. Since all the events in $Y \setminus X$ are independent, they can happen concurrently as a single step.

We sometime write $X \xrightarrow{A} Y$ where $A = Y \setminus X$ instead of $X \to Y$. Note that if $Y = X \cup \{a\}$ and $X, Y \in \mathsf{C}$ then $X \to Y$. This may no longer hold in the reversible setting. Consider $E = \{a, b\}$. Suppose that a causes b, so that b cannot occur unless a has already occurred. Then $\{b\}$ is not a possible configuration using forwards computation. However, if a is reversible, we can do a (namely, $\varnothing \to \{a\}$) followed by b ($\{a\} \to \{a, b\}$), followed by reversing a ($\{a, b\} \dashrightarrow \{b\}$) to reach $\{b\}$. Thus both $\varnothing$ and $\{b\}$ are configurations, but we do not have $\varnothing \xrightarrow{b} \{b\}$.

A definition of configuration systems appropriate for the reversible setting was first given in [19]. We first establish our notation before we recall the definition. We let $A, B, X, Y, Z, \ldots$ range over sets of events. If an event e is reversible, we have a corresponding reverse event $\underline{e}$. We write $\underline{B}$ for $\{\underline{e} : e \in B\}$.

Definition 2.1. *A* configuration system *is a quadruple $\mathcal{C} = (E, F, \mathsf{C}, \to)$ where E is a set of events, $F \subseteq E$ are the reversible events, $\mathsf{C} \subseteq \mathfrak{P}(E)$ is the set of* configurations *and $\to \subseteq \mathsf{C} \times \mathfrak{P}(E \cup \underline{F}) \times \mathsf{C}$ is a labelled transition relation such that if $X \xrightarrow{A \cup \underline{B}} Y$ then:*

- *$A \cap X = \varnothing$ and $B \subseteq X \cap F$ and $Y = (X \setminus B) \cup A$;*
- *$X \xrightarrow{A' \cup \underline{B'}} Z \xrightarrow{(A \setminus A') \cup (\underline{B \setminus B'})} Y$ (where $Z = (X \setminus B') \cup A' \in \mathsf{C}$) for every $A' \subseteq A$ and $B' \subseteq B$.*

We say that $A \cup \underline{B}$ is enabled *at X if there is Y such that $X \xrightarrow{A \cup \underline{B}} Y$. A transition $X \xrightarrow{A \cup \underline{B}} Y$ is* mixed *if both A and B are non-empty. If $B = \varnothing$ we say the transition is* forwards*, and if $A = \varnothing$ the transition is* reverse*.*

For simplicity, we do not discuss in depth mixed transitions in this paper. Most examples concern transitions where A and $\underline{B}$ are singleton sets. As a result, the transitions denote either performing an event or undoing an event.

Finally, we define reachable configurations. Let $\mathcal{C} = (E, F, \mathsf{C}, \to)$ be a configuration system. We say that configuration X is a *reachable* configuration if $\varnothing \xrightarrow{A_1 \cup \underline{B}_1} \cdots \xrightarrow{A_n \cup \underline{B}_n} X$ where $A_i \subseteq E$ and $B_i \subseteq F$ for each $i = 1, \ldots, n$.

3 Reversible Event Structures with Causality and Precedence

In this section we consider event structures where the causation, concurrency and precedence relations on events dictate how they compute.

In order to explore different forms of relations between events and how this impacts on performing and undoing of events, we shall consider mostly very small event structures, namely those that have three events a, b and c. Even in such a simple setting we will be able to describe most of the important forms of executing events forwards and in reverse. The events a, b, c are depicted by the three dimensions of the cube in Figure 1. Note that any of the four edges of any dimension (representing an event) denotes an *occurrence* of the event. The bottom-left vertex represents the empty configuration $\varnothing$ (the origin of computation) and the top-right vertex represents the configuration $\{a, b, c\}$. If there are no constraints the events can happen in any order, denoted by following the edges from the origin, or simultaneously, denoted by taking some diagonals in the cube. For simplicity we do not display transitions of simultaneous events in our figures.

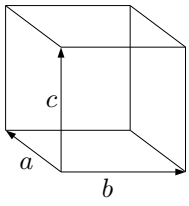

Fig. 1.

Causality is a binary irreflexive relation on events. It tells us which events cause which other events. We write $a \prec b$ to mean a causes b, so b cannot take place before a has occurred.

If $a \prec b$ and $b \prec c$ and $a \prec c$, namely $\prec$ is transitive, then we know that an execution contains b if it contains a, and c is in an execution if b is. The execution is $\varnothing \to \{a\} \to \{a, b\} \to \{a, b, c\}$. This is depicted in the left cube in Figure 2 by the sequence of thick arrows. An alternative way to represent an execution is with a sequence of events, for example, abc is the execution of the system with $a \prec b$, $b \prec c$ and $a \prec c$. And we can also write $\varnothing \xrightarrow{a} \{a\} \xrightarrow{b} \{a, b\} \xrightarrow{c} \{a, b, c\}$.

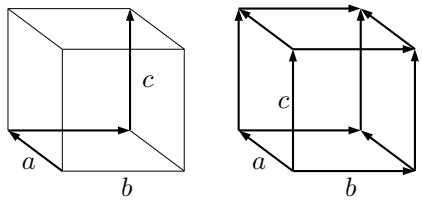

Fig. 2.

The cube on the right in Figure 2 shows all possible executions when a, b and c are *independent*, except those executions that involve steps (sets of simultaneous events) which we do not display for clarity. If events are not related by causality or other relations, then they are independent. This means that the events can take place in any order; hence six complete executions abc, acb, bac, bca, cab and cba are depicted. Each square of thick arrows represents graphically the independence of the events; we see that the events can happen in any order so we call them *concurrent* events. We have step transitions here, for example $\varnothing \to \{a, b, c\}$ and $\{a\} \to \{a, b, c\}$, but we do not display them in Figure 2. Also, there are several mixed transitions, for example, performing b and undoing a from $\{a\}$ is represented by $\{a\} \to \{b\}$, or $\{a\} \xrightarrow{b, \underline{a}} \{b\}$.

If $a \prec b$ and $a \prec c$, meaning that a causes both b and c, and b, c are independent, then there are only two complete executions: abc and acb. Correspondingly, $a \prec c$ and $b \prec c$ (namely, both a and b cause c) results in abc and bac.

So far we have illustrated how causality or independence (concurrency) affects the execution. Another very useful relation on events is *precedence*: $a \lhd b$,

read as event a *precedes* event b, means that if both a and b occur then a occurs first. The precedence relation has a dual interpretation: $b \rhd a$ says that b *prevents* a, meaning that if b is present in a configuration, then a cannot occur. Precedence is a form of *asymmetric conflict* [1]. Consider a system where $a \lhd b$ and $b \lhd a$, meaning that once either a or b occurs the other event cannot occur afterwards. In other words, a and b are in *conflict*, often denoted as $a \sharp b$. If, additionally, $a \prec c$, then we have two complete executions ac and b depicted by the left cube in Figure 3. We can use the precedence relation to disable events. For example, if $b \lhd b$, then b can never occur, and if we also have $a \lhd c$ then ac and c are the only complete executions (see the cube on the right in Figure 3).

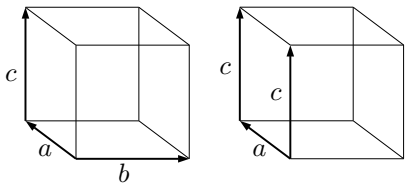

Fig. 3.

There are other forms of execution of the three events a, b, c which cannot be achieved by any combination of the causation, concurrency and precedence relations: we discuss this in the next section.

Next, we recall three forms of undoing events. *Backtracking* is when events are undone in the inverse order they occurred. The system $a \prec b \prec c$ in configuration $\{a, b, c\}$ backtracks by undoing c first, then undoing b and, lastly, undoing a. The left cube in Figure 4 shows the system backtracking c and then b (dashed arrows pointing in the opposite direction) from $\{a, b, c\}$, which is written as $\{a, b, c\}$ -⇢ $\{a, b\}$ -⇢ $\{a\}$.

Consider $a \prec b$ and $a \prec c$: a occurs first and then b, c can occur independently. Once in the configuration $\{a, b, c\}$ we have no way of working out which of b and c occurred last. Since the events are independent, the order of performing or undoing them does not matter. So *causal reversing*, or simply reversing, is undoing where (a) independent events can be undone in any order irrespective of the order they have actually occurred, and (b) events that cause other events can only be undone after the caused events are undone first.

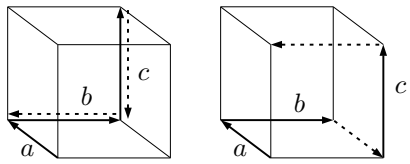

Fig. 4.

Both backtracking and reversing are *cause-respecting*, meaning that events caused by other events are undone first before the other events can be undone. There are, however, many important examples of undoing things *out-of-causal order*. In fact, this form of undoing plays the vital rôle the mechanisms driving long-running transactions and biochemical reactions. As an example, consider the following pattern of behaviour shown in the right cube of Figure 4. Event a causes event b. Once we have b, a is not needed so it is undone. Finally, c occurs and cannot be undone, and then b is undone. Informally speaking, a can

be thought as the *catalyst* of b, and b as the catalyst of c. The execution is $\varnothing \to \{a\} \to \{a,b\} \dashrightarrow \{b\} \to \{b,c\} \dashrightarrow \{c\}$. Note that we undo a, the cause of b, before we undo b. Overall, we can reach $\{c\}$ from $\varnothing$ via a combination of forwards and reverse moves but we cannot reach $\{c\}$ by executing forwards only.

Since there are different forms of undoing events, the question is how to model undoing of events formally. In [19] we extend the causation and precedence relations to define additionally undoing of events. Recall that $\underline{a}, \underline{b}, \underline{c}$ denote undoing of a, b, c. We can extend the causation relation $\prec$ with pairs $x \prec \underline{y}$, meaning that event y can be undone if y has occurred and x has occurred and has not been undone yet. For example, $a \prec \underline{a}$ means that a can be undone if it has occurred. Correspondingly, we also extend our precedence relation $\lhd$ with pairs $\underline{x} \lhd y$, meaning that x cannot be undone if y is present. What we have described informally so far are *reversible asymmetric event structures* ([19]):

Definition 3.1. A reversible asymmetric event structure (RAES) is a quadruple $\mathcal{E} = (E, F, \prec, \lhd)$ where E is a set of events and $F \subseteq E$ are those events of E which are reversible, and for any $a, b, c, e \in E$ and $\alpha \in E \cup \underline{F}$:

1. $\lhd \subseteq (E \cup \underline{F}) \times E$ is the *precedence* relation (with $a \lhd b$ if and only if $b \rhd a$), which is irreflexive;
2. $\prec \subseteq E \times (E \cup \underline{F})$ is the *direct causation* relation, which is irreflexive and well-founded, and such that $\{e \in E : e \prec \alpha\}$ is finite and $\lhd$ is acyclic on $\{e \in E : e \prec \alpha\}$;
3. $a \prec \underline{a}$ for all $a \in F$;
4. if $a \prec \alpha$ then not $a \rhd \alpha$;
5. $a \ll b$ implies $a \lhd b$, where *sustained direct causation* $a \ll b$ means that $a \prec b$ and if $a \in F$ then $b \rhd \underline{a}$;
6. $\ll$ is transitive;
7. if $a \sharp c$ and $a \ll b$ then $b \sharp c$, where $\sharp$ is defined to be $\lhd \cap \rhd$.

Causation can be explained in two different ways. Event a causes event b ($a \prec b$) means either (1) in any execution (computation), if b occurs then a occurs earlier or (2) if b is enabled at configuration X then we must have $a \in X$. The two views are equivalent if there is no reversing. Consider three events with $a \prec b \prec c$. Taking view (1) we deduce that $a \prec c$. View (2) also allows us to deduce that $a \prec c$, provided that X is left-closed (downwards closed under $\prec$), which will be the case for forward-only computation. Thus causation is transitive.

In the setting of reversible computation the second view of causation is simpler, and is adopted in this paper. If all reversing is causal, then all configurations are left-closed, and so it is still natural to require $\prec$ to be transitive. If, however, there is non-causal reversing, which leads to non-left-closed configurations (such as $\{b,c\}$ and $\{c\}$ in our example), it is no longer reasonable to insist on $\prec$ being transitive. If $a \prec b \prec c$ then a may have been reversed after b occurs, and before c occurs. Therefore, direct causation in RAESs is non-transitive. We introduce additionally the concept of sustained causation, where $a \ll b$ means that a causes b and a cannot reverse until b reverses. This is the analogue of standard causation for forwards computation, and we therefore take sustained causation to be transitive (condition 6 in Definition 3.1).

Next we consider the issue of *conflict inheritance*, namely if $a \prec b$ and $a \sharp c$ then $b \sharp c$, in the reversible setting. If $a \prec b$ and $a \sharp c$ and a is reversible, then we can undo a in $\{a, b\}$ to reach $\{b\}$. And there is nothing in $\{b\}$ to prevent c from taking place, so we expect that $\{b, c\}$ is a configuration, and b and c are not in conflict. Hence, there is no conflict inheritance with respect to $\prec$. However, we still have conflict inheritance with respect to sustained causation $a \ll b$ (condition 7 in Definition 3.1).

Definition 3.2. Let $\mathcal{E} = (E, F, \prec, \lhd)$ be an RAES. We define the associated configuration system $C(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$ as follows. Let C consist of those $X \subseteq E$ such that $\lhd$ is well-founded on X. For $X \in \mathsf{C}$ and $A \subseteq E$, $B \subseteq F$, we define $X \stackrel{A \cup \underline{B}}{\rightarrow} Y$ if and only if $X, Y \in \mathsf{C}$ and $Y = (X \setminus B) \cup A$ and $A \cup \underline{B}$ is *enabled* at X, which is

- $A \cap X = \varnothing$, $B \subseteq X$;
- for every $a \in A$, if $c \prec a$ then $c \in X \setminus B$;
- for every $a \in A$, if $c \rhd a$ then $c \notin X \cup A$;
- for every $b \in B$, if $d \prec \underline{b}$ then $d \in X \setminus (B \setminus \{b\})$;
- for every $b \in B$, if $d \rhd \underline{b}$ then $d \notin X \cup A$.

We are now able to model undoing of events. If we add $x \prec \underline{x}$, for all $x \in \{a, b, c\}$, and $\underline{a} \lhd b$, $\underline{b} \lhd c$ to $a \prec b \prec c$ and $a \prec c$, then we achieve backtracking in Figure 4. Note that only c can be undone in $\{a, b, c\}$ because $\underline{a} \lhd b$, $\underline{b} \lhd c$ and the presence of b, c prevents undoing of a, b, respectively.

In order to achieve causal reversing we impose the following global conditions: all events are reversible ($x \prec \underline{y}$ if and only if $x = y$ for all x), and causes are undone if and only if their effects are not present ($x \prec y$ if and only if $\underline{x} \lhd y$ for all x, y). In the case of the system $a \prec b, a \prec c$ we add the following to achieve causal reversibility: $a \prec \underline{a}$, $b \prec \underline{b}$, $c \prec \underline{c}$ and $\underline{a} \lhd b$, $\underline{a} \lhd c$. Here, once $\{a, b, c\}$ is reached, b, c can be undone in any order, and a can only be undone when b, c are not present (due to $\underline{a} \lhd b$, $\underline{a} \lhd c$). Overall, we have $\{a, b, c\} \dashrightarrow \{a, b\} \dashrightarrow \{a\}$ and $\{a, b, c\} \dashrightarrow \{a, c\} \dashrightarrow \{a\}$, and clearly $\{a\} \dashrightarrow \varnothing$.

Finally, we model the out-of-causal-order RAES in Figure 4. We have $a \prec b \prec c$ but no $a \prec c$ (so $\prec$ is not transitive) and $a \prec \underline{a}$, $b \prec \underline{b}$ (there is no $c \prec \underline{c}$ since c is irreversible). That a, b are undone only when b, c are present is ensured by $b \prec \underline{a}$, $c \prec \underline{b}$, respectively. In order to stop reversing b immediately after it occurs we add $\underline{b} \lhd a$. And, $a \lhd b$, $a \lhd c$ prevent a from re-occurring when b or c are present. As a result, there is a single execution $\varnothing \rightarrow \{a\} \rightarrow \{a, b\} \dashrightarrow \{b\} \rightarrow \{b, c\} \dashrightarrow \{c\}$.

The work on reversing asymmetric event structures in [19] led to several interesting results concerning reachable configurations. For example, we have given conditions under which finite and reachable configurations are guaranteed to be reachable without intermediate infinite configurations. Our models are general enough to allow several forms of reversibility to be defined and analysed, including the causal and *inverse causal* disciplines.

4 Reversible Event Structures with Enablings

There are forms of execution of three events a, b, c which cannot be achieved by any combination of the causation, concurrency and precedence relations. For example, consider an event that is caused by a disjunction of events: namely a or b causes c. This is called *disjunctive causation*. If no other relation holds of a, b, c, then there is an execution where only a occurs before c, there is another execution where only b occurs prior to c, and there are two executions where both a and b precede c. These complete executions acb, bca, abc and bac are depicted in the left cube in Figure 5. This event structure can be defined using the *enabling* relation as in [15, 26] as we shall see below. Another example of a relation on events that cannot be expressed in terms of causality, concurrency and precedence is *resolvable conflict*. Consider a temporary conflict between a and b which becomes resolved once a third event c occurs. This is represented by the executions acb, bca, cab and cba in the cube on the right in Figure 5. This event structure cannot be expressed with the traditional enabling relation; instead a more general enabling relation from [23] or our *enabling with prevention* relation, that we recall below, are necessary.

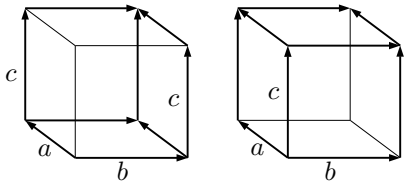

Fig. 5.

Firstly, we recall some definitions from [26]. Event structures are triples $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ where E is a set of events with typical elements e, e', $\mathsf{Con} \subseteq \mathcal{P}_{\mathrm{fin}}(E)$ is the *consistency* relation which is non-empty and satisfies the property $Y \subseteq X \in \mathsf{Con}$ implies $Y \in \mathsf{Con}$ (downwards closure), and $\vdash \subseteq \mathsf{Con} \times E$ is the *enabling* relation which satisfies the *weakening* condition $X \vdash e$ and $X \subseteq Y \in \mathsf{Con}$ implies $Y \vdash e$ for all $e \in E$. We omit brackets for singleton sets in expressions $X \vdash e$ where convenient. Informally, configurations are the sets of events that have occurred (in accordance with Con and $\vdash$). More formally, we let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ be an event structure. The set $S(\mathcal{E})$ of *configurations* of $\mathcal{E}$ consists of $X \subseteq E$ which are

- *consistent*: every finite subset of X is in Con;
- *secured*: for all $e \in X$ there is a sequence of events $e_0, \ldots, e_n \in X$ such that $e_n = e$ and for all $i < n$, $\{e_0, \ldots, e_{i-1}\} \vdash e_i$.

We shall now present several examples of event structures with enablings and their corresponding configurations.

Consider the events a, b with all subsets of $\{a, b\}$ in Con, and the enabling relation $\varnothing \vdash a$, $a \vdash b$. We notice that $\{a\}$ is a configuration because $\{a\} \in \mathsf{Con}$ and a is enabled without any preconditions: $\varnothing \vdash a$. Once a takes place, b can happen because $\{a, b\} \in \mathsf{Con}$ and b is enabled by the already performed a: $a \vdash b$. We can say here that a *causes* b and b cannot take place before a happens first.

Some events are in *conflict*: they cannot happen in the same computation. Consider the events a, b as above and the event c which is conflict with a. This

is represented by $\{a,c\} \notin \mathsf{Con}$ and, by the downwards closure property, $\{a,b,c\} \notin \mathsf{Con}$. The enabling relation is $\varnothing \vdash a$, $a \vdash b$ and $\varnothing \vdash c$. The configurations are $\varnothing$, $\{a\}, \{a,b\}$ and $\{c\}$ representing that either a or c can happen initially, but once one has taken place the other cannot happen; see left cube in Figure 6. Some events are independent of each other, or concurrent. Consider the events a, b and c, with no events in conflict. The enabling relation is $\varnothing \vdash a$, $a \vdash b$ and $\varnothing \vdash c$. Since a and c are not in conflict, $\varnothing \vdash a$, $\varnothing \vdash c$ imply that a, c can happen independently of one another, in any order. Moreover, b and c are independent and can happen in any order provided that b always follows a. The configurations are $\varnothing$, $\{a\}, \{a,b\}$, $\{c\}$, $\{a,c\}, \{a,b,c\}$, and can be seen in the cube on the right in Figure 6.

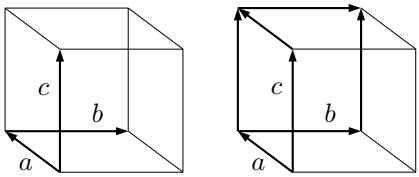

Fig. 6.

We now show how to define the disjunctive causation event structure from Figure 5. If we let the enabling relation as $\varnothing \vdash a$, $\varnothing \vdash b$, and $a \vdash c$ with $b \vdash c$, then we can deduce that $\{c\}$ is not a configuration since we have no $\varnothing \vdash c$. All other subsets of $\{a,b,c\}$ are configurations.

As we aim to generalise event structures with enablings to the reversible setting, we shall use this equivalent definition of a configuration. Let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ be an event structure. A set $X \subseteq E$ is a *configuration* of $\mathcal{E}$ if there is an infinite sequence $X_0, \ldots$ with $X = \bigcup_{n=0}^{\infty} X_n$, $X_0 = \varnothing$, $X_n \subseteq X_{n+1}$ and X_n consistent (all $n \in \mathbb{N}$), where for every $n \in \mathbb{N}$, and every $e \in X_{n+1} \setminus X_n$, there is a rule $X' \vdash e$ with $X' \subseteq_{\text{fin}} X_n$.

As in Section 3, there is a natural notion of computation for configurations in this setting. A transition relation can now be defined to represent how a new event can happen in a configuration giving rise to a bigger configuration. Given configurations X, Y we have $X \to Y$ if $Y = X \cup \{e\}$ (with $e \notin X$) and $X' \vdash e$, for some e and $X' \subseteq_{\text{fin}} X$. A computation of the event structure $\mathcal{E}$ is a computation (sequence of transitions) starting from $\varnothing_{\mathcal{E}}$, the empty configuration of $\mathcal{E}$. Subsequently we omit $\mathcal{E}$ in $\varnothing_{\mathcal{E}}$. As an illustration, $\varnothing \to \{c\} \to \{a,c\} \to \{a,b,c\}$ is a computation of the event structure in the right cube in Figure 6. We also have $\varnothing \to \{a\} \to \{a,c\} \to \{a,b,c\}$ and $\varnothing \to \{a\} \to \{a,b\} \to \{a,b,c\}$.

Finally, we consider undoing of events. Let E be a set of events. We define the corresponding set of *undone* events (strictly speaking, events that are to be undone) to be $\underline{E} = \{\underline{e} : e \in E\}$, where $\underline{E}$ is disjoint from E. For $e \in E$, let e^* be either e or $\underline{e}$; we sometimes use the notation $X + e^*$ to mean either $X \cup \{e\}$ or $X \setminus \{e\}$ respectively. *Reversible event structures* were first introduced in [22]:

Definition 4.1. A reversible event structure (RES for short) is a triple $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ where E and Con are as before and $\vdash \subseteq \mathsf{Con} \times \mathfrak{P}(E) \times (E \cup \underline{E})$ is the *enabling* relation satisfying:

1. if $X \circledcirc Y \vdash e^*$ then $(X \cup \{e\}) \cap Y = \varnothing$;
2. if $X \circledcirc Y \vdash \underline{e}$ then $e \in X$;

3. *weakening*: if $X \odot Y \vdash e^*$ and $X \subseteq X' \in \mathsf{Con}$ then $X' \odot Y \vdash e^*$, provided $X' \cap Y = \varnothing$.

When $Y = \varnothing$ we shall write $X \odot Y \vdash e^*$ as $X \vdash e^*$. Also we omit brackets for singleton sets in expressions $X \odot Y \vdash e^*$ where convenient.

Our enabling relation $\vdash$ extends the enabling relation of Winskel in two directions. Firstly, it permits reversing of events as e^* in $X \odot Y \vdash e^*$ can be an undone event. Secondly, it allows us to specify some of the events that prevent e^* (here those in Y) in addition to the events that enable e^* (those in X). For example, $\{a, b\} \odot \{c, d\} \vdash \underline{a}$ says that a can be undone in a configuration which contains a and b and does not contain c and d.

We are ready to define an RES for resolvable conflict in Figure 5.

Example 4.2. We let Con be $\mathfrak{P}(\{a, b, c\})$. The enabling relation is as follows: $\varnothing \vdash c$, $\varnothing \odot b \vdash a$ and $\varnothing \odot a \vdash b$, meaning that initially, either a or b can take place if the other event is not present. We also have $c \vdash a$ and $c \vdash b$, which imply that both a and b can happen after c.

Example 4.3. Consider an RES with a single event e and the enabling rule $\varnothing \vdash e$. The sets $\varnothing$ and $\{e\}$ are configurations. If we add another rule $e \vdash \underline{e}$ then this allows us to regress from $\{e\}$ to $\varnothing$. The sets $\varnothing$ and $\{e\}$ are reachable from $\varnothing$ in any number of steps; they are configurations according to Definition 4.5 below. And, there is an infinite computation sequence $\varnothing, \{e\}, \varnothing, \{e\}, \ldots$.

This example shows that sets of events can grow and and shrink as reversible computation progresses. Also, sets of events may grow non-monotonically as, for example, in $a_0, b, a_1, \underline{b}, a_2, b, a_3, \underline{b}, a_4, \ldots$. So we shall use limits of infinite sequences of subsets of E in order to define configurations as in [22] (recall that $S \subseteq \mathbb{N}$ is *cofinite* if $\mathbb{N} \setminus S$ is finite):

Definition 4.4. Let $X_0, \ldots$ be an infinite sequence of subsets of E. We say that $X = \lim_{n\to\infty} X_n$ if for every $e \in E$:

1. $\{n \in \mathbb{N} : e \in X_n\}$ is either finite or cofinite;
2. $e \in X$ if and only if $\{n : e \in X_n\}$ is cofinite.

We note that a sequence of sets does not necessarily have a limit. The sequence $\varnothing, \{e\}, \varnothing, \{e\}, \ldots$ in Example 4.3 has no limit, since e belongs to infinitely many sets and does not belong to infinitely many sets. However if $X_n \subseteq X_{n+1}$ (all $n \in \mathbb{N}$) then $\lim_{n\to\infty} X_n$ exists and is $\bigcup_{n=0}^{\infty} X_n$. A finite sequence $X_0, \ldots, X_n$ can be extended to an infinite sequence by letting $X_m = X_n$ for all $m > n$; the extended sequence has the limit X_n. In Example 4.3 the sequence $\varnothing, \{e\}$ can be extended to an infinite sequence $\varnothing, \{e\}, \{e\}, \ldots$ and has the limit $\{e\}$.

Next we state the definition of a configuration for an RES ([22]). As the notational convention we write $\underline{e} \in A \setminus B$ to mean $e \in B \setminus A$.

Definition 4.5. Let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ be an RES. A set $X \subseteq E$ is a *configuration* of $\mathcal{E}$ if there is an infinite sequence $X_0, \ldots$ with $X = \lim_{n\to\infty} X_n$, $X_0 = \varnothing$ and $X_n \cup X_{n+1}$ consistent (all $n \in \mathbb{N}$), where for every $n \in \mathbb{N}$, and every $e^* \in X_{n+1} \setminus X_n$, there is a rule $X' \odot Y' \vdash e^*$ such that:

1. $X' \subseteq_{\text{fin}} X_n$ and $X' + e^* \subseteq X_{n+1}$;
2. $Y' \cap (X_n \cup X_{n+1}) = \varnothing$.

We require $X_n \cup X_{n+1}$ to be consistent, as configurations can only be extended in a consistent fashion. However, there is no requirement that $X_i \cup X_j$ is consistent if $j > i+1$ because events in X_i which are inconsistent with X_j can be reversed in constructing $X_{i+1}, \ldots, X_{j-1}$. Also, we note that the X_is in the above definition can grow smaller as well as bigger as computation progresses. Moreover, a finite sequence $X_0, \ldots, X_n = X$ that satisfies the conditions of Definition 4.5 is sufficient for X to be a configuration. The sequence $\varnothing, \{e\}$ in Example 4.3 can be extended to an infinite sequence and, since the conditions of Definition 4.5 are satisfied, its limit $\{e\}$ is a configuration.

We return to Example 4.2. We note that although $\{a,b\} \in \mathsf{Con}$, $\{a,b\}$ is not a configuration according to Definition 4.5. Consider $\varnothing$, $\{a\}$, $\{a,b\}$ and b: there is no enabling $X' \oslash Y' \vdash b$ such that $X' \subseteq_{\text{fin}} \{a\}$ and $Y' \cap \{a,b\} = \varnothing$. Correspondingly for the sequence $\varnothing$, $\{b\}$, $\{a,b\}$ and a. Hence, $\{a,b\}$ is not a configuration.

It can be easily shown that RESs are a generalisation of event structures: RESs with enablings $X \oslash \varnothing \vdash e$ are just event structures of Winskel [26]. Moreover, our configurations in such setting are just the traditional configurations. We can also show that our generalised enabling rules are powerful enough that we no longer need the consistency relation.

We are now ready to define a transition relation between configurations of an RES. Again, as in Section 2, we shall use the dashed arrow notation for the part of the transition relation that represents undoing of events. Given configurations X, Y of an RES $\mathcal{E}$ we let

- $X \to Y$ if $Y = X \cup \{e\}$ and $X' \oslash Z \vdash e$ for some e, X', Z with $e \notin X$, $X' \subseteq_{\text{fin}} X$ and $Z \cap (X \cup \{e\}) = \varnothing$;
- $X \dashrightarrow Y$ if $Y = X \setminus \{e\}$ and $X' \oslash Z \vdash \underline{e}$ for some e, X', Z with $X' \subseteq_{\text{fin}} X$ and $Z \cap X = \varnothing$.

In contrast to Section 2, this transition relation represents only either performing a single event or undoing a single event. Having given the transition relation, we can now define a configuration system for an RES. Given an RES $\mathcal{E} = (E, \mathsf{Con}, \vdash)$, the associated configuration system $C(\mathcal{E})$ is $(E, E, \mathsf{C}, \to)$ where C is the set of configurations for $\mathcal{E}$ as in Definition 4.5.

We now show how to represent different forms of undoing of events in RESs. Consider events a and b with $\varnothing \vdash a$ and $a \vdash b$. We have that a causes b so if we wish to achieve causal reversing we need to add the following to the definition of $\vdash$: $b \vdash \underline{b}$ and $a \oslash b \vdash \underline{a}$. The configuration $\{a,b\}$ can regress to $\{a\}$ by undoing b as allowed by $b \vdash \underline{b}$. But it cannot regress to $\{b\}$ because $a \oslash b \vdash \underline{a}$ can only be applied in a configuration that contains a and does not contain b. See Figure 7(i).

If undoing events in the same order as they occurred is required, we instead add to the definition of $\vdash$ the following: $a \vdash \underline{a}$ and $b \oslash a \vdash \underline{b}$. This means that a can be reversed in any configuration that contains a (with or without b), and b can be reversed only when a is not present. Since a causes b, this means that b can be reversed only when a is reversed. See Figure 7(ii) where reverse transitions are

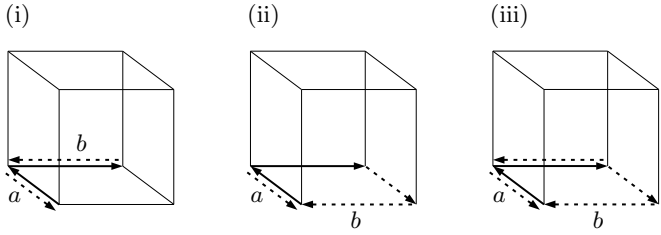

Fig. 7.

indicated by dashed lines. Finally, if we would like instead that a and b are reversed in any order, then we would extend the enabling relation simply with $b \vdash \underline{b}$ and $a \vdash \underline{a}$. See Figure 7(iii).

Finally, we give an example where we get an infinite configuration as a limit of a non-monotonically increasing sequence ([22]).

Example 4.6. Let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ where $E = \{a_i : i \in \mathbb{N}\} \cup \{b_j : j \in \mathbb{N}\}$ and Con consists of $\{a_i, b_0, \ldots, b_j\}$ (any $i, j \in \mathbb{N}$) plus deducible subsets, with

$$\varnothing \vdash a_0 \quad a_i \vdash b_i \quad \{a_i, b_i\} \vdash \underline{a}_i \quad b_i \vdash a_{i+1} \quad (\text{all } i \in \mathbb{N})$$

Informally, a_i is the catalyst of b_i, for all i, so once b_i occurs a_i can be undone.

The only possible computation of $\mathcal{E}$ is $a_0, b_0, \underline{a}_0, a_1, b_1, \underline{a}_1, \ldots$. It produces the following sequence of sets of events, which grow non-monotonically:

$$\begin{array}{l} \varnothing, \\ \{a_0\},\ \{a_0, b_0\},\ \{b_0\}, \\ \{b_0, a_1\},\ \{b_0, a_1, b_1\},\ \{b_0, b_1\}, \\ \{b_0, b_1, a_2\},\ \{b_0, b_1, a_2, b_2\},\ \{b_0, b_1, b_2\},\ \ldots \end{array}$$

Each of the sets is a configuration and this sequence has limit the infinite set $\{b_j : j \in \mathbb{N}\}$, so $\{b_j : j \in \mathbb{N}\}$ is also a configuration. Note that each a_i appears finitely often in the sequence, while each b_j appears cofinitely often.

5 Discussion and Conclusions

We indicate briefly several areas of ongoing research in reversing event structures.

We aim to investigate the expressiveness of event structures defined with our enabling relation with prevention (for forwards-only events), and compare them with other forms of event structures. In particular, it remains to be seen whether or not we can encode event structures of van Glabbeek and Plotkin [23], which are defined by a very general form of enabling relation ($X \vdash Y$ where X, Y are sets of events), as our event structures from Section 4, or vice versa.

The examples in the previous section indicate that it ought to be possible to represent an arbitrary RAES as a special form of RES. Given an RAES if $X_a = \{e \mid e \prec a\}$ and $Y_a = \{f \mid a \lhd f\}$, then the enabling rule $X_a \circledcirc Y_a \vdash a$ captures the idea that a can occur if all events in X_a have occurred (and are present) and

if no events from Y_a are present. It should be then routine to define conditions 3 to 7 of RAESs in terms of our enabling relation. For example, condition 3 is expressed as if $X \odot Y \vdash \alpha$ and $a \in X$, then $a \notin Y$: this is already guaranteed by condition 1 in Definition 4.1. Since disjunctive causation cannot be expressed in RAESs, RESs are strictly more expressive that RAESs.

Another challenge is to define step and mixed transitions between configurations in the RES setting. In order to define the notion of a set of enabled events and past events (as in Definition 3.2), we need to devise a way of dealing with enabling rules that are obtained via weakening (Definition 4.1). Assume that we wish to check if $A \cup \underline{B}$ is enabled at X. Clearly, we require $A \cap X = \varnothing$ and $B \subseteq X$. If $a \in A$ and $X_a \odot Y_a \vdash a$, then we also require that $X_a \subseteq X \setminus B$ and $(X \cup A) \cap Y_a = \varnothing$. However, the last condition will not necessarily hold for rules gotten from $X_a \odot Y_a \vdash a$ by weakening. Assume that $X_a \cup \{d\}$ is consistent and $d \notin X_a, Y_a, X$. Then $(X_a \cup \{d\}) \odot Y_a \vdash a$ is an enabling obtained from $X_a \odot Y_a \vdash a$ by weakening, but $X_a \cup \{d\}$ is not a subset of $X \setminus B$.

Concluding, we have shown how to model reversibility in concurrent computation as realised by two different forms of events structures, namely event structures defined in terms of the causation and precedence relations and event structures defined by the enabling relation. We have presented causal reversibility as well as out-of-causal-order reversibility.

Acknowledgements. The first author thanks the University of Leicester for granting Academic Study Leave and acknowledges partial support from the JSPS Invitation Fellowship grant S13054.

References

[1] Baldan, P., Corradini, A., Montanari, U.: Contextual Petri nets, asymmetric event structures, and processes. Information and Computation 171(1), 1–49 (2001)

[2] Bednarczyk, M.A.: Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical Report ICS PAS, Polish Academy of Sciences (1991)

[3] Berry, G., Boudol, G.: The chemical abstract machine. Theoretical Computer Science 96(1), 217–248 (1992)

[4] Cardelli, L., Laneve, C.: Reversible structures. In: 9th International Conference on Computational Methods in Systems Biology, pp. 131–140. ACM (2011)

[5] Cristescu, I., Krivine, J., Varacca, D.: A compositional semantics for the reversible p-calculus. In: Proceedings of LICS 2013, pp. 388–397. IEEE Computer Society (2013)

[6] Danos, V., Krivine, J.: Reversible communicating systems. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 292–307. Springer, Heidelberg (2004)

[7] Danos, V., Krivine, J.: Transactions in RCCS. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 398–412. Springer, Heidelberg (2005)

[8] Danos, V., Krivine, J.: Formal molecular biology done in CCS-R. In: Proceedings of BioConcur 2003. ENTCS, vol. 180, pp. 31–49 (2007)

[9] Giachino, E., Lanese, I., Mezzina, C.A.: Causal-consistent reversible debugging. In: Gnesi, S., Rensink, A. (eds.) FASE 2014 (ETAPS). LNCS, vol. 8411, pp. 370–384. Springer, Heidelberg (2014)
[10] van Glabbeek, R.J., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica 37, 229–327 (2001)
[11] Hennessy, M., Milner, R.: Algebraic laws for non-determinism and concurrency. Journal of the ACM 32, 137–161 (1985)
[12] Lanese, I., Mezzina, C.A., Schmitt, A., Stefani, J.-B.: Controlling reversibility in higher-order pi. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 297–311. Springer, Heidelberg (2011)
[13] Lanese, I., Mezzina, C.A., Stefani, J.-B.: Reversing higher-order pi. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 478–493. Springer, Heidelberg (2010)
[14] Lanese, I., Mezzina, C.A., Stefani, J.-B.: Controlled reversibility and compensations. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 233–240. Springer, Heidelberg (2013)
[15] Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, part I. Theoretical Computer Science 13, 85–108 (1981)
[16] Phillips, I.C.C., Ulidowski, I.: Reversing algebraic process calculi. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 246–260. Springer, Heidelberg (2006)
[17] Phillips, I.C.C., Ulidowski, I.: Reversing algebraic process calculi. Journal of Logic and Algebraic Programming 73, 70–96 (2007)
[18] Phillips, I.C.C., Ulidowski, I.: A hierarchy of reverse bisimulations on stable configuration structures. Mathematical Structures in Computer Science 22, 333–372 (2012)
[19] Phillips, I., Ulidowski, I.: Reversibility and asymmetric conflict in even structures. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013. LNCS, vol. 8052, pp. 303–318. Springer, Heidelberg (2013)
[20] Phillips, I.C.C., Ulidowski, I.: Event Identifier Logic. Mathematical Structures in Computer Science 24, 1–51 (2014)
[21] Phillips, I.C.C., Ulidowski, I., Yuen, S.: A reversible process calculus and the modelling of the ERK signalling pathway. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 218–232. Springer, Heidelberg (2013)
[22] Phillips, I.C.C., Ulidowski, I., Yuen, S.: Modelling of bonding with processes and events. In: Dueck, G.W., Miller, D.M. (eds.) RC 2013. LNCS, vol. 7948, pp. 141–154. Springer, Heidelberg (2013)
[23] van Glabbeek, R.J., Plotkin, G.D.: Configuration structures, event structures and Petri nets. Theoretical Computer Science 410(41), 4111–4159 (2009)
[24] van Glabbeek, R.J., Plotkin, G.D.: Event structures for resolvable conflict. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 550–561. Springer, Heidelberg (2004)
[25] Varacca, D., Yoshida, N.: Typed event structures and the linear π-calculus. Theoretical Computer Science 411(19), 1949–1973 (2010)
[26] Winskel, G.: Event structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 255, pp. 325–392. Springer, Heidelberg (1987)
[27] Winskel, G.: Events, causality and symmetry. Computer Journal 54(1), 42–57 (2011)