Modelling of Bonding with Processes and Events

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

[1] Department of Computing, Imperial College London, England
iccp@doc.ic.ac.uk
[2] Department of Computer Science, University of Leicester, England
iu3@mcs.le.ac.uk
[3] Graduate School of Information Science, Nagoya University, Japan
yuen@is.nagoya-u.ac.jp

Abstract. We introduce two forms of modelling of systems that consist of objects that are combined together by the means of bonds. In reaction systems for bonding we define how bonds are created and dissolved via reduction-style semantics. The usefulness of reaction systems is illustrated with examples taken from software engineering and biochemistry. We also introduce reversible event structures and define the notion of configuration. We then discuss how to give semantics of reaction systems for bonding in terms of reversible event structures.

1 Introduction

Undoing of computation in concurrent and distributed systems has many technical and conceptual challenges. There are several forms of undoing computation that have been studied over the last ten years. Backtracking and reversing of computation that preserves causal order were considered, for example, in [5, 3, 10, 11, 7, 2]. Reversing out of causal order, however, which is a very common mode of operation in biochemical systems, has not been studied widely. The first attempt was made in [12] where an extension of the reversible process calculus CCSK with the execution control operator was proposed. A different form of controlling reversibility based on the rollback construct of the higher-order π calculus was given in [6].

Let us recall what backtracking and reversing is, both in causal order and out of causal order [12]. Consider a computation where the event a causes the event b, written $a < b$, and the event c occurs independently of a and b. The three traces of this computation that preserve causality are abc, acb and cab: note that a always precedes b. There are several conceptually different ways of undoing these events. *Backtracking* is undoing in precisely the inverse order in which they happened. So, undo b undo c undo a, written as $\underline{b}\ \underline{c}\ \underline{a}$, is a backtrack of acb. *Reversing* is more general: here events can be undone in any order as long as causality is preserved, meaning that causes cannot be undone before effects. For example, $\underline{c}\ \underline{b}\ \underline{a}$ is a reversal of acb for a, b and c as defined above. However, there are processes, especially common in cell biochemistry, where events are undone, *out of causal order*. The creation and breaking of molecular bonds between

the proteins involved in the ERK signalling pathway or the creation of polymers by scaffolding proteins, described in Sections 2 and 3, are good examples. Simplifying, let us assume that the creation of molecular bonds is represented by events a, b, c where, as above, $a < b$ and c is independent of a and b. In the ERK pathway, the molecular bonds are broken in the following order: $\underline{a}$, $\underline{b}$ and $\underline{c}$, which apparently reverses the cause a before the effect b.

In the paper we propose two alternative methods (to [12]) for defining forwards and reverse computation. We draw some inspiration from the fields of graph and term rewriting, and define reaction systems for creating and dissolving bonds between objects, and explain how computation, both forwards and in reverse, can be modelled as a process of bonding and unbonding. We show with examples that reaction systems for bonding can represent naturally reversing out of order, and that they have an expressive power comparable to CCSK with the execution control operator [12].

Event structures were proposed by Winskel in [16] as a denotational model of concurrent computation. Systems are represented as sets of events which are constrained by relations of *consistency* and *enabling*. Event structures allow us to discuss directly relationships between events such as concurrency, causality and conflict. Our contribution is a definition of *reversible event structures*. To the best of our knowledge this is the first form of event structure where computation can proceed both forwards and in reverse, both in and out of causal order. We also describe, given a reaction system and an initial process, how to construct a reversible event structure that captures the behaviour of the initial process.

The benefits of reaction systems for bonding and reversible event structures are demonstrated with several examples taken from software engineering and cell biochemistry. We consider the modelling of long-running transactions, creation of a polymer by scaffolding proteins, and a signal-passing mechanism employed by a section of the ERK signalling pathway. All our examples show how crucial out-of-order reversing is in the world of artificial and natural systems.

2 Reaction systems for bonding

We develop reaction systems for representing objects that can bond with each other, thus creating more complex objects, and where bonds can be dissolved within a composite object. The building blocks of the calculus are the *base objects*, or simply called *bases*. A base object has a *sort*, for example A, and an *arity* which is the maximal number of bonds the base object can have with other base objects. Consider two bases of sort A and arity 1 and a base of sort B and arity 2. We shall write a collection of these three objects simply using their sort names as A, A, B, or as A_1, A_2, B if we wish to distinguish between objects of the same sort (here A). A creation of a *single* bond between A and B is defined by the relation $\rightarrow$, for example

$$A, B \rightarrow A \cdot B$$

where '$\cdot$' in $A \cdot B$ denotes the bond between the A and B, and $A \cdot B$ is a (composite) object consisting of bases A and B. A single bond can be dissolved and this is

given by the relation $\rightsquigarrow$, for example

$$A \cdot B \rightsquigarrow A, B$$

A system of objects, also called a *process*, is a collection of objects written with the comma operator ',', for example A, A, B. Many of our examples are inspired by biochemistry, and so we shall call objects *molecules* and a system of objects a *solution*. The order in which objects are written in a system is irrelevant, and can be changed. A molecule can move around in a solution, thus changing relative position with respect to other molecules. This is defined by structural equivalence $X, Y \equiv Y, X$, where X, Y are objects or molecules. Systems, processes or solutions, S and T can be combined by taking their multiset union, written as S, T. Clearly, $S, T \equiv T, S$.

Given $A, B \rightarrow A \cdot B$, we would like to deduce that a system containing A and B can evolve to a system that contains $A \cdot B$. This is done by having two global rules inspired by the laws for the chemical abstract machine [1]:

$$\frac{S \rightarrow S'}{S, T \rightarrow S', T} \quad (c1) \qquad \frac{S \rightsquigarrow S'}{S, T \rightsquigarrow S', T} \quad (c2)$$

We call this substitutivity in the ',' context. We shall also have rules for structural equivalence:

$$\frac{S \equiv S' \quad S' \rightarrow T' \quad T' \equiv T}{S \rightarrow T} \quad (s1) \qquad \frac{S \equiv S' \quad S' \rightsquigarrow T' \quad T' \equiv T}{S \rightsquigarrow T} \quad (s2)$$

We do not have substitutivity in the '$\cdot$' context. If $A \cdot B \rightsquigarrow A, B$, we do not always wish to have $C \cdot A \cdot B \rightsquigarrow C \cdot A, B$, for example, when C inhibits the dissolution of $A \cdot B$. Our reaction systems are unlike term or graph rewrite systems for this reason.

Two bases D of arity 2 can have two bonds between each other; this is written as $\overline{D \cdot D}$ or, equivalently, $\underline{D \cdot D}$. A ring of three copies of D is $\overline{D \cdot D \cdot D}$ or $\underline{D \cdot D \cdot D}$. Let $x \cdot A$ denote an object consisting of a base A and an object x where there is precisely one bond between A and some base in x. We can generalise this notation (using over- or under-bracket notation) to denote that there are k bonds between A and other base molecules in x, assuming that the arity of A is $n \geq k$ and x has capacity for at least k fresh bonds. Since all examples in this paper and in [12] use molecules with arity at most 3 the notation $x \cdot A$, $\overline{x \cdot A}$, $\underline{x \cdot A}$ is sufficient.

Next, we set out rules for structural equivalence for bonding $\equiv$. Objects can be seen as undirected graphs, where the bases of an object are the nodes and the bonds between the bases of the object are the edges, and the arity of a base is an upper bound on the degree of the corresponding node. Structural equivalence of two objects implies that the underlying graphs are isomorphic, and we note that isomorphism of graphs with bounded degree is decidable in polynomial time [8]. For illustration, we give a set of rules for $\equiv$ for objects with bases of arity

at most two. Let y represent a possibly empty arbitrary string of bonded bases $A_1 \cdots A_n$ $(n \geq 0)$, and let y^{-1} represent y in reverse order, i.e. $A_n \cdots A_1$.

$$A \cdot y \equiv y^{-1} \cdot A \qquad \overline{A \cdot y \cdot B} \equiv \overline{B \cdot y^{-1} \cdot A}$$

$$\overline{A \cdot y \cdot B \cdot C} \equiv \overline{C \cdot A \cdot y \cdot B} \qquad \overline{A, y \cdot B} \equiv A \cdot B \cdot y^{-1}$$

$$\overline{A \cdot y \cdot B} \equiv \underline{A \cdot y \cdot B}$$

Our first example is a reaction system that models how a catalyst molecule helps otherwise inactive molecules to bond.

Example 2.1. Consider molecules of sort A and B and arity 2 and 1 respectively that cannot bond easily unless they are "assisted" by a catalyst molecule of sort C and arity 1. In a solution that contains copies of A, B and C, molecules C and A bond initially by the rule

$$C, A \to C \cdot A$$

Then, molecules $C \cdot A$ combine with copies of B

$$C \cdot A, B \to C \cdot A \cdot B$$

Finally, having helped A and B to react, the bond between C and A is broken thus releasing C into a solution to help other A, B pairs

$$C \cdot A \cdot B \rightsquigarrow C, A \cdot B$$

Consider a solution $S = A, B, A, B, C$. It computes as follows:

$$S \equiv^* C, A, B, A, B \to C \cdot A, B, A, B \to C \cdot A \cdot B, A, B \rightsquigarrow C, A \cdot B, A, B$$

Next, the molecules rearrange themselves into $A \cdot B, C, A, B$ and the computation continues as follows:

$$A \cdot B, C, A, B \to A \cdot B, C \cdot A, B \to A \cdot B, C \cdot A \cdot B \rightsquigarrow A \cdot B, C, A \cdot B$$

producing two molecules $A \cdot B$ and, of course, the original C. This solution could have produced the same outcome by following a different route: C could have reacted with the second pair of A and B first by positioning itself initially in the middle of the solution.

Definition 2.2. A *reaction system for bonding* is a tuple $\mathrm{P} = (\Sigma, \to, \rightsquigarrow)$ where the signature Σ contains the bonding operator '$\cdot$', the solution operator ',' and the definitions of base objects in the form of sort, arity pairs. Objects are either base objects or collections of bonded bases such that the arities of the bases are not exceeded and each base is connected to another base via a sequence of bonds. A process (or a system or solution) is either an object or a collection of objects composed with the solution operator. Reduction relations $\to$ and $\rightsquigarrow$ are binary relations over processes. A *computation* starts with an initial process (solution), which is a multi-set of bases taken from Σ, and is a sequence of transitions derived from $\to$ and $\rightsquigarrow$ and the rules (c1-c2) and (s1-s2).

In the setting of reaction systems for bonding, we specify which bonds are created and dissolved, and in which order, by the reduction style relations $\to$ and $\rightsquigarrow$. This is in contrast to the majority of previously considered reversible calculi where the syntax of processes (prefixing, keys, parallel composition and restriction) and global operational semantics determine the forwards and reverse computation.

The next example is a reaction system for the modelling of long-running transactions with a compensation. Previously, transactions were also considered in the setting of reversible process calculi in [4, 12].

Example 2.3. We define the signature first. A transaction is a sequence of $n \geq 3$ steps A_i for $1 \leq i < n$ of sort A and arity 3. It starts with the initial step I, arity 2, which never fails. I is followed by the steps A_i, where A_1 bonds with I and then each consecutive A_{i+1} bonds to A_i, indicating a successful completion of A_i. Finally, a success step S of arity 1 occurs which is represented by a bond between A_n and S. A transaction can fail at any stage after I. This is represented by a fail object F of arity 2 bonding with the last A_i, which represents the failure of A_i. When this happens all steps A_k for $1 \leq k \leq i$ are undone, and then the compensation step C of arity 1 takes place.

The forwards and reverse reduction relations determine which bonds are created and which are dissolved. Firstly, the chain of A_is is created

$$\begin{aligned}
&I, A_1 && \to I \cdot A_1 && \\
&I \cdot A_1, A_2 && \to I \cdot A_1 \cdot A_2 && \\
&x \cdot A_i \cdot A_{i+1}, A_{i+2} && \to x \cdot A_i \cdot A_{i+1} \cdot A_{i+2} && 1 \leq i < n-2 \\
&x \cdot A_n, S && \to x \cdot A_n \cdot S &&
\end{aligned}$$

Fail can occur at any stage of building the chain of A_is, where $1 \leq i < n$:

$$\begin{aligned}
&I \cdot A_1, F && \to I \cdot A_1 \cdot F && \to \overline{I \cdot A_1 \cdot F} \\
&x \cdot A_i \cdot A_{i+1}, F && \to x \cdot A_i \cdot A_{i+1} \cdot F && \to x \cdot \overline{A_i \cdot A_{i+1} \cdot F}
\end{aligned}$$

Following a fail transaction steps are undone

$$\begin{aligned}
&\overline{I \cdot A_1 \cdot F} && \rightsquigarrow \overline{I,\ A_1 \cdot F} && \rightsquigarrow I \cdot F,\ A_1 \\
&x \cdot \overline{A_i \cdot A_{i+1} \cdot F} && \rightsquigarrow x \cdot \overline{A_i,\ A_{i+1} \cdot F} && \rightsquigarrow x \cdot A_i \cdot F,\ A_{i+1}
\end{aligned}$$

Note that $\overline{I,\ A_1 \cdot F} \equiv I \cdot F \cdot A_1$ cannot bond with A_2, and correspondingly $x \cdot A_i \cdot F \cdot A_{i+1}$ cannot bond with A_{i+2}. This is due to the form of the first four rules for $\to$.

Finally, compensation takes place

$$C,\ I \cdot F \to C \cdot I \cdot F \rightsquigarrow C \cdot I,\ F$$

The initial process $S = I, A_1, \ldots, A_n, S, C, F$ can reach one of the two terminated processes: the transaction has completed successfully $I \cdot A_1 \cdot \ldots \cdot A_n \cdot S,\ C, F$, or the transaction has failed and the compensation has taken place $I \cdot C,\ A_1, \ldots, A_n, S, F$.

Example 2.4. This reaction system describes how a polymer is constructed by a scaffolding protein. We have scaffolding proteins S of arity 2 and and polymer molecules Q of arity 3. Firstly, two molecules of sort S combine into a scaffolding and then attract copies of Q

$$S, S \to S \cdot S \qquad S \cdot S, Q \to S \cdot S \cdot Q \qquad x \cdot S \cdot S, Q \to x \cdot S \cdot S \cdot Q$$

Once two copies of Q are bonded to the scaffolding they bond together

$$\begin{aligned} Q \cdot S \cdot S \cdot Q \quad &\to \overline{Q \cdot S \cdot S \cdot Q} \quad \equiv \overline{Q \cdot Q \cdot S \cdot S} \\ x \cdot Q \cdot S \cdot S \cdot Q \to x \cdot \overline{Q \cdot S \cdot S \cdot Q} &\equiv x \cdot \overline{Q \cdot Q \cdot S \cdot S} \end{aligned}$$

Note that for the last structural equivalence we need a further rule, where y, z can be empty, in addition to those we gave earlier for arities at most two

$$y \cdot \overline{A \cdot B \cdot z \cdot C} \equiv y \cdot \overline{A \cdot C \cdot z^{-1} \cdot B}$$

Now the bond between the last S and Q breaks

$$\begin{aligned} \overline{Q \cdot Q \cdot S \cdot S} \quad &\rightsquigarrow Q \cdot Q \cdot S \cdot S \\ x \cdot \overline{Q \cdot Q \cdot S \cdot S} &\rightsquigarrow x \cdot Q \cdot Q \cdot S \cdot S \end{aligned}$$

The S at the end is now available to bond to an unattached Q (using the rule $x \cdot S \cdot S, Q \to x \cdot S \cdot S \cdot Q$ given above), and the process can continue while unattached Qs remain.

An initial solution consisting of two copies of S and four copies of Q becomes eventually the solution $Q \cdot Q \cdot Q \cdot Q \cdot S \cdot S$. If we add two further rules below for breaking bonds between molecules bonded to S, the solution computes further to $Q \cdot Q \cdot Q \cdot Q, S, S$.

$$\begin{aligned} x \cdot S \cdot S &\rightsquigarrow x,\ S \cdot S \\ S \cdot S \quad &\rightsquigarrow S,\ S \end{aligned}$$

The addition of $S \cdot S \rightsquigarrow S,\ S$ makes the computation potentially non-terminating: $S,\ S \to S \cdot S \rightsquigarrow S,\ S \ldots$.

The last two examples show that reactive systems are capable of expressing both causal reversing and out-of-causal-order reversing and, in this respect, reactive systems are comparable to CCSK with the execution control operator [12] and extend what RCCS [3] and higher order roll-π [6] can express.

3 Reversible event structures

In this section we recall what event structures are, define the reversible form of event structures including a new notion of configuration in the reversible setting, and discuss how simple reaction systems from Section 2 can be given meaning in terms of reversible event structures.

3.1 Event structures

Event structures were defined by Winskel [16] following earlier work by Nielsen, Plotkin and Winskel [9]. They were further developed, for example, in [14, 13, 17] and in [15].

Definition 3.1 ([16, Def. 1.1.1]). *Event structures* are triples $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ where E is a set of events with typical elements e, e', $\mathsf{Con} \subseteq \mathcal{P}_{\text{fin}}(E)$ 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, events are things that happen, for example a creation of a bond between bases A and B, a communication of a value between a sender and a receiver, a part of a long-running transaction. Configurations are the sets of events that have occurred (in accordance with Con and $\vdash$):

Definition 3.2 ([16, Def. 1.1.2]). 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 $i < n$, $\{e_0, \ldots, e_{i-1}\} \vdash e_i$.

Example 3.3. Consider the events a, b with all subsets of $\{a, b\}$ in Con, and the enabling relation $\emptyset \vdash a$, $a \vdash b$. We notice that $\{a\}$ is a configuration because $\{a\} \in \mathsf{Con}$ and a is enabled without any preconditions: $\emptyset \vdash a$. 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 $\emptyset \vdash a$, $a \vdash b$ and $\emptyset \vdash c$. The configurations are $\emptyset$, $\{a\}, \{a, b\}$ and $\{c\}$ representing that either a or c can happen initially, but once one has taken place the other cannot happen.

Example 3.4. Some events are *independent* of each other, or concurrent. Consider the events a, b and d, with no events in conflict. The enabling relation is $\emptyset \vdash a$, $a \vdash b$ and $\emptyset \vdash d$. Since a and d are not in conflict $\emptyset \vdash a$, $\emptyset \vdash d$ imply that a, d can happen independently of one another, in any order. Moreover, b and d are independent and can happen in any order provided that b always follows a. The configurations are $\emptyset$, $\{a\}, \{a, b\}$, $\{d\}$, $\{a, d\}, \{a, b, d\}$.

The next definition is equivalent to Definition 3.2; it will be easier to generalise to the reversible setting. It is partly inspired by the step-wise securings of [13, Definition 3.5].

Definition 3.5. 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 $X = \bigcup_{n=0}^{\infty} X_n$, $X_0 = \emptyset$, $X_n \subseteq X_{n+1}$ 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 $X' \subseteq_{\text{fin}} X_n$.

Proposition 3.6. *Let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ be an event structure and let $X \subseteq E$. Then X is a configuration according to Definition 3.2 iff X is a configuration according to Definition 3.5.*

There is a natural notion of *computation* for configurations. 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 E is a computation (sequence of transitions) starting from $\emptyset_{\mathcal{E}}$, the empty configuration of $\mathcal{E}$. As an illustration, $\emptyset \to \{d\} \to \{a, d\} \to \{a, b, d\}$ is a part of a computation of the event structure in Example 3.4. We also have $\emptyset \to \{a\} \to \{a, d\} \to \{a, b, d\}$ and $\emptyset \to \{a\} \to \{a, b\} \to \{a, b, d\}$. See Figure 1.

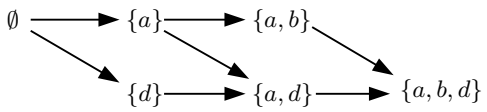

Fig. 1. Configurations and transitions in Example 3.4.

We now return to Example 2.1. The bonds in $C \cdot A$ and $A \cdot B$ are the events, and we denote them as ca and ab. The enabling relation is $\emptyset \vdash ca$, $ca \vdash ab$. If we consider the order in which the bonds are created we deduce that ca causes ab. If these bonds were to be dissolved in a causality preserving manner, then ab ought to be reversed first, and only then ca. But breaking the bonds in this manner would not lead to any real change or progress: we would end up where we started. If the bonds are undone out of causal order, then there may be progress. In this example, if ca is dissolved and ab is left untouched, we have the molecule $A \cdot B$ at the end. This would have not been possible if we reversed in causal order. The main question of this paper thus arises: how do we represent undoing of events in any order in the setting of event structures?

3.2 Reversible event structures

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.

Definition 3.7. A *reversible event structure* (RES for short) is a triple $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ where E and Con are as before and $\vdash \subseteq \mathsf{Con} \times \mathcal{P}(E) \times (E \cup \underline{E})$ is the *enabling* relation satisfying:

1. if $X \odot Y \vdash e^*$ then $(X \cup \{e\}) \cap Y = \emptyset$;
2. if $X \odot Y \vdash \underline{e}$ 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 $X' \cap Y = \emptyset$.

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

The new enabling relation $\vdash$ extends the enabling relation from Definition 3.1 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 which events 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.

Example 3.8. Consider an RES with a single event e and the enabling rule $\emptyset \vdash e$. As in the previous subsection the sets $\emptyset$ and $\{e\}$ are configurations. Next we add another rule $e \vdash \underline{e}$. This allows us to regress from $\{e\}$ to $\emptyset$. Now the sets $\emptyset$ and $\{e\}$ are reachable from $\emptyset$ in any number of steps; they are configurations according to Definition 3.10 below. There is, however, an infinite computation sequence $\emptyset, \{e\}, \emptyset, \{e\}, \ldots$.

The example illustrates that in the reversible setting sets of events can grow and and shrink as computation progresses. Also, it may happen that sets of events 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 need to consider limits of infinite sequences of subsets of E in order to define configurations. Recall that a subset $S \subseteq \mathbb{N}$ is *cofinite* if $\mathbb{N} \setminus S$ is finite.

Definition 3.9. 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. $e \in X$ iff $\{n : e \in X_n\}$ is cofinite.

Note that a sequence of sets does not necessarily have a limit. The sequence $\emptyset, \{e\}, \emptyset, \{e\}, \ldots$ 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 $\bigcup_{n=0}^{\infty} X_n$. Also note that 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 3.8 the sequence $\emptyset, \{e\}$ can be extended to an infinite sequence $\emptyset, \{e\}, \{e\}, \ldots$ and has the limit $\{e\}$.

Next we define configurations for RESs. Our aim is that they generalise configurations in Definition 3.5. We use the notational convention that $\underline{e} \in A \setminus B$ means $e \in B \setminus A$.

Definition 3.10. 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 $X = \lim_{n\to\infty} X_n$, $X_0 = \emptyset$ 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 $X' \otimes Y' \vdash e^*$ such that:

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

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 $j > i + 1$; events in X_i which are inconsistent with X_j can be reversed in constructing $X_{i+1}, \ldots, X_{j-1}$. Also, 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 3.10 is sufficient for X to be a configuration. The sequence $\emptyset, \{e\}$ in Example 3.8 can be extended to an infinite sequence and, since the conditions of Definition 3.10 are satisfied, its limit $\{e\}$ is a configuration.

The next result shows that RESs are a generalisation of event structures.

Proposition 3.11. *Suppose $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ is an event structure. Then $\mathcal{E}' = (E, \mathsf{Con}, \vdash')$ is a reversible event structure, where we define $X \otimes \emptyset \vdash' e$ iff $X \vdash e$ (and there are no reverse enablings $X \otimes Y \vdash' \underline{e}$). Moreover, X is a configuration of $\mathcal{E}$ according to Definition 3.5 iff X is a configuration of $\mathcal{E}'$ according to Definition 3.10.*

Our generalised enabling rules are powerful enough that we no longer need the consistency relation.

Proposition 3.12. *Let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ be an RES. Define $\mathsf{Con}' = \mathcal{P}_{\text{fin}}(E)$ and define $\vdash'$ by $X \otimes [Y \cup (E \setminus Z)] \vdash' e^*$ whenever X, Y, Z are such that $X \otimes Y \vdash e^*$, Z is consistent with respect to Con and $X + e^* \subseteq Z$. Then $\mathcal{E}' = (E, \mathsf{Con}', \vdash')$ is an RES, and X is a configuration of $\mathcal{E}$ iff X is a configuration of $\mathcal{E}'$.*

In the light of the previous result, we could dispense with Con altogether in the setting of RESs. However we allow Con as sometimes it may be natural or convenient to identify certain configurations as being consistent or inconsistent, before defining enabling rules in detail.

Example 3.13. Let $E = \{a, b, c\}$, $\mathsf{Con} = \{\{a, c\}, \{b, c\}\}$ plus deducible subsets, and $\emptyset \vdash a$, $\emptyset \vdash b$, $a \vdash c$, $b \vdash c$. Then $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ is a (reversible) event structure where either a or b causes c, and $\{a, b\}$ is inconsistent. We can use the procedure of Proposition 3.12 to convert $\mathcal{E}$ into $\mathcal{E}' = (E, \mathsf{Con}', \vdash')$ where $\mathsf{Con}' = \mathcal{P}_{\text{fin}}(E)$ and $\emptyset \otimes b \vdash' a$, $\emptyset \otimes \{b, c\} \vdash' a$, $\emptyset \otimes a \vdash' b$, $\emptyset \otimes \{a, c\} \vdash' b$, $a \otimes b \vdash' c$, $b \otimes a \vdash' c$.

Configurations are $\emptyset$, $\{a\}$, $\{b\}$, $\{a, c\}$, $\{b, c\}$ for both $\mathcal{E}$ and $\mathcal{E}'$. However in $\mathcal{E}'$ there are two extra consistent sets, namely $\{a, b\}$ and $\{a, b, c\}$.

Note that the converted RES can be optimised by removing $\emptyset \otimes \{b, c\} \vdash' a$ and $\emptyset \otimes \{a, c\} \vdash' b$, since they are implied by $\emptyset \otimes b \vdash' a$ and $\emptyset \otimes a \vdash' b$, respectively.

Definition 3.14. Given configurations X, Y of a reversible event structure $\mathcal{E}$ we let

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

As before, a computation of $\mathcal{E}$ is a computation starting from $\emptyset_{\mathcal{E}}$.

Example 3.15. Consider events a and b in Example 3.4. We have that a causes b so if we wish to achieve causal-order reversing we need to add the following to the definition of $\vdash$: $b \vdash \underline{b}$ and $a \odot b \vdash \underline{a}$. The configuration $\{a, b\}$ can reverse to a by undoing b as allowed by $b \vdash \underline{b}$. But it cannot regress to $\{b\}$ because $a \odot b \vdash \underline{a}$ can only be applied in a configuration that contains a and does not contain b. See Figure 2(i).

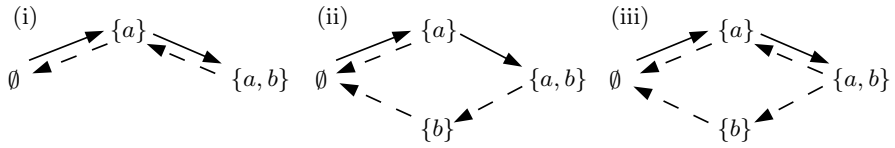

Fig. 2. Configurations and transitions in Example 3.15.

If reversing out of order is required, we instead add to the definition of $\vdash$ in Example 3.4 the following: $a \vdash \underline{a}$ and $b \odot a \vdash \underline{b}$. 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 2(ii), where reverse transitions are 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 2(iii).

We now give an example where we get an infinite configuration as a limit of a non-monotonically increasing sequence.

Example 3.16. Let $\mathcal{E} = (E, \mathsf{Con}, \vdash)$ where $E = \{a_i : i \in \mathbb{N}\} \cup \{b_j : j \in \mathbb{N}\}$ and $\mathsf{Con} = \{a_i, b_0, \ldots, b_j\}$ (any $i, j \in \mathbb{N}$) plus deducible subsets, with

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

The only possible computation sequence is $a_0, b_0, \underline{a}_0, a_1, b_1, \ldots$, with which we can associate a sequence $X_0 = \emptyset$, $X_1 = \{a_0\}, \ldots$. This has limit the infinite set $\{b_j : j \in \mathbb{N}\}$, which is therefore a configuration of $\mathcal{E}$; note that each a_i appears finitely often in the sequence X_n, while each b_j appears cofinitely often.

3.3 Modelling of bonding with events

We now discuss how reaction systems from Section 2 can be given meaning in terms of reversible event structures. Recall that an object of a reaction system can be seen as an undirected graph, where the bases of the object are the nodes and the bonds between the bases are the edges. We shall represent the bonds, and thus the edges of the associated graph, as events. Given bases X, Y of a reaction system each bond $X \cdot Y$ will be denoted by the event xy.

We begin with a simple reaction system in Example 2.1. The events are ca and ab (representing the bonds $C \cdot A$ and $A \cdot B$) and $\mathsf{Con} = \mathcal{P}(\{ca, ab\})$. The bonds are created by $\emptyset \vdash ca$, $ca \vdash ab$, and are broken by $\{ca, ab\} \vdash \underline{ca}$. Note that we do not require here the full generality of the new enabling relation.

The next example is inspired by the ERK signalling pathway [12].

Example 3.17. We describe bonding and unbonding that takes place along a section of the ERK signalling pathway. The molecule A receives a signal P at the top of the pathway by bonding to it. The molecule $P \cdot A$ travels then towards the middle of the pathway where it combines with B. A bond between P and B is then created and the bond between P and A is dissolved thus, in a sense, passing the signal P to B. Once the bond between A and B is broken B is able to pass P towards the bottom of the pathway. The forwards reduction rules are

$$P,\ A \to P \cdot A \qquad P \cdot A, B \to P \cdot A \cdot B \qquad P \cdot A \cdot B \to \overline{P \cdot A \cdot B}$$

and the reverse rules for dissolving the bonds are

$$\overline{P \cdot A \cdot B} \rightsquigarrow \overline{P,\ A \cdot B}\ (\equiv A \cdot B \cdot P) \quad A \cdot B \cdot P \rightsquigarrow A,\ B \cdot P \quad B \cdot P\ \rightsquigarrow B,\ P$$

The events are xy for every $X \cdot Y$ and Con is defined as $\mathcal{P}(\{pa, ab, bp\})$. We derive the following enabling rules from the forwards reduction rules

$$\emptyset \vdash pa \qquad pa \vdash ab \qquad \{pa, ab\} \vdash bp$$

and we obtain the following enabling rules from the reverse reduction rules

$$\{pa, ab, bp\} \vdash \underline{pa} \qquad \{ab, bp\} \oslash pa \vdash \underline{ab} \qquad bp \oslash \{pa, ab\} \vdash \underline{bp}$$

Note the form of the last three rules and how the operator $\oslash$ is used in the last two rules to enforce the order of undoing of pa, ab and bp.

The configurations are $\emptyset, \{pa\}, \{pa, ab\}, \{pa, ab, bp\}, \{ab, bp\}, \{bp\}$, and the creation and dissolving of the bonds happens in the following order: pa, ab, bp, $\underline{pa}, \underline{ab}, \underline{bp}$. We deduce that pa causes ab which causes bp, and we note that the bonds are reversed out of causal order.

We now return to the reaction system in Example 2.3.

Example 3.18. The events are xy for every bond $X \cdot Y$ among the bases X, Y in Example 2.3. We take Con to be the powerset of the set of all events. Step A_i of the transaction either succeeds by bonding to the next step or it fails by bonding

to F for $1 \leq i \leq n$. So we need to express this in the enabling relation by stating that if $a_i a_{i+1}$ takes place then (a) $a_{i+1}a_{i+2}$ can happen if $a_{i+1}f$ did not take place, and (b) $a_{i+1}f$ can happen if $a_{i+1}a_{i+2}$ did not take place for $i < n-1$. This negative information is represented using $\oslash$ in the following two sets of enabling rules. Transaction steps occur as follows:

$$\begin{array}{lll}
\emptyset \oslash i f & \vdash i a_1 & \\
i a_1 \oslash a_1 f & \vdash a_1 a_2 & \\
a_i a_{i+1} \oslash a_{i+1} f & \vdash a_{i+1} a_{i+2} & 1 \leq i \leq n-2 \\
a_{n-1} a_n \oslash a_n f & \vdash a_n s &
\end{array}$$

Fail can bond with the transaction steps as follows:

$$\begin{array}{lll}
i a_1 \oslash a_1 a_2 & \vdash a_1 f & \\
a_1 f & \vdash i f & \\
a_i a_{i+1} \oslash a_{i+1} a_{i+2} & \vdash a_{i+1} f & 1 \leq i \leq n-2 \\
a_{n-1} a_n \oslash a_n s & \vdash a_n f & \\
a_{i+1} f \oslash \{a_{i+1} a_{i+2}, a_{i+2} f\} & \vdash a_i f & 1 \leq i \leq n-2 \\
a_n f & \vdash a_{n-1} f &
\end{array}$$

Next, bonds of the transaction steps are undone so we need to use the full strength of the enabling relation, where $1 \leq i < n$

$$\begin{array}{ll}
\{i a_1, a_1 f, i f\} & \vdash \underline{i a_1} \\
\{a_1 f, i f\} \oslash i a_1 & \vdash \underline{a_1 f} \\
\{a_i a_{i+1}, a_{i+1} f, a_i f\} & \vdash \underline{a_i a_{i+1}} \\
\{a_{i+1} f, a_i f\} \oslash a_i a_{i+1} & \vdash \underline{a_{i+1} f}
\end{array}$$

Consider $\{a_i a_{i+1}, a_{i+1} f, a_i f\}$. Here F is bonded with A_i and A_{i+1}. We require that the bond $A_i \cdot A_{i+1}$ breaks first, and then $A_{i+1} \cdot F$ breaks. We achieve this by requiring that all events $a_i a_{i+1}, a_{i+1} f, a_i f$ are present in order to undo $a_i a_{i+1}$, and we undo $a_{i+1} f$ when $a_{i+1} f, a_i f$ are present and $a_i a_{i+1}$ is not. Correspondingly for $\{i a_1, a_1 f, i f\}$.

Finally, compensation takes place

$$i f \oslash \{i a_1, a_1 f\} \vdash c i \qquad \{i f, c i\} \vdash \underline{i f}$$

We can reach form $\emptyset$ two terminated configurations (where no forwards or reverse transitions are possible), namely $\{i a_1, \ldots, a_i a_{i+1}, \ldots, a_n s\}$ which denotes the successful completion of the transaction, or $\{c i\}$ which is the compensation following the failure.

4 Conclusion

We have introduced simple reaction systems for bonding and illustrated their usefulness with examples taken from software engineering and biochemistry, including long running transactions with compensation, polymer creation by scaffolding proteins, and a signal passing mechanism used by the ERK pathway. We

have proposed reversible event structures, which has not been done before, defined the notion of configuration, and discussed how to give semantics to reaction systems for bonding in terms of reversible event structures.

It remains for future work to clarify the expressive power of reversible event structures and in particular whether they can model reversible process calculi.

Acknowledgements We thank the referees of Reversible Computation 2013 for their comments and suggestions. The second author acknowledges partial support from the JSPS Invitation Fellowship grant S13054.

References

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

[2] L. Cardelli and C. Laneve. Reversible structures. In *Proceedings of CMSB 2011*, pages 131–140. ACM, 2011.

[3] V. Danos and J. Krivine. Reversible communicating systems. In *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.

[4] V. Danos and J. Krivine. Transactions in RCCS. In *Proceedings of CONCUR 2005*, volume 3653 of *LNCS*, pages 398–412. Springer, 2005.

[5] V. Danos and J. Krivine. Formal molecular biology done in CCS-R. In *Proceedings of BioConcur 2003*, volume 180 of *ENTCS*, pages 31–49, 2007.

[6] I. Lanese, C.A. Mezzina, A. Schmitt, and J-B. Stefani. Controlling reversibility in higher-order pi. In *Proceedings of CONCUR 2011*, volume 6901 of *LNCS*, pages 297–311. Springer, 2011.

[7] I. Lanese, C.A. Mezzina, and J-B. Stefani. Reversing higher-order pi. In *Proceedings of CONCUR 2010*, volume 6269 of *LNCS*, pages 478–493. Springer, 2010.

[8] E.M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[9] M. Nielsen, G.D. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.

[10] I.C.C. Phillips and I. Ulidowski. Reversibility and models for concurrency. In *Proceedings of SOS 2007*, volume 192 of *ENTCS*, pages 93–108, 2007.

[11] I.C.C. Phillips and I. Ulidowski. Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming*, 73:70–96, 2007.

[12] I.C.C. Phillips, I. Ulidowski, and S. Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In *Proceedings of Reversible Computation 2012*, volume 7581 of *LNCS*, pages 218–232. Springer, 2013.

[13] R. J. van Glabbeek and G. D. Plotkin. Configuration structures, event structures and Petri nets. *Theoretical Computer Science*, 410(41):4111–4159, 2009.

[14] R.J. van Glabbeek and G.D. Plotkin. Event structures for resolvable conflict. In *Proceedings of MFCS 2004*, volume 3153 of *LNCS*, pages 550–561. Springer, 2004.

[15] D. Varacca and N. Yoshida. Typed event structures and the linear π-calculus. *Theoretical Computer Science*, 411(19):1949–1973, 2010.

[16] G. Winskel. Event structures. In *Advances in Petri Nets 1986*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.

[17] G. Winskel. Events, causality and symmetry. *Computer Journal*, 54(1):42–57, 2011.