A calculus for local reversibility

Stefan Kuhn and Irek Ulidowski

Department of Computer Science, University of Leicester
Leicester, LE1 7RH, United Kingdom
{shk12,iu3}@le.ac.uk

Abstract. We introduce a process calculus with a new prefixing operator that allows us to model locally controlled reversibility. Actions can be undone spontaneously, as in other reversible process calculi, or as pairs of concerted actions, where performing a weak action forces undoing of another action. The new operator in its full generality allows us to model out-of-causal order computation, where effects are undone before their causes are undone, which goes beyond what typical reversible calculi can express. However, the core calculus, with a restricted form of the new operator, is well behaved as it satisfied causal consistency. We demonstrate the usefulness of the calculus by modelling the hydration of formaldehyde in water into methanediol, an industrially important reaction, where the creation and breaking of some bonds are examples of locally controlled out-of-causal order computation.

Keywords: Reversible process calculi, local reversibility, modelling of chemical reactions

1 Introduction

There are many different computation tasks which involve undoing of previously performed steps or actions. Consider a computation where the action a causes the action b, written $a < b$, and where the action c occurs independently of a and b. There are three executions of this computation that preserve *causality*, namely abc, acb and cab. We note that a always comes before b. There are several conceptually different ways of undoing these actions [18]. *Backtracking* is undoing in precisely the reverse order in which they happened. So, undo b undo c undo a is a backtrack of the execution acb. *Reversing* is a more general form of undoing: here actions can be undone in any order provided causality is preserved (meaning that causes cannot be undone before effects). For example, undo c undo b undo a is a reversal of acb for the events a, b and c above.

In biochemistry, however, there are networks of reactions where actions are undone seemingly *out of causal order*. The creation and breaking of molecular bonds between the proteins involved in the ERK signalling pathway is a good example of this phenomenon [16]. Let us assume for simplicity that the creation of molecular bonds is represented by actions 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: undo a, undo b, undo c, which seems to undo the cause a before the effect b. The first process calculus for the out-of-causal order reversible computation was proposed in [16], where the calculus CCSK [13] which is extended with an *execution control mechanism* for managing the pattern and the direction of computation. The control mechanism is external to the processes it controls, and it can have a global scope. Out-of-causal order computation was also studied in [15,14]. Other reversible process calculi were proposed in [4,5,12,13,9,8,10,3].

We introduced informally a novel and purely local in character mechanism for undoing of computation in a short paper [7]. Here, we build a process calculus around this mechanism and give it operational semantics. We then discuss various properties that hold in the calculus. Most importantly, we show that out-of-causal order computation can be modelled in the calculus. Hence, in general, the *causal consistency* property [4] does not hold. There are reachable states that can only be arrived at by a mixture of forward and reverse steps. However, we argue that causal consistency holds in a restricted version of our calculus, thus the full calculus is in effect a "conceptual" extension of a causally consistent reversible process calculus. The benefits of the calculus are shown by modelling hydration of formaldehyde in water. The molecules of formaldehyde and water are modelled as compositions of carbon, oxygen and hydrogen atoms. When composed in parallel, the molecules react and the reactions are represented by sequences of transitions of *concerted actions*. We are able to represent different forms of reversibility, including out-of-causal-order reversibility, and computation can proceed in any directions without without external control.

The novel features of our calculus are introduced via an example of catalytic reaction. Consider two molecules A and B that are only able to bond if assisted by the catalyst C. We assume $A \stackrel{def}{=} (a;p).A'$, $B \stackrel{def}{=} (b,p).B'$ and $C \stackrel{def}{=} (a,b).C'$. We use a new prefix operator $(s;p).P$ where s is a sequence of actions or executed actions and p is a *weak* action. Initially the actions in s, p take place, and then we compute with P. The three molecules can bond by performing synchronously the matching actions according to the function $\gamma(a,a) = c$, $\gamma(b,b) = d$ and $\gamma(p,p) = q$, producing thus new actions c, d and q. A weak action p can be left out resulting in the prefix $(s;p).P$ (as in B and C above). Actions in s can take place in any order, and p can happen if all actions in s have already taken place. Once p takes place, one of the executed actions in s must be undone immediately: this is our new mechanism for triggering reverse computation. We shall model these two almost simultaneous events as a transition of concerted actions. This is a simple but realistic representation of the mechanism of covalent bonding, the most common type of chemical bonds between atoms, hence our calculus is called a *Calculus of Covalent Bonding*.

Returning to our example, we represent the system of molecules A, B and C as $((a;p).A' \mid (b,p).B' \mid (a,b).C') \setminus \{a,b,p\}$, where ' $\mid$ ' is the parallel composition and '$\setminus$' the restriction as in CCS and ACP [11,1]. We note that A and B cannot

interact initially since $\gamma(a,b)$ is not defined. But they can both interact with C:

$$(a;p).A' \mid (b,p).B' \mid (a,b).C' \xrightarrow{c[1]} (a[1];p).A' \mid (b,p).B' \mid (a[1],b).C' \xrightarrow{d[2]} (a[1];p).A' \mid (b[2],p).B' \mid (a[1],b[2]).C'$$

where 1 and 2 are communication keys [13] indicating which pairs of actions created bonds. Molecules A and B can now do p synchronously, producing action q. This causes immediately the breaking of the bond c, which means undoing of actions a in A and C, leaving A and B bonded. We model such pairs of events by pairs of concerted actions:

$$(a[1];p).A' \mid (b[2],p).B' \mid (a[1],b[2]).C' \xrightarrow{\{q[3],\underline{c}[1]\}} (a;p[3]).A' \mid (b[2],p[3]).B' \mid (a,b[2]).C'$$

The bond 3 on weak actions p is unstable and thus gets *promoted* to a stable stronger bond on a and p. Finally, the catalyst dissolves the bond with B:

$$(a;p[3]).A' \mid (b[2],p[3]).B' \mid (a,b[2]).C' \Rightarrow (a[3];p).A' \mid (b[2],p[3]).B' \mid (a,b[2]).C' \xrightarrow{\underline{d}[2]} (a[3];p).A' \mid (b,p[3]).B' \mid (a,b).C'$$

We note that A and B are now bonded although the synchronisation function did not allow it to happen initially. The main consequence of this is that the bond between $a[3]$ and $p[3]$ is *irreversible*, namely it cannot be undone. Looking at the pattern of doing and undoing of bonds we obtain $c[1]d[2]q[3]\underline{c}[1]\underline{d}[2]$. Since creation of bonds c and d causes the bond q, we have here an example of out-of-causal order computation.

Biochemical reactions can also be modelled, for example, with the kappa calculus [6]. Various calculi have also been employed to model biochemical processes (e.g. [5,2]), where the focus was on the modelling the reaction rates in complex networks and their interdependence. On the other hand, the question of how the behaviour of a network emerges out of the behaviour of its components has not been often addressed. An attempt at a structural modelling was [13], where global controllers were used to drive reactions forwards and in reverse. In contrast the calculus introduced in this paper has no global control and the behaviour of a biochemical network emerges from its components.

2 A Calculus of Covalent Bonding

We define the set of (forward) action labels $\mathcal{A}$ which is ranged over by a, b, c, d, e. We partition $\mathcal{A}$ into the set of *strong actions*, written as $\mathcal{SA}$, and the set of *weak actions* $\mathcal{WA}$. Reverse action labels belong to $\underline{\mathcal{A}}$, with typical members $\underline{a}, \underline{b}, \underline{c}, \underline{d}, \underline{e}$, and represent undoing of actions. The set $\mathcal{P}(\mathcal{A} \cup \underline{\mathcal{A}})$ is ranged over by L.

Let $\mathcal{K}$ be an infinite set of *communication keys* (or *keys* for short), ranged over by k, l, m, n. The Cartesian product $\mathcal{A} \times \mathcal{K}$, denoted by $\mathcal{AK}$, represents past

actions, which are written as $a[k]$ for $a \in \mathcal{A}$ and $k \in \mathcal{K}$. Correspondingly, we have the set $\underline{\mathcal{A}}\mathcal{K}$ that represents undoing of past actions. We use α, β to identify actions which are either from $\mathcal{A}$ or $\mathcal{A}\mathcal{K}$. It will be useful to consider sequences of actions or past actions, namely the elements of $(\mathcal{A} \cup \mathcal{A}\mathcal{K})^*$, which are ranged over by s, s' and sequences of purely past actions, namely the elements of $\mathcal{A}\mathcal{K}^*$, which are ranged over by t, t'. The empty sequence is denoted by ϵ and $\alpha : s$ is the sequence with the head α and the tail s.

We shall also use two sets of auxiliary action labels, namely the set $(\mathcal{A}) = \{(a) \mid a \in \mathcal{A}\}$, and its product with the set of keys, namely $(\mathcal{A})\mathcal{K}$.

We now define a Calculus of Covalent Bonding, or CCB for sort. The syntax is given below, where $f : \mathcal{A} \to \mathcal{A}$. We have a set of process identifiers (constants) $\mathcal{PI}$, with typical elements S, T, which contains the deadlocked process $\mathbf{0}$. The set of CCB closed terms is denoted by Proc. We shall refer to closed terms as processes, and let P, Q, R to range over processes. Each process identifier S has a defining equation $S \stackrel{def}{=} P$.

$$P ::= S \;\mid\; (s;b).P \;\mid\; P \mid Q \;\mid\; P\backslash L \;\mid\; P[f]$$

We have a prefixing operator $(s;b).P$, where s is a non-empty sequence of actions or past actions. The actions in s, which have not happened yet, can happen in any order. The action b is a weak action in $\mathcal{WA}$ and it can only happen after all actions in s have taken place. Performing b then forces undoing one of the past actions in s (using the concert rule in Figure 4). The action after the ; in $(s;b).P$ can be omitted, in which case the prefixing is simply $(s).P$, and is the prefixing in [16]. In this form, one of the actions in s may be a weak action from $\mathcal{WA}$. If s is a single element sequence, then the action is a strong action in $\mathcal{SA}$ and the prefixing operator is the prefixing of CCS [11]. We often omit trailing $\mathbf{0}$s so, for example, $(s).\mathbf{0}$ is written as (s). All actions in s in $(s;b).P$ are strong actions (in $\mathcal{SA}$).

$P \mid Q$ represents processes P and Q which can perform actions or reverse actions on their own, or which can interact with each other according to a communication function γ (much like in ACP [1]). Or, they can perform a pair of the so-called *concerted actions*, which is the new feature of our calculus. We also have the usual restriction (encapsulation) operator $\backslash L$, where L is a set of labels, and the relabelling operator $[f]$.

The forward and reverse SOS rules for CCB are in Figures 2-5, where the rules in Figures 2-3 are influenced by [13]. Since we do not use the relabelling operator in the systems modelled in this paper, we omit all SOS rules for $[f]$. Note that the reverse rules in Figure 3 are simply the symmetric versions of the corresponding forward rules.

We use two predicates, $\mathsf{std}(P) : \mathcal{P}(\mathsf{Proc})$ and $\mathsf{fsh}[m](P) : \mathcal{P}(\mathcal{K} \times \mathsf{Proc})$ in our SOS rules. They are defined in Figure 1. Two further auxiliary functions, $\mathsf{k}(i) : (\mathcal{A} \cup \mathcal{A}\mathcal{K})^* \to \mathcal{P}(\mathcal{K})$ and $\mathsf{keys}(P) : \mathsf{Proc} \to \mathcal{P}(\mathcal{K})$, are also used. The function $\mathsf{k}()$ is defined as follows: $\mathsf{k}(\epsilon) = \emptyset$; $\mathsf{k}(\alpha : s) = \{l\} \cup \mathsf{k}(s)$ if $\alpha = a[l], a \in \mathcal{A}, l \in \mathcal{K}$; and $\mathsf{k}(\alpha : s) = \mathsf{k}(s)$ if $\alpha \in \mathcal{A}$. The function $\mathsf{keys}()$ is defined as $\mathsf{keys}(\mathbf{0}) = \emptyset$; $\mathsf{keys}(S) = \mathsf{keys}(P)$ if $S \stackrel{def}{=} P$; $\mathsf{keys}((s;b).P) = \mathsf{k}(s) \cup \mathsf{k}(b) \cup \mathsf{keys}(P)$; $\mathsf{keys}(P \mid Q) =$

$$\frac{}{\mathsf{std}(\mathbf{0})} \qquad ;\frac{\mathsf{std}(P)}{\mathsf{std}(S)}\ S \stackrel{def}{=} P \qquad \frac{\mathsf{k}(s)=\emptyset \quad \mathsf{std}(P)}{\mathsf{std}((s;b).P)} \qquad \frac{\mathsf{std}(P) \quad \mathsf{std}(Q)}{\mathsf{std}(P \mid Q)} \qquad \frac{\mathsf{std}(P)}{\mathsf{std}(P \setminus L)}$$

$$\frac{}{\mathsf{fsh}[m](\mathbf{0})} \qquad \frac{\mathsf{fsh}[m](P)}{\mathsf{fsh}[m](S)}\ S \stackrel{def}{=} P \qquad \frac{m \notin \mathsf{k}(s)\ m \neq n\ \mathsf{fsh}[m](P)}{\mathsf{fsh}[m]((s;b[n]).P)} \qquad \frac{m \notin \mathsf{k}(s) \quad \mathsf{fsh}[m](P)}{\mathsf{fsh}[m]((s;b).P)}$$

$$\frac{\mathsf{fsh}[m](P) \quad \mathsf{fsh}[m](Q)}{\mathsf{fsh}[m](P \mid Q)} \qquad \frac{\mathsf{fsh}[m](P)}{\mathsf{fsh}[m](P \setminus L)}$$

Fig. 1. Predicates std and fsh

$$\text{act1}\ \frac{\mathsf{std}(X) \quad \mathsf{fsh}[k](s)}{(s,a;b).X \xrightarrow{a[k]} (s,a[k];b).X} \qquad \text{act2}\ \frac{X \xrightarrow{a[k]} X' \quad \mathsf{fsh}[k](t)}{(t;b).X \xrightarrow{a[k]} (t;b).X'}$$

$$\text{par}\ \frac{X \xrightarrow{a[k]} X' \quad \mathsf{fsh}[k](Y)}{X \mid Y \xrightarrow{a[k]} X' \mid Y} \qquad \text{com}\ \frac{X \xrightarrow{a[k]} X' \quad Y \xrightarrow{b[k]} Y'}{X \mid Y \xrightarrow{c[k]} X' \mid Y'}\ \gamma(a,b)=c$$

$$\text{res}\ \frac{X \xrightarrow{a[k]} X'}{X\backslash L \xrightarrow{a[k]} X'\backslash L}\ a \notin L \qquad \text{con}\ \frac{X \xrightarrow{a[k]} X'}{S \xrightarrow{a[k]} X'}\ S \stackrel{def}{=} X$$

Fig. 2. Forward SOS rules

$\mathsf{keys}(P) \cup \mathsf{keys}(Q)$; and $\mathsf{keys}(P\backslash L) = \mathsf{keys}(P)$. Informally $\mathsf{keys}(P)$ associates with each P the set of its keys. A process P is standard, written $\mathsf{std}(P)$, if it contains no past actions. A key n is fresh in Q, written $\mathsf{fsh}[n](Q)$, if n is not used in Q. We extend the notion of fresh keys to the sequences of actions and past actions s and t via the function $\mathsf{k}()$.

The semantics of CCB is given by the labelled transition system (lts),

$$(\mathsf{Proc}, L, \rightarrow : \subseteq \mathsf{Proc} \times L \times \mathsf{Proc})$$

where the set of action labels L is $\mathcal{AK} \cup \underline{\mathcal{A}}\mathcal{K} \cup (\mathcal{AK} \times \underline{\mathcal{A}}\mathcal{K})$: it contains the pairs of concerted actions $\mathcal{AK} \times \underline{\mathcal{A}}\mathcal{K}$ (see Figure 4) as well as actions and past actions. The transition relation $\rightarrow$ is the least relation defined by our SOS rules and reduction rules in Definition 2.

Figure 4 contains the rule concert that defines when a pair of concerted actions takes place. We also have two auxiliary rules aux1 and aux2 which define the auxiliary transition relations needed in the concert rule. Note that aux1 and aux2 define transitions with the auxiliary labels $(b)[k]$ for all $(b) \in \mathcal{A}$ and $k \in \mathcal{K}$. Overall, transitions are labelled with $a[k] \in \mathcal{AK}$, or with $\underline{b}[l] \in \underline{\mathcal{A}}\mathcal{K}$, or with concerted pairs $\{a[k], \underline{b}[l]\}$. Note that the concert rule uses *lookahead* [17].

$$\text{rev act1}\ \frac{\mathsf{std}(X) \quad \mathsf{fsh}[k](s)}{(s, a[k]; b).X \xrightarrow{\underline{a}[k]} (s, a; b).X} \qquad \text{rev act2}\ \frac{X \xrightarrow{\underline{a}[k]} X' \quad \mathsf{fsh}[k](t)}{(t; b).X \xrightarrow{\underline{a}[k]} (t; b).X'}$$

$$\text{rev par}\ \frac{X \xrightarrow{\underline{a}[k]} X' \quad \mathsf{fsh}[k](Y)}{X \mid Y \xrightarrow{\underline{a}[k]} X' \mid Y} \qquad \text{rev com}\ \frac{X \xrightarrow{\underline{a}[k]} X' \quad Y \xrightarrow{\underline{b}[k]} Y'}{X \mid Y \xrightarrow{\underline{c}[k]} X' \mid Y'}\ \gamma(a, b) = c$$

$$\text{rev res}\ \frac{X \xrightarrow{\underline{a}[k]} X'}{X\backslash L \xrightarrow{\underline{a}[k]} X'\backslash L}\ a \notin L \qquad \text{rev con}\ \frac{X \xrightarrow{\underline{a}[k]} X'}{X \xrightarrow{\underline{a}[k]} S}\ S \stackrel{def}{=} X'$$

Fig. 3. Reverse SOS rules

$$\text{aux1}\ \frac{\mathsf{std}(X) \quad \mathsf{fsh}[k](t)}{(t; b).X \xrightarrow{(b)[k]} (t; b[k]).X} \qquad \text{aux2}\ \frac{X \xrightarrow{(b)[k]} X' \quad \mathsf{fsh}[k](t)}{(t; a).X \xrightarrow{(b)[k]} (t; a).X'}$$

$$\text{concert}\ \frac{X \xrightarrow{(a)[k]} X' \quad X' \xrightarrow{\underline{b}[l]} X'' \qquad Y \xrightarrow{\alpha[k]} Y' \quad Y' \xrightarrow{\underline{d}[l]} Y''}{X \mid Y \xrightarrow{\{e[k], \underline{f}[l]\}} X'' \mid Y''}$$

$$\text{concert act}\ \frac{X \xrightarrow{\{a[k], \underline{b}[l]\}} X' \quad \mathsf{fsh}[k](t)}{(t; a).X \xrightarrow{\{a[k], \underline{b}[l]\}} (t; a).X'}$$

$$\text{concert par}\ \frac{X \xrightarrow{\{a[k], \underline{b}[l]\}} X' \quad \mathsf{fsh}[k](Y)}{X \mid Y \xrightarrow{\{a[k], \underline{b}[l]\}} X' \mid Y} \qquad \text{concert res}\ \frac{X \xrightarrow{\{a[k], \underline{b}[l]\}} X'}{X\backslash L \xrightarrow{\{a[k], \underline{b}[l]\}} X'\backslash L}$$

Fig. 4. SOS rules for concerted transitions. Rule concert applies if 1. α is c or (c) and $\gamma(a, c) = e$ for some $c \in \mathcal{A}$, and 2. $\gamma(b, d) = f$. Rule concert res applies if $a, \underline{b} \notin L \cup (L)$.

We also need a reduction relation to define *promotion* of actions. First we define *free names* of processes.

Definition 1. Function fn, with $\mathsf{fn} : \mathsf{Proc} \to \mathcal{P}(\mathcal{K})$, is defined as follows: $\mathsf{fn}(\mathbf{0}) = \emptyset$, $\mathsf{fn}(S) = \mathsf{fn}(P)$ if $S \stackrel{def}{=} P$, $\mathsf{fn}((\alpha : s; b).P) = \{\alpha\} \cup \mathsf{fn}(s; b).P)$, $\mathsf{fn}((a; b).P) = \{a, b\} \cup \mathsf{fn}(P)$, $\mathsf{fn}(P \mid Q) = \mathsf{fn}(P) \cup \mathsf{fn}(Q)$ and $\mathsf{fn}(P\backslash L) = \mathsf{fn}(P)\backslash L$.

Definition 2. The reduction relation $\Rightarrow$ is the smallest reflexive and transitive binary relation that satisfies the following rules: (red1) $P \mid Q \Rightarrow Q \mid P$, (red2) $P \mid (Q \mid R) \Rightarrow (P \mid Q) \mid R$, (red3) $(P \mid Q) \mid R \Rightarrow P \mid (Q \mid R)$, (red4) $P \mid \mathbf{0} \Rightarrow P$, (red5) $(P \mid Q)\backslash L \Rightarrow P\backslash L \mid Q$ if $\mathsf{fn}(Q) \cap L = \emptyset$, (red6) $P\backslash L \mid Q \Rightarrow (P \mid Q)\backslash L$ if $\mathsf{fn}(Q) \cap L = \emptyset$, (red7) $(s; b).P\backslash(s'; b).P$ if s' is a permutation of s, (prom) $(a : t; b[k]) \Rightarrow (a[k] : t; b)$ if $a \in \mathcal{SA}, b \in \mathcal{WA}$, (move) $(a : b[k] : s) \Rightarrow (a[k] : b : s)$ if $a \in \mathcal{SA}, b \in \mathcal{WA}$, where $t \in \mathcal{AK}^*$ and $s \in (\mathcal{A} \cup \mathcal{AK})^*$.

$$\text{sc}\ \frac{X \Rightarrow^* Y \quad Y \stackrel{\mu}{\to} Y' \quad Y' \Rightarrow^* X'}{X \stackrel{\mu}{\to} X'} \qquad \text{rev sc}\ \frac{X \Rightarrow^* Y \quad Y \stackrel{\mu}{\to} Y' \quad Y' \Rightarrow^* X'}{X \stackrel{\mu}{\to} X'}$$

Fig. 5. Structural congruence rules

We have two promotion rules in Definition 2. The rule prom promotes a weak bond to a strong bond. Since weak bonds are only temporary they get replaced by bonds on strong actions as soon as these become available. In more detail, after a bond is created on the weak action b another bond is broken at the same location involving a strong action, here a. This pair of concerted actions $\{b[k], a[l]\}$, for some l, results in $(a : t; b[k])$, which is subjected immediately to bond promotion from a weak b to a strong a, giving us $(a[k] : t; b)$. Now weak b can bond again. We have another rule move which promotes correspondingly a weak bond b to a strong a. In order to model what happens in chemical reactions more faithfully, we assume that prom and move are used as soon as they becomes applicable. We also have the usual structural congruence rules (sc and rev sc) in Figure 5, where $\mu \in \mathcal{AK} \cup \underline{\mathcal{A}}\mathcal{K} \cup (\mathcal{AK} \times \underline{\mathcal{A}}\mathcal{K})$, which combine potentially several reductions (including prom reductions) with transitions.

Definition 3. A process P is *consistent* if $\mathsf{std}(P)$ or $Q \to^* P$ for some process Q such that $\mathsf{std}(Q)$.

Example 1. Consider the process $(a; b) \mid a \mid b$ with $\gamma(a, a) = c$ and $\gamma(b, b) = d$. After the initial synchronisation of actions a, which produces the transition $c[1]$, we have a transition with a pair of concerted actions by rule concert in Figure 4

$$(a[1]; b) \mid a[1] \mid b \xrightarrow{\{d[2], \underline{c}[1]\}} (a; b[2]) \mid a \mid b[2]$$

since $(a[1]; b) \xrightarrow{(b[2])} (a[1]; b[2]) \xrightarrow{\underline{a}[1]} (a; b[2])$ and $a[1] \mid b \xrightarrow{b[2]} a[1] \mid b[2] \xrightarrow{\underline{a}[1]} a \mid b[2]$.

Example 2. Consider $(a[1]; b) \mid (a[1]; b) \mid e$ with $\gamma(a, a) = c$ and $\gamma(b, b) = d$. We clearly have the following pair of concerted actions

$$(a[1]; b) \mid (a[1]; b) \mid e \xrightarrow{\{d[2], \underline{c}[1]\}} (a; b[2]) \mid (a; b[2]) \mid e.$$

There are processes with weak actions that can potentially communicate but there are no concerted actions due to our SOS rules:

Example 3. Consider $(a[1]; b) \mid (e[2]; b) \mid (a[1], e[2])$ with $\gamma(a, a) = c$ and $\gamma(b, b) = d$. It cannot perform any concerted actions: Although $(a[1]; b) \xrightarrow{(b)[l]} \xrightarrow{\underline{a}[1]} (a; b[l])$, for any l different from 1 and 2, but $(e[2]; b) \mid (a[1], e[2])$ cannot perform the $(b[l])$ transition since there are no SOS rules for parallel composition and auxiliary actions (b). This forces us to treat $(a[1]; b)$ and $(e[2]; b)$ as X and Y in the concert rule, respectively, and we notice that we cannot undo a communication on a or e.

Example 4. The transition $(a[1];b) \mid a[1] \mid b \xrightarrow{\{d[2],\underline{c}[1]\}} (a;b[2]) \mid a \mid b[2]$ from Example 1 is followed by the application of the reduction rule prom that moves the bond 2 from the weak b to the strong a:

$$(a;b[2]) \mid a \mid b[2] \Rightarrow (a[2];b) \mid a \mid b[2]$$

As a result, we can bond on the weak b again and, importantly, the $a[2]$ to $b[2]$ bond is irreversible as $\gamma(a,b)$ is undefined. Note that reaching this bond by computing forwards alone is not possible.

3 Properties of CCB

In this section we establish some properties of the lts for CCB. We start by showing the expected properties of keys, namely that when an action takes place it uses a fresh key, and when a past action is undone its key is removed from the resulting process. We also show that the reverse part of the transition relation inverts the forward part.

Proposition 1. *Let P be consistent. Then*

1. *If $P \xrightarrow{a[k]} Q$ then $k \notin \mathsf{keys}(P)$ and $\mathsf{keys}(Q) = \mathsf{keys}(P) \cup \{k\}$ for all Q.*
2. *If $P \xrightarrow{\underline{a}[k]} Q$ then $k \in \mathsf{keys}(P)$ and $\mathsf{keys}(Q) = \mathsf{keys}(P) \setminus \{k\}$ for all Q.*
3. *$P \xrightarrow{a[k]} P'$ if and only if $P' \xrightarrow{\underline{a}[k]} P$ for all P'.*

Next, we introduce some notation. We define a new transition relation $\longmapsto$ by $P \overset{a[k]}{\longmapsto} Q$ if $P \xrightarrow{a[k]} Q$ or $P \xrightarrow{\underline{a}[k]} Q$. Process P is called the *source* and Q the *target* of $P \overset{a[k]}{\longmapsto} Q$. We will use $t, t', t_1, \ldots$ to denote transitions, for example $t : P \overset{a[k]}{\longmapsto} Q$. Two $\longmapsto$ transitions are *coinitial* if they have the same source, and they are *cofinal* if their targets are identical.

We define when two transitions are concurrent.

Definition 4. Two coinitial transitions $P \overset{a[k]}{\longmapsto} P'$ and $P \overset{b[l]}{\longmapsto} P''$ are *concurrent* if and only if there exists $M \neq P$ such that $P' \overset{b[l]}{\longmapsto} M$ and $P'' \overset{a[k]}{\longmapsto} M$.

Note that two concurrent transitions are coinitial and, together with the two transitions (with the target M) required by Definition 4, they form a "diamond" structure with the nodes P, P', P'' and M.

When transitions in Definition 4 are forward, we may not be able to complete the diamond as the following example shows. In such case, we say that the transitions are in *conflict*. Consider $P \overset{def}{=} (a).\mathbf{0} \mid (b).\mathbf{0} \mid (b).\mathbf{0}$ with $\gamma(a,b) = c$. The two coinitial transitions below are in conflict:

$$(a).\mathbf{0} \mid (b).\mathbf{0} \mid (b).\mathbf{0} \xrightarrow{c[1]} (a[1]).\mathbf{0} \mid (b[1]).\mathbf{0} \mid (b).\mathbf{0}$$
$$(a).\mathbf{0} \mid (b).\mathbf{0} \mid (b).\mathbf{0} \xrightarrow{c[2]} (a[2]).\mathbf{0} \mid (b).\mathbf{0} \mid (b[2]).\mathbf{0}$$

However, coinitial reverse transitions are concurrent:

$$act_f\ \frac{}{(a:s).X \xrightarrow{a}_f (s).X} \qquad par_f\ \frac{X \xrightarrow{a}_f X'}{X \mid Y \xrightarrow{a}_f X' \mid Y}$$

$$com_f\ \frac{X \xrightarrow{a}_f X' \quad Y \xrightarrow{b}_f Y'}{X \mid Y \xrightarrow{c}_f X' \mid Y'}\ \gamma(a,b) = c \qquad es_f\ \frac{X \xrightarrow{a}_f X'}{X\backslash L \xrightarrow{a}_f X'\backslash L}\ a \notin L$$

$$con_f\ \frac{X \xrightarrow{a}_f X'}{S \xrightarrow{a}_f X'}\ S \stackrel{def}{=} X \qquad sc\ \frac{X \Rightarrow^* Y \quad Y \xrightarrow{a}_f Y' \quad Y' \Rightarrow^* X'}{X \xrightarrow{a}_f X'}$$

Fig. 6. Syntax and SOS rules for CCB_f.

Proposition 2 (Reverse Diamond). *Let P be a consistent process and let $t' : P \xrightarrow{\underline{a}[k]} P'$ and $t'' : P \xrightarrow{\underline{b}[l]} P''$ with $l \neq k$. Then t' and t'' are concurrent.*

Coinitial forward transitions are concurrent if they result in cofinal computations:

Proposition 3 (Forward Diamond). *If P is a consistent process and $t_1 \equiv P \xrightarrow{a[k]} P'$, $t_2 \equiv P \xrightarrow{b[l]} P''$, with $l \neq k$, and $P' \to^* T$ and $P'' \to^* T$, for some T, then there is M such that $P' \xrightarrow{b[l]} M$, $P'' \xrightarrow{a[k]} M$ and $M \to^* T$.*

3.1 CCB without weak actions

We now discuss the main properties of the sub-calculus of CCB that uses the simplified form of prefixing $(s).P$: namely without a weak action b following ; in $(s;b).P$. We call this calculus CCB_s. Its SOS rules are as for CCB except that the rules in Figure 4 do not apply as there are no weak actions. We shall also consider the forward-only version of CCB_s called CCB_f. The syntax of CCB_f is $P ::= S \mid (s).P \mid P \mid Q \mid P\backslash L$ and the SOS rules are given in Figure 6 (relabelling is not included); we also have the reduction rules from Definition 2 which, together with rules in Figure 6, generate the transition relation $\to_f$ for CCB_f. Note that we do not record past actions (act_f rule); hence CCB_f is similar to the core of ACP. We note that CCB_s is different from CCSK [12,13] as it uses multiset prefixing and ACP-like communication.

We show firstly that $\to$ for CCB_s is essentially conservative over $\to_f$. A process of CCB_s is converted to a CCB_f process by "pruning" past actions:

Definition 5. The pruning map $\pi : \mathsf{Proc}_{\mathrm{CCB}_s} \to \mathsf{Proc}_{\mathrm{CCB}_f}$ is defined as follows, where $t \in \mathcal{AK}^*$ and $s \in \mathcal{A}^*$:

$$\pi(\mathbf{0}) = \mathbf{0} \qquad \pi((s,t).P) = (s).\pi(P) \qquad \pi((t).P) = \pi(P)$$
$$\pi(P \mid Q) = \pi(P) \mid \pi(Q) \qquad \pi(P \setminus L) = \pi(P) \setminus L \qquad \pi(S) = \pi(P) \text{ if } S \stackrel{def}{=} P$$

Theorem 1 (Conservation). *Let* $P \in \mathsf{Proc}_{\mathrm{CCB}_s}$.
1. If $P \xrightarrow{\mu[k]} Q$ *then* $\pi(P) \xrightarrow{\mu}_f \pi(Q)$.
2. If $\pi(P) \xrightarrow{\mu}_f Q$, *then for any* $k \in \mathcal{K} \backslash \mathsf{keys}(P)$ *there is* Q' *such that* $P \xrightarrow{\mu[k]} Q'$ *and* $\pi(Q') = Q$.

We now consider the causal consistency property [4] in CCB_s. We define when a set of keys is a cause for another key:

Definition 6. The set of causes of a key k in P is defined as follows:

$$cau(\mathbf{0}, k) = \emptyset \qquad cau(P \mid Q, k) = cau(P, k) \cup cau(Q, k)$$
$$cau((s).P, k) = \mathsf{k}(s) \cup cau(P, k) \text{ if } k \in \mathsf{keys}(P) \qquad cau((\mu[k] : s).P, k) = \emptyset$$
$$cau((s).P, k) = \emptyset \text{ if } k \notin \mathsf{keys}(P) \qquad cau(S) = cau(P) \text{ if } S \stackrel{def}{=} P$$
$$cau(P \backslash L, k) = cau(P, k)$$

If one of two coinitial transitions is forward and the other reverse, either they are concurrent or the forward transition depends causally on the reverse one. The following result holds in the full calculus CCB:

Proposition 4. If $t_1 \equiv P \xrightarrow{\mu[k]} P'$ and $t_2 \equiv P \xrightarrow{\nu[l]} P''$, then either t_1 and t_2 are concurrent or $k \in cau(P'', l)$.

We introduce a trace: a sequence of pairwise composable forward and reverse transitions over CCB_s. Traces are ranged over by $\sigma, \sigma', \sigma_1, \ldots$. Two transitions are composable if the target of the first transition is the source of the second transition. The composition of transitions is denoted by ';'. We denote the reverse transition corresponding to a forward transition t (and the forward transition corresponding to a reverse transition t) as $t^\bullet$. Similarly to denoting reverse transitions by $^\bullet$, we denote the reverse trace of σ as $\sigma^\bullet$. The empty trace with the source P is written as ϵ_P. We can now define causal equivalence between traces.

Definition 7. *Causally equivalent* traces are defined by the least equivalence relation $\asymp$ which is closed under composition and obeys the following rules, where t_1 is $P \xmapsto{a[k]} Q$, t_2 is $P \xmapsto{b[l]} R$, d_1 is $Q \xmapsto{b[l]} S$ and d_2 is $R \xmapsto{a[k]} S$:

$$t_1; d_1 \asymp t_2; d_2 \qquad t; t^\bullet \asymp \epsilon_{source(t)} \qquad t^\bullet; t \asymp \epsilon_{target(t)}$$

The first rule in Definition 7 states that the concurrent transitions t_1 and t_2 are causally independent, hence they can happen in any order. The trace $t_1; d_1$ forms a diamond with $t_2; d_2$, so the traces are causally equivalent. The remaining rules state that doing a transition and its reverse version is the same as doing nothing.

The next two results are needed to prove causal consistency for CCB_s; they follow closely [4]. The first states that any computation has a causally equivalent version in which we first compute in reverse for a while and then we only compute forwards. The second result says that a trace which has a forward-only coinitial and cofinal and causally equivalent trace can always be shortened to a forward-only trace. Then, we have the second important result for CCB_s.

Proposition 5 (Rearrangement). *If σ is a trace then there exist forward traces σ_1 and σ_2 such that $\sigma \asymp \sigma_1^\bullet; \sigma_2$.*

Proposition 6 (Shortening). *If σ_1 and σ_2 are coinitial and cofinal traces, with σ_2 forward, then there exists a forward trace σ_1' of length at most that of σ_1 such that $\sigma_1' \asymp \sigma_2$.*

Theorem 2 (Causal consistency). *Let σ_1 and σ_2 be traces. Then $\sigma_1 \asymp \sigma_2$ if and only if σ_1 and σ_2 are coinitial and cofinal.*

One of the consequences of causal consistency for sub-calculus CCB_s concerns reachability: any state that can be reached from a standard process during an arbitrary computation can be reached by computing forwards alone. This property is not valid in the full calculus CCB as can be seen in the Introduction and in Example 4. The next section explores some properties of concerted transitions.

3.2 Concerted transitions

The properties of keys corresponding to those in parts 1 and 2 of Proposition 1 hold also for the concerted transitions in CCB.

Proposition 7. *Let P be consistent. If $P \xrightarrow{\{\mu[k],\underline{\nu}[l]\}} Q$ then $k \notin \mathsf{keys}(P)$, $l \in \mathsf{keys}(P)$ and $\mathsf{keys}(Q) = \mathsf{keys}(P) \cup \{k\} \setminus \{l\}$ for all Q.*

The property corresponding to part 3 of Proposition 1, namely $P \xrightarrow{\{\mu[k],\underline{\nu}[l]\}} P'$ if and only if $P' \xrightarrow{\{\nu[l],\underline{\mu}[k]\}} P$ does not hold in general but only in certain circumstances. Consider $(a[k];b).Q \mid R$ and c, d such that $\gamma(a,c) = d = \gamma(b,c)$ with $R \xrightarrow{c[l]} R'$ and $R' \xrightarrow{\underline{c}[k]} R''$. We obtain, by concert and prom rules,

$$(a[k];b).Q \mid R \xrightarrow{\{d[l],\underline{d}[k]\}} (a;b[l]).Q \mid R'' \Rightarrow (a[l];b).Q \mid R''$$

Since $R'' \xrightarrow{c[k]} R' \xrightarrow{\underline{c}[l]} R$, we get, again by concert and prom rules

$$(a[l];b).Q \mid R'' \xrightarrow{\{d[k],\underline{d}[l]\}} (a;b[l]).Q \mid R \Rightarrow (a[k];b).Q \mid R$$

This gives us the following result:

Proposition 8. *Consider $(a[k];b).Q$ for any Q and c, d such that $\gamma(a,c) = d = \gamma(b,c)$. There exist R, S and l such that $(a[k];b).Q \mid R \xrightarrow{\{d[l],\underline{d}[k]\}} S$ if and only if $S \xrightarrow{\{d[k],\underline{d}[l]\}} (a[k];b).Q \mid R$.*

4 The hydration of formaldehyde in water

In this section we model the hydration of formaldehyde in an aqueous solution. Formaldehyde is a good preservative and is well known for its use in preserving specimen samples. It also serves as an important building block in many industrial processes and is therefore produced in large quantities. The reaction is shown in Figure 7: two water molecules and formaldehyde are on the left and the resulting compound, methanediol, and one molecule of water is on the right. Note that the carbon atom is not shown in line with a common convention. It resides at the point where the lines from the oxygen and the hydrogens meet.

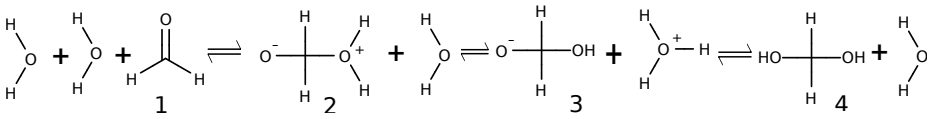

Fig. 7. The most common path through hydration of formaldehyde

The carbon in the formaldehyde has a positive charge and the oxygen in the water is attracted by the carbon and forms a bond to the carbon. This bond is formed out of the electrons of one of the lone pairs of the oxygen. Since the carbon cannot have more than four bonds this reaction is compensated by the double bond in the formaldehyde becoming a single bond and the electrons from the double bond forming a lone pair on the oxygen (which now has three lone pairs). These movements are concerted, namely they happen together without a stable intermediate state and cannot be separated. The resulting intermediate (denoted by 2 in Figure 7) has one oxygen which is negatively charged, whereas the other oxygen is positively charged. The intermediate 2 abstracts one of the hydrogens to the positively charged oxygen. This leads to the intermediate 3 and a H_3O molecule, a water with and additional hydrogen and a positively charged oxygen. One of these hydrogens can be re-donated to the negatively charged oxygen. We then get the final products: methanediol and a molecule of water.

4.1 The most common path through the reaction

We shall represent the formaldehyde molecule and the two water molecules as appropriate compositions of hydrogen, oxygen and carbon. We use our general prefixing operator, noting that O has no weak action:

$$H \stackrel{def}{=} (h;p).H' \qquad O \stackrel{def}{=} (o,o,n).O' \qquad C \stackrel{def}{=} (c,c,c,c;p).C'$$

Carbon has four strong actions c, representing the potential for four covalent bonds, and a weak action p, standing for a positive partial charge. The oxygen is modelled as a flexible element with up to 3 bonds. The action n represents the potential for a negative partial charge. The hydrogen has one strong bond h and

one weak bond p. We employ subscripts to denote individual copies of actions and atoms. The synchronisation function is defined as follows: $\gamma(c_i, h_j) = c_i h_j$ for $i \in \{1, \ldots, 4\}$ and $j \in \{1, \ldots, 6\}$; $\gamma(c_i, n) = c_i n$ for $i \in \{1, \ldots, 4\}$; $\gamma(h_i, n) = h_i n$ and $\gamma(h_i, o_j) = h_i o_j$ for $i, j \in \{1, \ldots, 6\}$; and $\gamma(n, p) = np$.

The three molecules of the reaction are placed in parallel: $CH_2O \mid H_2O \mid H_2O$. Each molecule is a parallel composition of its atoms, and we use restriction to force the atoms to bond together (and in some cases to stay bonded). We also restrict actions n, p so that they can only happen together. The reaction starts from the following initial configuration, where keys $1, \ldots, 8$ specify the bonds existing initially among the atoms of formaldehyde and the two waters.

$$\begin{aligned}&\big((c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \mid (o_1[3], o_2[4], n).O_1'\\ &\mid (h_3[5]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3[5], o_4[6], n).O_2'\\ &\mid (h_5[7]; p).H_5' \mid (h_6[8]; p).H_6' \mid (o_5[7], o_6[8], n).O_3'\big) \setminus L\end{aligned}$$

We have grouped all restricted actions at the outer-most level and L is $\{c_1, c_2, c_3, c_4, h_1, h_2, h_3, h_4, h_5, h_6, o_1, o_2, o_3, o_4, o_5, o_6, n, p, \underline{c_1 h_1}, \underline{c_2 h_2}\}$. Apart from the restrictions of the appropriate versions of the c_i, o_j and h_k actions, we also restrict $\underline{c_i h_i}$ for $i \in \{1, 2\}$. It prevents breaking any of the bonds between C_1 and its hydrogens H_1, H_2. This serves two purposes. Firstly, it makes sure that once we have done the p action of the carbon, we will break one of the bonds between the carbon and the oxygen. This is justified since in reality it is one of the oxygen bonds which is broken. Secondly, it also prevents O_2 or O_3 from abstracting H_1 or H_2 from the carbon.

We now model the reactions in Figure 7. The first step is the n, p reaction between C_1 and O_2 or O_3. There are other n, p reactions that are allowed by our model: we describe them in Section 4.2. We assume that O_2 bonds with C_1 with key 9, followed immediately by breaking of the bond 3 or 4. Note that breaking of 1 or 2 is not possible because of the restriction on breaking $c_1 h_1$ and $c_2 h_2$. Without a loss of generality we break bond 4. These two partial reactions give us a concerted transition: we create the bond $np[9]$ and break the bond $\underline{c_4 o_2}[4]$:

$$\begin{aligned}&\xrightarrow{\{np[9], \underline{c_4 o_2}[4]\}} \big((c_1[1], c_2[2], c_3[3], c_4; p[9]).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \mid\\ &\qquad (o_1[3], o_2, n).O_1' \mid (h_3[5]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3[5], o_4[6], n[9]).O_2'\\ &\mid (h_5[7]; p).H_5' \mid (h_6[8]; p).H_6' \mid (o_5[7], o_6[8], n).O_3'\big) \setminus L\end{aligned}$$

Next, we promote the bond 9 of the carbon on a weak p to a stronger bond on c_4, which has become available. Using prom in Definition 2 we obtain

$$\begin{aligned}&\Rightarrow \big((c_1[1], c_2[2], c_3[3], c_4[9]; p).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \mid (o_1[3], o_2, n).O_1'\\ &\mid (h_3[5]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3[5], o_4[6], n[9]).O_2'\\ &\mid (h_5[7]; p).H_5' \mid (h_6[8]; p).H_6' \mid (o_5[7], o_6[8], n).O_3'\big) \setminus L\end{aligned}$$

We note that O_1 is now negatively charged (it has only one bond), but we do not need to consider it to get our desired result. The next step is to form a bond

between O_3 and either H_3 or H_4. We bond with H_3 with key 10 and break the bond 5, producing a pair of concerted actions. We then promote a weak bond 9 on n in O_2 using rule move from Definition 2 to a strong bond on o_3 which has become available. Also, we promote a weak bond 10 in H_3 to a strong bond on h_3, and, by the structural congruence rule in Figure 5, we derive the transition

$$\begin{aligned}\xrightarrow{\{np[10],\underline{h_3o_3}[5]\}} \big((c_1[1],c_2[2],c_3[3],c_4[9];p).C' \mid (h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid \\ (o_1[3],o_2,n).O_1' \mid (h_3[10];p).H_3' \mid (h_4[6];p).H_4' \mid (o_3[9],o_4[6],n).O_2' \\ \mid (h_5[7];p).H_5' \mid (h_6[8];p).H_6' \mid (o_5[7],o_6[8],n[10]).O_3'\big) \setminus L.\end{aligned}$$

The next step is a proton transfer from O_3 to O_1. We transfer H_5, but we could have used H_6 or H_3 since they all have the p action ready. Performing the transfer of H_5 from O_3 to O_1 (and breaking the bond 7), we obtain

$$\begin{aligned}\xrightarrow{\{np[11],\underline{h_5o_5}[7]\}} \big((c_1[1],c_2[2],c_3[3],c_4[9];p).C' \mid (h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid \\ (o_1[3],o_2,n[11]).O_1' \mid (h_3[10];p).H_3' \mid (h_4[6];p).H_4' \mid (o_3[9],o_4[6],n).O_2' \\ \mid (h_5;p[11]).H_5' \mid (h_6[8];p).H_6' \mid (o_5,o_6[8],n[10]).O_3'\big) \setminus L\end{aligned}$$

and promoting the bond 10 in O_3 by the rule move and the bond 11 in H_5 by rule prom we obtain the final products of the reaction:

$$\begin{aligned}\big((c_1[1],c_2[2],c_3[3],c_4[9];p).C' \mid (h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid (o_1[3],o_2[11],n).O_1' \\ \mid (h_3[10];p).H_3' \mid (h_4[6];p).H_4' \mid (o_3[9],o_4[6],n).O_2' \\ \mid (h_5[11];p).H_5' \mid (h_6[8];p).H_6' \mid (o_5[10],o_6[8],n).O_3'\big) \setminus L\end{aligned}$$

We have methanediol $CH_2(OH)_2$ and a molecule of water (oxygen O_3 plus hydrogens H_6 and H_3). Note that the n, p actions are ready again and all the existing bonds are on strong actions. So we can now reverse the reaction by getting O_3 to abstract a hydrogen from H_4 or H_5.

Finally, let us inspect the bonds with keys 4, 5 and 7 which are broken during this sequence of reactions. These bonds were formed prior to the reaction starting. They are broken as a result of application of our new general prefixing operator. This operator, in conjunction with the driving forces of the partial charges, guides the reaction without relying on any sort of global memory or global control. This is one the main advantages of our approach.

4.2 Other paths through the reaction

There are two other less common ways in which the hydration of formaldehyde in water can happen. They require an additional molecule of water. The three paths through the reaction are shown in Figure 8, now with three waters. The path in Figure 7 is from FA | W | W | W via i2 | W | W and i3 | H_3O | W where FA stands for formaldehyde, W is water, i2 and i3 are the intermediates 2 and 3 in Figure 7 and MD is methanediol. The other two paths start with an interaction

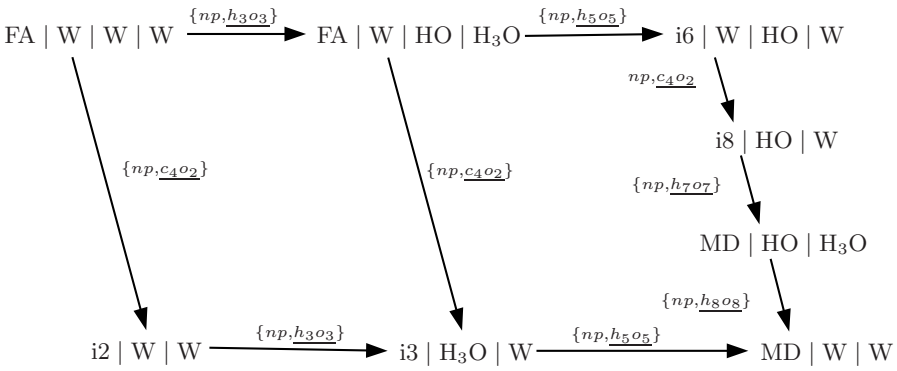

Fig. 8. Three paths through hydration of formaldehyde. Communication keys in concerted transitions are omitted for clarity. The intermediates i6 and i8 are CH_2O^+H and $COH_3O^+H_2$ respectively.

of two water molecules which involves a hydrogen transfer and which leads to FA | W | HO | H_3O. The reaction now branches: either the HO interacts with the formaldehyde, which takes us to i3 | H_3O | W and then we follow the remainder of the main path, or we can go via a more complicated sequence of reactions. The H_3O interacts with the formaldehyde, then a water molecule attaches and finally an interaction with HO brings us to the final state. As we can see all the reactions but one are driven by concerted actions.

We note that in this example the rates of the individual reactions, and the overall rates achieved through the various paths, vary because of the change of energy in the products compared to the reactants. We have decided not to model rates at this stage but rather to concentrate on obtaining all possible valid reactions. We also do not consider spatial arrangement of molecules.

5 Conclusion

We have introduced a reversible process calculus CCB with a novel prefixing operator which is inspired by the mechanism of covalent bonding that allows us to model locally controlled reversibility. We have given the calculus operational semantics. The new operator permits us to perform pairs of concerted actions, where the first element of the pair is a creation of a (weak) bond and the second element is breaking one of the existing bonds. Moreover, our prefixing provides a purely local control of computation; there is no need for an extensive memory or global control. We have shown that the sub-calculus CCB_s satisfies conservation and causal consistency, and the full calculus satisfies several diamond properties. CCB is more expressive than other reversible calculi as it can also model out-of-causal order computation. We have shown that biochemical reactions with covalent bonding can be represented naturally and faithfully thanks to our new prefixing operator and concerted actions transitions.

Acknowledgements The authors acknowledge partial support of COST Action IC1405 on Reversible Computation - extending horizons of computing.

References

1. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
2. L. Cardelli and C. Laneve. Reversible structures. In *9th International Conference on Computational Methods in Systems Biology*, pages 131–140. ACM, 2011.
3. I. Cristescu, J. Krivine, and D. Varacca. A compositional semantics for the reversible pi-calculus. In *Proceedings of LICS 2013*, pages 388–397. IEEE, Computer Society, 2013.
4. V. Danos and J. Krivine. Reversible communicating systems. In *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.
5. V. Danos and J. Krivine. Formal molecular biology done in CCS-R. In *Proceedings of the 1st Workshop on Concurrent Models in Molecular Biology BioConcur 2003*, volume 180 of *ENTCS*, pages 31–49, 2007.
6. V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.
7. S. Kuhn and I. Ulidowski. Towards modelling of local reversibility. In *Proceedings of Reversible Computation 2015*, pages 279–284. Springer, 2015.
8. 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 *Lecture Notes in Computer Science*, pages 297–311. Springer, 2011.
9. I. Lanese, C.A. Mezzina, and J-B. Stefani. Reversing higher-order pi. In *Proceedings of CONCUR 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 478–493. Springer, 2010.
10. I. Lanese, C.A. Mezzina, and J-B. Stefani. Controlled reversibility and compensations. In *Proceedings of Reversible Computation 2012*, volume 7581 of *LNCS*, pages 240–246. Springer, 2012.
11. R. Milner. *A Calculus for Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.
12. I.C.C. Phillips and I. Ulidowski. Reversing algebraic process calculi. In *Proceedings of FOSSACS 2006*, volume 3921 of *LNCS*, pages 246–260. Springer, 2006.
13. I.C.C. Phillips and I. Ulidowski. Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming*, 73:70–96, 2007.
14. I.C.C. Phillips and I. Ulidowski. Reversibility and asymmetric conflict in event structures. In *Proceedings of CONCUR 2013*, volume 8052 of *LNCS*, pages 303–318. Springer, 2013.
15. I.C.C. Phillips, I. Ulidowski, and S. Yuen. Modelling of bonding with processes and events. In *Proceedings of Reversible Computation 2013*, volume 7948 of *LNCS*, pages 141–154. Springer-Verlag, 2013.
16. 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.
17. I. Ulidowski. Equivalences on observable processes. In *Proceedings of LICS 1992*, pages 148–159. IEEE, Computer Science Press, 1992.
18. I. Ulidowski, I.C.C. Phillips, and S. Yuen. Concurrency and reversibility. In *Proceedings of Reversible Computation 2014*, volume 8507 of *LNCS*, pages 1–14. Springer, 2014.