Local reversibility in a Calculus of Covalent Bonding

Stefan Kuhn, Irek Ulidowski

Department of Informatics, University of Leicester, Leicester, LE1 7RH, United Kingdom

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 causes are undone before their effects are undone, which goes beyond what typical reversible calculi can express. However, the core calculus, which uses only the reduced 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 biochemical 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 [38]. 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.

There are networks of reactions in biochemistry, however, 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 [29]. 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 $b$.

We introduced informally a novel and purely local in character mechanism for undoing of computation in short papers \[16, 17\]. 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 \[18\] 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 external control.

The novel features of our calculus are introduced via an example of a simple catalytic reaction. Consider two molecules $A$ and $B$ that are only able to bond if assisted by a catalyst $C$. Once $A$ and $B$ are bonded with the catalyst $C$, $A$ bonds with $B$ and, at the same time, the bond between $A$ and $C$ is broken. Finally, the bond between $B$ and $C$ is broken. This is illustrated below.

\[
\begin{align*}
A & \quad B \\
C & \rightarrow \\
A & \quad B \\
C & \rightarrow \quad A \quad B \\
C & \rightarrow \quad A \quad B \\
C & \rightarrow \\
C & \rightarrow \\
C &
\end{align*}
\]

Figure 1: A catalytic reaction.

We assume $A \overset{df}{=} (a;p).A'$, $B \overset{df}{=} (b;p).B'$ and $C \overset{df}{=} (a,b).C'$, where $A'$, $B'$ and $C'$ represent further potential behaviour of the molecules $A$, $B$ and $C$. 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 $s$ take place, and then $p$ takes place. The molecules $A$, $B$ and $C$ can bond by performing synchronously the matching actions according to the communication function $\gamma(a,a) = c$, $\gamma(b,b) = d$ and $\gamma(p,p) = q$, producing thus new actions $c$, $d$ and $q$ respectively. A weak action $p$ can be left out in $(s;p)$ resulting in the simple prefix $(s).P$ (as in $B$ and $C$ above). In general, the actions of $s$ in $(s;p).P$ can take place in any order, very much like in \[8, 29\], and the new feature is that $p$ can happen only 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 realistic representation of the mechanism of covalent bonding, the most common type of chemical bonding.
between atoms, hence we call our calculus a *Calculus of Covalent Bonding*.

Returning to our example, we represent the system of molecules $A$, $B$ and $C$ as\((\langle a; p \rangle.A' | (b; p).B' | (a, b).C') \setminus \{a, b, p\}\), where ‘$\setminus$’ is the parallel composition and ‘$\setminus$’ the restriction as in ACP [1]. We note that $A$ and $B$ cannot interact initially since $\gamma(a, b)$ is not defined. They can however both interact with $C$:

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

Numbers 1 and 2 are the *communication keys* [25, 26]; they indicate which pairs of actions have bonded. Molecules $A$ and $B$ can now bond on $p$ (with the key 3), producing the action $q[3]$. This causes immediately the breaking of the bond $c[1]$, which means undoing of the action $a$ in $A$ and action $a$ in $C$ (and still leaving $A$ and $B$ bonded). We model such an event of creating a bond and simultaneously breaking another bond by a pair of *concerted actions*:

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

The bond with the key 3 on the weak action $p$ in $A$ is unstable, and thus gets *promoted* to a stable and stronger bond on $a$ and $p$, which is modelled by the following rewrite:

\[
(a; p[3]).A' | (b[2], p[3]).B' | (a, b[2]).C' \Rightarrow (a[3]; p).A' | (b[3], p[3]).B' | (a, b[2]).C'.
\]

Finally, the catalyst dissolves the bond with $B$:

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

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 $c[1]d[2]q[3]c[1]d[2]$. Since creation of bonds $c$ and $d$ causes the bond $q$, we have here an example of an out-of-causal order computation.

The calculus CCB is given *Structural Operational Semantics* (SOS for short) style semantics. This includes novel SOS rules for concerted actions and three rewrite rules that prescribe when bonds on weak actions can be promoted to strong action bonds. We show that CCB is a well behaved calculus by proving a number of useful properties. For example, the sub-calculus with the simple prefixing operator $(s).P$ satisfies causal consistency. We show that the full calculus allows us to represent out-of-causal order computation patterns via the hydration of formaldehyde in water case study.

Next we summarise the main items of related work.
1.1. Related Work

Scientists started to investigate the speed of chemical reactions and the rates achieved as soon as the concept of chemical reactions was first developed. The behaviour of a system of compounds over time can be modelled using a set of ordinary differential equations (ODEs). The fast calculations of such ODEs were popularised by Gillespie [13] in order to show the dynamic behaviour of systems of chemical compounds. When biochemical processes, which involve not only small molecules but also macromolecules, cells and membranes, were modelled attention turned to how the individual objects were represented and how they behaved, and the usefulness of computer science methods was demonstrated in [12].

Process calculi are very successful formalisms for representing concurrent and distributed systems. Each component of a system has its definition which specifies what it does and how it interacts with other components. The behaviour of the system emerges then from the independent actions of the components and from the interactions between them. Calculus of Communicating Systems (CCS) [21], Communicating Sequential Processes (CSP) [14], and the \( \pi \)-calculus [22] are examples of process calculi. Starting with Regev et al. [32, 35, 34] process calculi, specifically the \( \pi \)-calculus, were used to model biochemical systems. The biochemical compounds are represented as processes, and how they react is modelled by communication on ports. A creation of a bond or a dissolution of a bond is represented as establishing or breaking of a communication between ports. So there is a natural analogy between concurrent processes and biochemical entities in natural systems, and between communication among processes and reactions between biochemical entities. The aim of this work, as stated in [35], was to represent suitably biological knowledge in processes and to enable computer-based analysis of this representation. This approach was extended shortly afterwards to include reaction rates [31] using the stochastic \( \pi \)-calculus, which was introduced previously in [30].

Various other calculi, including Bio-PEPA [9], the biochemical abstract machine (BIOCHAM, [10]), P systems [24], BioAmbients [33], the kappa calculus [8] and Brane Calculus [2], followed aiming to capture various other aspects of biological systems such as, for example, compartments and membranes.

Most of biochemical reactions are reversible, and creating bonds is as important in biochemical processes as breaking of bonds. Hence, it became useful to be able to represent directly reversibility in process calculi. This realisation lead to the development of a number of reversible process calculi [1, 8, 23, 20, 19, 18, 20, 6], which have application far beyond biochemistry. Out-of-causal order reversibility, an important aspect of biochemical systems which is typically not captured in traditional reversible process calculi, was first proposed in [23], where the calculus CCSK [20] is extended with an \textit{execution control mechanism} for managing the pattern and the direction of computation. The control mechanism for reversibility is external to the processes it controls, and it can have a global scope. The calculus introduced in this paper has in
contrast no global control and the behaviour of a biochemical systems emerges from the behaviour of its components. Out-of-causal order computation was also studied in \[28, 27\].

2. A Calculus of Covalent Bonding

In this section we define the calculus introduced informally in the Introduction. First, we introduce some preliminary notions and notations.

Let \( A \) be the set of (forward) action labels, ranged over by \( a, b, c, d, e, f \). We partition \( A \) into the set of strong actions, written as \( \mathcal{S}A \), and the set of weak actions, written as \( \mathcal{W}A \). Reverse (or past) action labels are members of \( \mathcal{A} \), with typical members \( a, b, c, d, e, f \), and represent undoing of actions. The set \( \mathcal{P}(A \cup \mathcal{A}) \) is ranged over by \( L \).

Let \( K \) be an infinite set of communication keys (or keys for short) \([25, 26]\), ranged over by \( k, l, m, n \). The Cartesian product \( A \times K \), denoted by \( \mathcal{AK} \), represents past actions, which are written as \( a[k] \) for \( a \in A \) and \( k \in K \). Correspondingly, we have the set \( \mathcal{AK}^* \) that represents undoing of past actions. We use \( \alpha, \beta \) to identify actions which are either from \( A \) or \( \mathcal{AK} \). It would be useful to consider sequences of actions or past actions, namely the elements of \( (A \cup \mathcal{A})^* \), which are ranged over by \( s, s' \) and sequences of purely past actions, namely the elements of \( \mathcal{AK}^* \), which are ranged over by \( t, t' \). The empty sequence is denoted by \( \epsilon \). We use the notation \( \alpha, s \) and \( s, s' \) to denote a concatenation of elements, which can be strings or single actions.

We shall also use two sets of auxiliary action labels, namely the set \( (A) = \{(a) \mid a \in A\} \), and its product with the set of keys, namely \( (A)K \). These labels will be used in the auxiliary rules when defining the semantics of CCB.

We now define the Calculus of Covalent Bonding, or CCB for short. The syntax of CCB is given below where \( P \) is a process term:

\[
P ::= S \mid (s; b).P \mid P \parallel Q \mid P \setminus L
\]

The set of process identifiers (constants) \( \mathcal{PI} \) contains typical elements \( S \) and \( T \). A process identifier \( S \) has normally a defining equation \( S \equiv P \) where \( P \) contains only forward actions (and no past actions). There is also a special identifier \( 0 \), denoting the deadlocked process, which has no defining equation.

We have a general prefixing operator \( (s; b).P \), where \( s \) is a non-empty sequence of actions or past actions. This operator extends the prefixing operator in \([29]\). The action \( b \) is a weak action and it can be omitted, in which case the prefixing is written as \( (s).P \) and is called the simple prefix. The simple prefix is the prefixing operator in \([29]\). One of the actions in \( s \) in \( (s).P \) may be a weak action from \( \mathcal{WA} \). If \( s \) is a sequence that contains a single action, then the action is a strong action and the operator is the prefixing operator of CCS \([21]\). We omit trailing \( 0s \) so, for example, \( (s).0 \) is written as \( (s) \). The new feature of the operator \( (s; b).P \) is the execution of the weak action \( b \), which can happen only after all the actions in \( s \) have taken place. Performing \( b \) then forces undoing one of the past actions in \( s \) (by the concert rule in Figure [4]).
$P \mid Q$ represents two systems $P$ and $Q$ which can perform actions or reverse actions on their own, or which can interact with each other according to a communication function $\gamma$. As in the calculus ACP [11], the communication function is a partial function $\gamma : A \times A \rightarrow A$ which is commutative and associative. The function $\gamma$ is used in the operational semantics to define when two processes can interact. Processes $P$ and $Q$ in $P \mid Q$ can also perform a pair of concerted actions, which is the new feature of our calculus. We also have the ACP-like restriction operator \( \setminus \), where $L$ is a set of labels. It prevents actions from taking place and, due to the synchronisation algebra used, it also blocks communication. If $\gamma(a, b) = c$ then $a.P$ and $b.Q$ cannot communicate in $(a.P \mid b.Q) \setminus c$. Note that we do not use here the usual relabelling operator $[f]$, where $f : A \rightarrow A$, which could be easily added.

The example in the Introduction and our main example in Section 5 seem to indicate that only simple processes of the form $(s; b).0$ are sufficient in the modelling of chemical reactions. However, there are examples where a nested prefix $(s; b)(s'; b').P$ is useful. Consider a base excision repair as in [15] where a protein “walks” along a strand of DNA and repairs faults which occurred in the DNA replication. The walking along a DNA strand could be modelled by actions in $s$, and, once a fault is found, the repair mechanism could be modelled by the actions in $s'$. Another example where the full calculus is useful is a model of long standing transactions with compensations in [20].

The set of process terms is ranged over by $P, Q$ and $R$ and is denoted by $\text{Proc}$. In the setting of CCB these terms are called simply processes. A context $C[ ]$ is a process term containing a hole, represented by $[ ]$. Formally, contexts are defined by the following syntax: $C ::= [ ] \mid (s; b).C \mid P \mid C \mid P \mid C \setminus L$. The term $C[Q]$ denotes the result of filling the hole in the context $C[ ]$ with the process $Q$. We say that $R$ is a subprocess of $P$ if $P$ is $C[R]$ for some context $C[ ]$.

We define the semantics of our calculus by a labelled transition system, LTS for short, which is a structure $(St, AL, \rightarrow : St \times AL \times St)$ with $St$ the set of states, $AL$ the set of action labels and $\rightarrow : St \times AL \times St$ the labelled transition relation. The set of states $St$ is the set $\text{Proc}$. In practice, all our results and examples hold for consistent processes, namely processes reachable from standard processes (see Definition 1). The action labels are the forward actions $AK$, the reverse actions $\overline{AK}$ and the pairs of concerted actions $AK \times \overline{AK}$. The labelled transition relation is defined by SOS rules (Figures 3–6) and rewrite rules (Figure 7), where the rules in Figures 3–4 are influenced by [26]. Note that sequences $s$ and $t$ are members of $(A \cup AK)^*$ and $AK^*$ respectively in Figures 3–5.

We now introduce and explain the SOS rules before returning to the rewrite rules. Let $r$ be an SOS rule for an operator $f$ of CCB as in Figures 3–5. Transitions above the horizontal bar in $r$ are called premises. The set of premises is written as $\text{pre}(r)$. The transition below the bar in $r$ is the conclusion and is written as $\text{con}(r)$. We use two predicates $\text{std}(P) : \text{Proc}$ and $\text{fsh}[m](P) : K \times \text{Proc}$ in our SOS rules. They are defined in Figure 2 and they use two auxiliary functions $k(s) : (A \cup AK)^* \rightarrow \mathcal{P}(K)$ and $\text{keys}(P) : \text{Proc} \rightarrow \mathcal{P}(K)$. The function
k() is defined as follows: k(ε) = ∅, k(α : s) = {l} ∪ k(s) if α = a[l], for a ∈ A and l ∈ K, and k(α : s) = k(s) if α ∈ A. The function keys() is given by keys(0) = ∅, keys(S) = keys(P) if S ≡ P, keys((s; b).P) = k(s) ∪ k(b) ∪ keys(P), keys(P | Q) = keys(P) ∪ keys(Q), and keys(P \ L) = keys(P). Informally keys(P) associates with each term P the set of its keys. A process P is standard, written std(P), if it contains no past actions (hence no keys). A key n is fresh in Q, written fsh[n](Q), if Q contains no past action with the key n. We extend the notion of fresh keys to the sequences of actions and past actions s and t via the function k().

**Example 1.** We illustrate how processes compute forwards using the new prefixing operator. Consider a standard process (a; b).(c) | (a, d, c) and the communication function γ given by γ(a, a) = a and γ(c, c) = c. We have

\[(a; b).(c) \rightarrow (a, d, c) a[l]. \rightarrow (a[1]; b).(c) \rightarrow (a[1], d, c)\]
by the SOS rules \( \text{act1} \) and \( \text{com} \) from Figure 3. This is because \( (c) \) is standard and the key 1 is fresh in \( \varepsilon \). The next step of computation involves a communication of the actions \( c \), which we obtain by rules \( \text{act2} \) and \( \text{com} \):

\[
(a[1]; b).(c) \rightarrow (a[1], d, c[2]) \rightarrow (a[1]; b.(c[2])) \rightarrow (a[1], d, c[2])
\]

We note that the key 2 is fresh in \( a[1] \). Finally, the action \( d \) takes place by \( \text{act1} \) and, informally, the symmetric version of \( \text{par} \).

\[
(a[1]; b).(c[2]) \rightarrow (a[1], d, c[2]) \rightarrow (a[1]; b).c[2] \rightarrow (a[1], d[3], c[2])
\]

Formally, we use \( \text{par} \), the structural congruence rule \( \text{sc} \) in Figure 6 and the reduction rule \( \text{red1} \) in Figure 7.

The next example illustrates how some of the reverse SOS rules work.

**Example 2.** Consider \((a[1], b).(c).S\) where \( S \triangleright (a, b).(c).S \). We have

\[
(a[1], b).(c).S \triangleright (a, b).(c).S
\]

by \( \text{rev act1} \) since \( (c).S \) is standard. Since \((a, b).(c).S\) is the definition of \( S \) we obtain by rule \( \text{rev con} \) \((a[1], b).(c).S \triangleright (a[1], b).S\).

Figure 5 contains the SOS rules that define the new concerted action transitions. The main rule is the rule \( \text{concert} \) that defines when a pair of concerted actions takes place. We also have two auxiliary rules \( \text{aux1} \) and \( \text{aux2} \) which define only an auxiliary transition relation needed in the \( \text{concert} \) rule. Note that the \( \text{concert} \) rule uses \( \text{lookahead} \) \( [3] \). Also note that transitions in \( \text{aux1} \) and \( \text{aux2} \) use the auxiliary labels \((b)[k]\) for all \( b \in W.A \) and \( k \in K \). The rule \( \text{concert par} \) requires that \( k \) is fresh in \( Q \), correspondingly as in \( \text{par} \). Moreover, we need to ensure that when we reverse \( h \) with the key \( l \) in \( P \) we do not leave out any

---

**Figure 4:** Reverse SOS rules. The condition (*) is \( \gamma(a, d) = c \), and and \( b \in WA \).
actions with the key \( l \) in \( Q \) which make up a multiaction communication with the key \( l \). Hence, we also include the premise \( \text{fsh}[l](Q) \) in concert par. The rule concert act requires, correspondingly as act, that \( k \) is fresh in \( t \). Our operational semantics guarantees that if a standard process evolves to \( (t;b),P \), for some \( P \), and \( P \) reverses an action with the key \( l \), then \( l \) is fresh in \( t \). Hence, we do not include \( \text{fsh}[l](t) \) in the premises of concert act. Next, we illustrate how concerted actions transitions work.

**Example 3.** Consider the process \((a;b)\mid a\mid b\mid c\mid a\mid c\mid b\) 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 5

\[
(a[1];b)\mid a[1]\mid b\xrightarrow{a[1]}(a[2];a[1];b[2])\mid a\mid b[2]
\]

since \((a[1];b)\xrightarrow{b[2]}(a[1];b[2])\) by aux1, \((a[1];b[2])\xrightarrow{a[1]}(a[1];b[2])\) by rev act1, and since \( a[1]\mid b \xrightarrow{b[2]} a[1]\mid b[2] \xrightarrow{a[2]} a\mid b[2] \) by par and rev par.
of actions. First we define the function \( fn \) with rewrite rules in Figure 7. The reduction relation is needed to define promotion which potentially combine reductions (defined below) with transitions.

**Example 5.** Consider \((a[1]; b) | (a[1]; b) | e\) with \( \gamma(a, a) = c \) and \( \gamma(b, b) = d \). We clearly have the following pair of concerted actions

\[(a[1]; b) | (a[1]; b) | e \xrightarrow{\{d[2]; a[1]\}} (a; b[2]) | (a; b[2]) | e.\]

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

**Example 4.** Consider \((a[1]; b) | (a[1]; b) | e[2]; b) | (a[1], e[2])\) with \( \gamma(a, a) = c \) and \( \gamma(b, b) = d \). The process cannot perform any concerted actions: Although \((a[1]; b) | (a[1]; b) | e[2]; b) | (a[1], e[2])\) cannot perform the auxiliary \((b[l])\) 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 \( P \) and \( Q \) in the concert rule, respectively, and we notice that we cannot undo a communication on \( a \) or \( e \).

Overall, the transitions in Figures 3, 4 are labelled with \( a[k] \in AK, \) or with \( l \in AK, \) or with concerted actions \( \{a[k], l\} \).

We also have the usual structural congruence rules \( sc \) and \( rev \) in Figure 6 which potentially combine reductions (defined below) with transitions.

Next, we introduce our reduction relation which is given by the reduction (rewrite) rules in Figure 7. The reduction relation is needed to define promotion of actions. First we define the function \( fn \) for free names of processes.

**Definition 1.** The function \( fn : \text{Proc} \rightarrow \mathcal{P}(\mathcal{K}) \) is given as follows: \( fn(0) = \emptyset \), \( fn(S) = fn(P) \) if \( S \xrightarrow{def} P \), \( fn((a : s; b).P) = \{a\} \cup fn((s; b).P) \), \( fn((a; b).P) = \{a, b\} \cup fn(P) \), \( fn(P \mid Q) = fn(P) \cup fn(Q) \), and \( fn(P \setminus L) = fn(P) \setminus L \).
Our reduction rules have names such as, for example, red and we write red:
P \Rightarrow Q to indicate that the reduction rule P \Rightarrow Q is called red. The process P
in the rule P \Rightarrow Q is called a redex, and the process Q is called a contractum. A
reduction rule P \Rightarrow Q can be seen as a prescription for deriving rewrites C[P] \Rightarrow
C[Q] for arbitrary context C[\_]. A P redex may be replaced by its contractum
Q in an arbitrary context C[\_] giving rise to a reduction step: C[P] \Rightarrow C[Q].

**Definition 2.** The reduction relation \( \Rightarrow \) is the smallest reflexive and transitive
relation on CCB processes that is preserved by all contexts, and that satisfies
the rules in Figure 7.

Note that we do not want \( \Rightarrow \) to be symmetric as we wish to apply prom only
from left to right.

The rewrite rules in Figure 7 include prom, move-r, and move-l which promote
weak bonds (here b) to strong bonds (here a). The rule prom applies to the full
version of our prefix operator (with the ; construct), and move-r and move-l apply only to the simple prefix. These three rules are here to model what
happens in chemical systems: a bond on a weak action is temporary and as soon
as there is a strong action that can accommodate that bond (as the result of
concerted actions) the bond establishes itself on the strong action thus releasing
the weak action. In order to align the use of these three rules to what happens
in chemical reactions, we insist that they are used as soon as they becomes
applicable: this is made precise in Definition 2. We could have used the idea
of ordering on SOS rules and rewrite rules [37, 23] to specify that the rewrite
rules prom, move-r and move-r are higher in the ordering than all SOS rules and
the remaining rewrite rules, implying that they should be applied first when
deriving transitions. Alternatively, we could have tried to employ some of the
techniques presented in [5] to define our transition relation. This would require
the use of negative information in the premisses, and the definitions in the style
as those in [37, 23]. However, since we combine SOS rules with rewrite rules,
we opted for a directly defined transition relation.

We now define the transition relation for the labelled transition system for
CCB. Recall that the states of the LTS are processes in Proc and the labels are
members of \( A \), \( AK \), \((A)K \) and the concerted actions labels in \( AK \times AK \). Let
d : Proc \rightarrow \mathbb{N} be the operator depth function defined by 
\( d(P) = 0 \) if \( P \) is a constant, and 
\( d(f(P_1, \ldots, P_n)) = 1 + \max \{ d(P_i) \mid 1 \leq i \leq n \} \) otherwise, where \( f \)
is an operator of CCB. The transition relation is given as follows:

**Definition 3.** We associate to Proc and \( AK \cup AK \cup (A)K \cup (AK \times AK) \) a
transition relation \( \rightarrow \) given by \( \bigcup_{k<\omega} \rightarrow^l \), where transition relations \( \rightarrow^l \subseteq Proc \times
AK \cup AK \cup (A)K \cup (AK \times AK) \times Proc \) are as follows, with \( b \in AK \) and \( \mu \in
AK \cup AK \cup (AK \times AK) \):

1. \( P \xrightarrow{(b)[l]} P' \) if \( d(P) = l \), \( P \xrightarrow{(b)[k]} P' = \rho(con(r)) \), where \( r \) is either
aux1 or aux2, and each premise in pre(r) is a valid transition in \( \bigcup_{k<l} \rightarrow^k \)
or a valid predicate.

11
2. \( P \xrightarrow{a[k]} P' \epsilon \rightarrow \) if \( d(P) = l \), \( P \Rightarrow Q \), for some \( Q \) such that \( Q \) does not contain any \textit{prom}, \textit{move-r} and \textit{move-l} redex, \( Q \xrightarrow{r} Q' = \text{con}(r) \), for some rule \( r \) where each member of \( \text{pre}(r) \) is either a valid transition in \( \bigcup_{k<l} \rightarrow 
olimits^k \), a valid rewrite or a valid predicate, and \( Q' \Rightarrow P' \).

The first part of the definition specifies the auxiliary transitions using rules \textit{aux1} and \textit{aux2}. The second part tells us how to use the remaining rules to define transitions. If \( P \) has no \textit{prom}, \textit{move-r} and \textit{move-l} redex, then we apply our rules in a standard way. Otherwise, we are required to reduce \( P \) to \( Q \) with \textit{prom}, \textit{move-r} and \textit{move-l} first, then we define a transition of \( Q \) to \( Q' \) in a standard way, and finally we reduce \( Q' \) to \( P' \) (if needed). This implies that if \( P \) has a \textit{prom}, \textit{move-r} or \textit{move-l} redex, then we must use one of the structural congruence rules in Figure 6. And, if we use any of these rules, then the reduced process \( Q \) must no longer have any \textit{prom}, \textit{move-r} and \textit{move-l} redex.

The next example illustrates the application of the promotion rewrite rule.

\textbf{Example 6.} The transition \((a[1]; b) | a[1] | b \xrightarrow{\{d[2], a[1]\}} (a; b[2]) | a | b[2] \) from Example 3 cannot be followed by a communication of actions \( a \) because there is a \textit{prom} redex \((a; b[2]) \) in \((a; b[2]) | a | b[2] \). The rewrite of this redex takes priority: the bond 2 moves from the weak \( b \) to the strong \( a \) by \textit{prom}:

\((a; b[2]) | a | b[2] \Rightarrow (a[2]; b) | a | b[2] \)

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.

We shall call henceforth the transitions derived by the forward SOS rules as the \textit{forward transitions} and, the the transitions derived by the reverse SOS rules as the \textit{reverse transitions}. Correspondingly, there are the \textit{concerted (action)} transitions.

\section{Properties of CCB}

In this section we establish several 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 transitions invert the corresponding forward transitions, and vice versa.

\textbf{Definition 4.} A process \( P \) is \textit{consistent} if \( Q \rightarrow^* P \) for some process \( Q \) such that \( \text{std}(Q) \).

\textbf{Proposition 1.} Let \( P \) be consistent. Then

1. If \( P \xrightarrow{a[k]} Q \) then \( k \notin \text{keys}(P) \) and \( \text{keys}(Q) = \text{keys}(P) \cup \{k\} \) for all \( Q \).
2. If \( P \xrightarrow{a[k]} Q \) then \( k \in \text{keys}(P) \) and \( \text{keys}(Q) = \text{keys}(P) \setminus \{k\} \) for all \( Q \).
3. If \( P \xrightarrow{a[k]} P' \) then \( P' \xrightarrow{a[k]} P \). If \( P' \xrightarrow{a[k]} P \) and \( P' \) has no \textit{move-r} or \textit{move-l} redexes, then \( P \xrightarrow{a[k]} P' \).
PROOF. By induction on the depth of the inference tree of transitions $P \overset{a[k]}{\rightarrow} Q$ or $P \overset{a[k]}{\rightarrow} Q$.

Next, we introduce some notation. We define a new transition relation $\rightarrow$ by $P \overset{a[k]}{\rightarrow} Q$ if $P \overset{a[k]}{\rightarrow} Q$ or $P \overset{a[k]}{\rightarrow} Q$. Process $P$ is called the source and $Q$ the target of $P \overset{a[k]}{\rightarrow} Q$. We will use $t, t', t_1, \ldots$ to denote transitions, for example $t : P \overset{a[k]}{\rightarrow} Q$. Two $\rightarrow$ 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 5.** Two coinitial transitions $P \overset{a[k]}{\rightarrow} P'$ and $P \overset{b[l]}{\rightarrow} P''$ are concurrent if there exists $M \neq P$ such that $P' \overset{b[l]}{\rightarrow} M$ and $P'' \overset{a[k]}{\rightarrow} M$.

Note that two concurrent transitions are coinitial and, together with the two transitions (with the target $M$) required by Definition 5, they form a “diamond” structure with the nodes $P, P', P'', M$.

When transitions in Definition 5 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 $(a) | (b) | (b)$ with $\gamma(a, b) = c$. The two coinitial transitions below are in conflict:

$$(a) | (b) | (b) \overset{c[1]}{\rightarrow} (a[1]) | (b[1]) | (b)$$

$$\overset{c[2]}{\rightarrow} (a[2]) | (b) | (b[2])$$

However, coinitial reverse transitions are concurrent. We shall denote the syntactical equality of process expressions by $\equiv$.

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

**Proof.** We prove Proposition 2 by induction on the depth of the inference tree of transition of $P$.

1. Base case: Processes with an inference tree of depth 0 have no reverse transitions, so the proposition is valid.

2. Inductive hypothesis: We assume that for all subprocesses $R$ of $P$ and all $d[m], d[n]$, if $R$ is a consistent process, $R \overset{d[m]}{\rightarrow} R'$ and $R \overset{d[n]}{\rightarrow} R''$, with $m \neq n$, then there is an $N$ so that $R' \overset{d[n]}{\rightarrow} N$ and $R'' \overset{d[m]}{\rightarrow} N$.

3. Induction step: We consider cases depending on the structure of $P$:

   a. $P \equiv (s; b).R$ with $s$ containing two or more past actions: This includes the the case $P \equiv (t; b).R$. This is by rule rev act1. With $s'$ being the sequence obtained from $s$ by removing $a[k]$ and $b[l]$ with $k, l \notin \text{keys}(s')$ we have $(a[k], b[l], s'; c).R \overset{a[k]}{\rightarrow} (a, b[l], ts; c).R \overset{b[l]}{\rightarrow} (a, b, s'; c).R \overset{a[k]}{\rightarrow} (a[k], b, s'; c).R \overset{a[k]}{\rightarrow} (a, b, s'; c).R$. Let $M \equiv (a, b, s'; c).U \mid T$ as required.
(b) \( P \equiv (s; b). R \) with \( s \) containing one or none past action: We cannot deduce the required transitions \( P \xrightarrow{a[k]} P' \) and \( P \xrightarrow{b[l]} P'' \) for any \( a, b, k, l \) and \( l \neq k \) by any SOS rule. Hence the proposition is vacuously valid.

(c) \( P \equiv Q \mid R \): There are three cases:

i. \( P \xrightarrow{a[k]} P' \) by rule \( \text{rev par} \) and \( P \xrightarrow{b[l]} P'' \) by rule \( \text{rev par} \). There are two subcases here:

A. Transitions in the same subprocess: Assume without loss of generality \( Q \xrightarrow{a[k]} Q' \) and \( Q \xrightarrow{b[l]} Q'' \). By the inductive hypothesis there is an \( N \) so that \( Q' \xrightarrow{\gamma} N \) and \( Q'' \xrightarrow{\gamma} N \). We can conclude by using rule \( \text{rev par} \) that \( Q' \mid R \xrightarrow{\gamma} N \mid R \) and \( Q'' \mid R \xrightarrow{a[k]} N \mid R \). With \( M \equiv N \mid R \) we get the result.

B. Transitions in different subprocesses: Assume without loss of generality that \( Q \xrightarrow{a[k]} Q' \) and \( R \xrightarrow{b[l]} R' \). By rule \( \text{rev par} \) \( Q \mid R \xrightarrow{a[k]} Q' \mid R \xrightarrow{b[l]} Q' \mid R' \) and \( Q \mid R \xrightarrow{b[l]} Q' \mid R' \xrightarrow{a[k]} Q' \mid R' \) are valid. These form the required reversal diamond with \( M \equiv Q' \mid R' \).

ii. \( P \xrightarrow{a[k]} P' \) by rule \( \text{rev com} \) and \( P \xrightarrow{b[l]} P'' \) by rule \( \text{rev par} \): Without loss of generality this covers all cases with one \( \text{rev par} \) and one \( \text{rev com} \) transition. We assume that \( a[k] \) is by rule \( \text{rev com} \), that \( \gamma(a_1, a_2) = a \) and that \( b[l] \) is by \( \text{rev par} \). We also assume that \( b \) happens in \( Q \), so that \( Q \xrightarrow{b[l]} Q' \) and \( \text{fsh}[l](R) \), and that \( Q \xrightarrow{a[k]} Q'' \) and \( R \xrightarrow{a[k]} R' \). We know that \( l \neq k \) because \( \text{fsh}[l](R) \) and \( R \xrightarrow{a[k]} R'' \), a transition which could not happen if \( l = k \), since according to Proposition 1.2 a key cannot be fresh for a reverse transition to happen with this key. By the inductive hypothesis there is an \( N \) so that \( Q' \xrightarrow{a[k]} N \) and \( Q'' \xrightarrow{b[l]} N \). Using the \( \text{rev com} \) rule we can deduce \( P \xrightarrow{a[k]} Q' \mid R' \), \( P \xrightarrow{b[l]} Q' \mid R \), \( Q'' \mid R' \xrightarrow{b[l]} N \mid R' \) and \( Q' \mid R \xrightarrow{a[k]} N \mid R' \). Taking \( M \equiv N \mid R' \) we get the result.

iii. \( P \xrightarrow{a[k]} P' \) by \( \text{rev com} \) and \( P \xrightarrow{b[l]} P'' \) by \( \text{rev com} \): Without loss of generality this covers all cases with two \( \text{rev com} \) transitions. We assume \( \gamma(a_1, a_2) = a \) and \( \gamma(b_1, b_2) = b \). Also \( Q \xrightarrow{a[k]} Q' \), \( Q \xrightarrow{b[l]} Q'' \), \( R \xrightarrow{a[k]} R' \) and \( R \xrightarrow{b[l]} R'' \). Since \( Q \xrightarrow{a[k]} Q' \) and \( Q \xrightarrow{b[l]} Q'' \) by the inductive hypothesis it follows that there is an \( N \) so that \( Q' \xrightarrow{b[l]} N \) and \( Q'' \xrightarrow{a[k]} N \) and since \( R \xrightarrow{a[k]} R' \) and \( R \xrightarrow{b[l]} R'' \) there is an \( N' \) so that \( R' \xrightarrow{b[l]} N' \) and \( R'' \xrightarrow{a[k]} N' \).
By rule $\text{rev par}$ it follows that $P \stackrel{a[k]}{\longrightarrow} Q' | R' \stackrel{b[l]}{\longrightarrow} N | N'$ and $P \stackrel{b[l]}{\longrightarrow} Q'' | R'' \stackrel{a[k]}{\longrightarrow} N | N'$. Let $M \equiv N | N'$ as required.

(d) cases $P \equiv R \setminus L$ and $P \equiv S$ with $S \overset{\text{def}}{=} R$ follow in a standard way by using rules $\text{rev res}$ and $\text{rev con}$ in Figure 4, respectively, and the inductive hypothesis.

Before we show that coinitial forward transitions are concurrent if they result in cofinal computations, we introduce traces. A trace is a sequence of composable forward and reverse transitions over CCB. 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 and traces is denoted by ‘‘.’’. The source of a trace is the source of the first transition of the trace, and the target of a trace is the target of the last transition in the trace. As with transitions, two traces are coinitial if they have the same source, and they are cofinal if their targets are identical. The syntactical equality of transitions is also denoted by $\equiv$.

**Proposition 3 (Forward Diamond).** If $P$ is a consistent process and $t_1 \equiv P \stackrel{a[k]}{\longrightarrow} P', t_2 \equiv P \stackrel{b[l]}{\longrightarrow} P''$, with $l \neq k$, and $P' \rightarrow^* R$ and $P'' \rightarrow^* R$, for some $R$, then there is $M \neq P$ such that $P' \stackrel{b[l]}{\longrightarrow} M$, $P'' \stackrel{a[k]}{\longrightarrow} M$ and $M \rightarrow^* R$.

**Proof.** By induction on the depth of the inference tree for transition of $P$.

1. Base case: obvious.
2. Inductive hypothesis: We assume that Proposition 3 holds for all subprocesses $R$ of $P$ and all $c[m], d[n]$, namely if $R$ is a consistent process, $t'_1 \equiv R \stackrel{c[m]}{\longrightarrow} R'$ and $t'_2 \equiv R \stackrel{d[n]}{\longrightarrow} R''$ with $m \neq n$, and $t'_1; \sigma'_1$ and $t'_2; \sigma'_2$, for some $\sigma'_1$ and $\sigma'_2$, are cofinal then there is an $N$ so that $R' \stackrel{d[n]}{\longrightarrow} N$, $N \stackrel{c[m]}{\longrightarrow} N$ and $N \rightarrow^* N'$ is cofinal with $\sigma'_1$ and $\sigma'_2$.
3. Induction step: We assume $t_1 \equiv P \stackrel{a[k]}{\longrightarrow} P'$ and $t_2 \equiv P \stackrel{b[l]}{\longrightarrow} P''$, and consider cases depending on the structure of $P$:
   a. $P \equiv (s; b).R$ with $s$ containing two or more fresh actions: This happens by rule $\text{act1}$. $s$ is of the structure $a, b, s'$ with $k, l \not\in \text{keys}(s')$ so that $P \equiv (a, b, s'; b).R$. So we have $t_3 \equiv P' \stackrel{b[l]}{\longrightarrow} (a[k], b[l], s'; c).R$ and $t_4 \equiv P'' \stackrel{a[k]}{\longrightarrow} (a[k], b[l], s'; c).R$. With $M \equiv (a[k], b[l], s'; c).R$ this gives us the result.
   b. $P \equiv Q | R$:
   i. $P \stackrel{a[k]}{\longrightarrow} P'$ by rule $\text{par}$ and $P \stackrel{b[l]}{\longrightarrow} P'$ by rule $\text{par}$.
      A. Transitions in the same subprocess: Assume without loss of generality $Q \stackrel{a[k]}{\longrightarrow} Q'$ and $Q \stackrel{b[l]}{\longrightarrow} Q''$. By the inductive hypothesis there is an $N$ so that $Q' \stackrel{b[l]}{\longrightarrow} N$ and $Q'' \stackrel{a[k]}{\longrightarrow} N$ and there is a $T$ so that $Q' \rightarrow^* T$ and $Q'' \rightarrow^* T$ implying
3.1. Concerted transitions

The next subsection explores some properties of concerted transitions.

The properties of keys corresponding to those in parts 1 and 2 of Proposition hold also for the concerted transitions in CCB.
Proposition 4. Let $P$ be consistent. If $P \xrightarrow{\{\nu[k]\}\in[l]} Q$ then $k \notin \text{keys}(P)$, $l \in \text{keys}(P)$ and $\text{keys}(Q) = (\text{keys}(P) \cup \{k\}) \setminus \{l\}$ for all $Q$.

Proof. See Appendix.

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

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

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

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

Assume that $R$ is $(c, c[k]).R_1$ for some $R_1$. If we take $S$ as $(a[l]; b).Q \parallel R''$, then the following result could be seen as corresponding to part 3 of Proposition 1.

Proposition 5. 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 $(a[k]; b).Q \parallel R \xrightarrow{d[k]} S$ if and only if $S \xrightarrow{d[k]} (a[k]; b).Q \parallel R$.

4. CCB without weak actions

In this section we discuss the main properties of the sub-calculus of CCB that uses no weak actions. Our prefix operator is thus $(s).P$, where $s$ contains only strong actions. We call this calculus CCB_s. Its SOS rules are as for CCB except that the rules in Figure 5 do not apply as there are no weak actions. The congruence rules prom, move-r and move-l also do not apply since there are no weak actions. We shall use $\mu, \nu$ to denote strong actions in this section.

We shall also consider the forward-only version of CCB_s called CCB_f. The syntax of CCB_f is

$$P ::= S \mid (s).P \mid P \parallel Q \mid P \setminus L$$

and the SOS rules are given in Figure 5. We also have the reduction rules from Figure 7 (without prom, move-r and move-l) which, together with rules in Figure 5, generate the transition relation $\rightarrow_f$ 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 [23, 24] as it uses multiset prefixing (very much like in [8, 23]) and ACP-like communication.

We show firstly that $\rightarrow_f$ for CCB_s is essentially conservative over $\rightarrow_f$. A process of CCB_s is converted to a CCB_f process by “pruning” past actions:
Proof.
1. We use induction on the depth of the inference tree of $P \mu[k] \rightarrow Q$.
   (a) Base case: obvious.
   (b) Inductive hypothesis: We assume that for all subprocesses $R$ of $P$ and all $\nu[l]$, if $R$ is a consistent process and if $R \nu[l] \rightarrow \pi(R')$ for some $R'$, then $\pi(R) \mu[f] \rightarrow \pi(R')$.
   (c) Induction step: We consider cases depending on the structure of $P$.
      Assume that $P$ is consistent and $P \mu[k] \rightarrow P'$ in all cases.
      i. $P \equiv (s; b).R$: There are two cases:
         A. $P \mu[k] \rightarrow P'$ by act1 in Figure 8. Assume $s = \mu : s', t$ where $s' \in A^*$ and $t \in A\pi^*$. Then $(\mu : s'; t; b).R \mu[k] \rightarrow (\mu[k] : s', t; b).R$. We can apply $\pi$ to the processes: $\pi((\mu[k] : s'; t; b).R) = (\mu : s').R = \pi(P)$ and $\pi((\mu[k] : s'; t; b).R) = (s').R = \pi(P')$. The transition $(\mu : s').R \mu[f] (s').R$ is by act1 and we get $\pi(P) \mu[f] \rightarrow \pi(P')$.
         B. $P \mu[k] \rightarrow P'$ by act2 in Figure 8. We deduce that $P \equiv (t; b).R$, $fsh[k](t)$ and $P' \equiv (t; b).R'$. Recall that $t$ contains only past actions. The transition must be $(t; b).R \mu[k] \rightarrow (t; b).R'$, and...
we get by $\text{act}_f R \xrightarrow{\mu[k]} R'$ and $\text{fsh}[k](t)$. Applying $\pi$ to the processes we get $\pi((t; b).R) = \pi(R)$ and $\pi((t; b).R') = \pi(R')$.

Since $R \xrightarrow{\mu[k]} R'$, by the inductive hypothesis, we obtain $\pi(R) \xrightarrow{\mu_f} \pi(R')$. The last implies $\pi((t; b).R) \xrightarrow{\mu_f} \pi((t; b).R')$ which is $\pi(P) \xrightarrow{\mu_f} \pi(P')$.

ii. $P \equiv R \mid Q$: There are two cases:

A. This happens by $\text{par}$ in Figure 3. Since $P \xrightarrow{\mu[k]} P'$, by rule $\text{par}$, $R \xrightarrow{\mu[k]} R'$ and $\text{fsh}[k](Q)$ must hold. By the inductive hypothesis $\pi(R) \xrightarrow{\mu_f} \pi(R')$ is true. By rule $\text{par}$ $\pi((R \mid Q) \xrightarrow{\mu_f} \pi((R' \mid Q)$ holds, which means $\pi(R \mid Q) \xrightarrow{\mu_f} \pi(R' \mid Q)$ and $\pi(P) \xrightarrow{\mu_f} \pi(P')$ as required.

B. This is by $\text{com}$ in Figure 3. The inductive hypothesis in this case is not only true about $R$, but also about $Q$, so that if $Q$ is consistent and $Q \xrightarrow{\mu[k]} Q'$ then $\pi(Q) \xrightarrow{\mu_f} \pi(Q')$.

Assume $P$ is consistent, $P \xrightarrow{\mu[k]} P'$ and $\gamma(\mu_1, \mu_2) = \mu$. We can calculate $\pi(P) = \pi(R \mid Q) = \pi(R) \mid Q$ and $\pi(P') = \pi(R' \mid Q') = \pi(R') \mid Q'$. Since $\pi(R) \xrightarrow{\mu_f} \pi(R')$ and $\pi(Q) \xrightarrow{\mu_f} \pi(Q')$ it follows that according to rule $\text{com}$ $\pi((R \mid Q) \xrightarrow{\mu_f} \pi((R' \mid Q')$ which means $\pi(P) \xrightarrow{\mu_f} \pi(P')$ as required.

iii. Cases for $P \equiv R \setminus L$ and $P \equiv S$ with $S \overset{\text{def}}{=} R$ are straightforward.

2. We prove Theorem 2 by using induction on the definition of $\pi(P)$, which means on the structure of $P$.

(a) Base case: obvious.

(b) Inductive hypothesis: We assume that for all subprocesses $R$ of $P$ and all $\nu$, if $\pi(R) \xrightarrow{\nu} R'$, then for any $l \in K \setminus \text{keys}(R)$ there is $R''$ such that $R \xrightarrow{\mu[l]} R''$ and $\pi(R'') = R'$.

(c) Induction step: We show the result for $\pi(P)$ as well. We consider cases depending on the structure of $P$. Assume that $P$ is consistent and $\pi(P) \xrightarrow{\mu_f} Q$ for some $Q$ in all cases.

i. $P \equiv (s, s'; b)$ and $\pi((s, s'; b).R) = (s').\pi(R)$: Here rule $\text{act}_f$ applies with $s'$ being of the structure $\nu, s''$. The transition is $(\nu, s'').R \xrightarrow{\nu} (s'').R$. Then with $k \in K \setminus \text{keys}(P)$ we have $(\nu, s'').R \xrightarrow{\mu[k]} (\nu[k], s'').R$ and we let $Q' \equiv (\nu[k], s'').R$ with $\pi(Q') \equiv (s'').R$.

ii. $P \equiv (t; b).R$ and $\pi((t; b).R) = \pi(R)$: Here the transition is in $R$ by $\pi(R) \xrightarrow{\nu} R'$. By the inductive hypothesis $R \xrightarrow{\mu[k]} R''$ and $\pi(R'') = R'$ for some $R''$. We deduce that $\pi(P) \xrightarrow{\mu_f} R'$ and $P \xrightarrow{\mu[k]} R''$. Let $Q' \equiv R''$ with the required properties.

iii. $P \equiv R \mid T$ and $\pi(P) = \pi(R \mid T)$: There are two cases:
A. The transition is by $\text{par}_f$. Assume $\pi(R \mid T) \overset{\mu}{\rightarrow}_f Q$ for some $Q$. So $\pi(R \mid T) \overset{\mu}{\rightarrow}_f Q$ implies without loss of generality $\pi(R) \overset{\mu}{\rightarrow}_f R'$ for some $R'$ and $Q \equiv R' \mid \pi(T)$ since $\pi(P) = \pi(R \mid T) = \pi(R) \mid \pi(T)$. Since $\pi(R) \overset{\mu}{\rightarrow}_f R'$ by the inductive hypothesis there exists a $U$ and a $k \in K \setminus (\text{keys}(R) \cup \text{keys}(T))$ such that $R \overset{\nu[k]}{\rightarrow}_f U$ and $\pi(U) = R'$. Since $R \overset{\nu[k]}{\rightarrow}_f U$ and since $\text{fsh}[k](T)$ we obtain by rule $\text{par}_R \mid T \overset{\mu[k]}{\rightarrow}_f U \mid T$. Now we calculate $\pi(U \mid T) = \pi(U) \mid \pi(T) = R' \mid \pi(T) = Q$. Let $Q' \equiv U \mid T$ with the required properties.

B. The transition is by $\text{com}_f$. Since $\pi(P) \overset{\mu}{\rightarrow} P'$, by rule $\text{com}_f$, $R \overset{\nu_1}{\rightarrow} R', T \overset{\nu_2}{\rightarrow} T'$ and $\gamma(\nu_1, \nu_2) = \mu$ for some $R', T'$ must hold without loss of generality. The inductive hypothesis in this case is not only true about $R$, so there is an $R''$ such that $R \overset{\nu_1[k]}{\rightarrow} R''$ and $\pi(R'') = R'$ and a $T''$ such that $T \overset{\nu_2[k]}{\rightarrow} T''$ and $\pi(T'') = T'$. We can deduce that $R \mid T \overset{\nu[k]}{\rightarrow} R'' \mid T''$. Also $\pi(R'' \mid T'') = \pi(R''') \mid \pi(T'') = R' \mid T' = P'$. Let $Q' \equiv R'' \mid T''$. This has the required properties.

iv. Cases for $P \equiv R \setminus L$ and $P \equiv S$ with $S \overset{\text{def}}{=} R$ are straightforward.

We now consider the causal consistency property, first defined and discussed in $[13, 16]$, for CCB. We define when a set of keys is a cause for another key:

**Definition 7.** The set of causes of a key $k$ in $P$ is defined as follows:

- $\text{cau}(0, k) = \emptyset$
- $\text{cau}((s).P, k) = k(s) \cup \text{cau}(P, k)$ if $k \in \text{keys}(P)$
- $\text{cau}((s).P, k) = \emptyset$ if $k \notin \text{keys}(P)$
- $\text{cau}(P \mid Q, k) = \text{cau}(P, k) \cup \text{cau}(Q, k)$
- $\text{cau}(P \setminus L, k) = \text{cau}(P, k)$
- $\text{cau}(P \setminus L, k) = \text{cau}(P, k)$
- $\text{cau}(S) = \text{cau}(P)$ if $S \overset{\text{def}}{=} P$

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 by induction on the depth of the inference tree for transition of $P$: see Appendix.

**Proposition 6.** If $t_1 \equiv P \overset{\nu[k]}{\rightarrow} P'$ and $t_2 \equiv P \overset{\nu[l]}{\rightarrow} P''$, then either $t_1$ and $t_2$ are concurrent or $k \in \text{cau}(P'', t)$.

Next we introduce further notation on *traces*. We denote the reverse transition corresponding to a forward transition $t$ (and the forward transition corresponding to a reverse transition $t$) as $t^\ast$. Similarly to denoting the reverse transitions by $\ast$, we denote the reverse trace of $\sigma$ as $\sigma^\ast$. The empty trace with a process $P$ is written as $\epsilon_P$, and when the process is the source or the target of the transition $t$, we write $\epsilon_{\text{source}(t)}$ respectively $\epsilon_{\text{target}(t)}$. Similarly as with forward and reverse
transitions, we shall use forward traces (traces composed of forward transitions only) and reverse traces (traces composed of reverse transitions only).

We can now define causal equivalence between traces.

**Definition 8.** Causally equivalent traces are defined by the least equivalence relation \( \approx \) which is closed under composition and obeys the following rules, where \( t_1 \) is \( P \xrightarrow{a[k]} Q \), \( t_2 \) is \( P \xrightarrow{b[l]} R \), \( d_1 \) is \( Q \xrightarrow{b[l]} S \) and \( d_2 \) is \( R \xrightarrow{a[k]} S \):

\[
\begin{align*}
t_1; d_1 \approx t_2; d_2, & \quad t_1; t^* \approx \epsilon_{\text{source}(t)}, & \quad t^*; t \approx \epsilon_{\text{target}(t)}
\end{align*}
\]

The first relation 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 relations 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 \([7, 19]\). 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. The proofs are provided in Appendix.

**Proposition 7 (Rearrangement).** If \( \sigma \) is a trace then there exist forward traces \( \sigma_1 \) and \( \sigma_2 \) such that \( \sigma \approx \sigma_1; \sigma_2 \).

**Proposition 8 (Shortening).** If \( \sigma_1 \) and \( \sigma_2 \) are coinitial and cofinal traces, with \( \sigma_2 \) a forward trace, then there exists a forward trace \( \sigma_1' \) of length at most that of \( \sigma_1 \) such that \( \sigma_1' \approx \sigma_2 \).

Next we have the second important result for CCB\(_s\).

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

**Proof.** The statement, if \( \sigma_1 \approx \sigma_2 \) then \( \sigma_1 \) and \( \sigma_2 \) are coinitial and cofinal, follows directly from the construction of \( \approx \) in Definition 8.

Next we show that if \( \sigma_1 \) and \( \sigma_2 \) are coinitial and cofinal then \( \sigma_1 \approx \sigma_2 \). We use induction on the sum of the lengths of \( \sigma_1 \) and \( \sigma_2 \) and on the distance between the end of \( \sigma_1 \) and the earliest pair of transitions \( t_1 \) in \( \sigma_1 \) and \( t_2 \) in \( \sigma_2 \) which are not equal. The coinitial and cofinal traces \( \sigma_1 \) and \( \sigma_2 \) are rewritten as compositions of reverse and forward traces by Proposition 7. There are three cases:

1. \( t_1 \) is forward and \( t_2 \) is reverse: We can assume that \( \sigma_1 = \sigma^*; t_1; \sigma' \) and \( \sigma_2 = \sigma^*; t_2; \sigma'' \) by Proposition 4. Since \( t_1 \) is forward \( t_1; \sigma' \) must be forward, whereas \( t_2; \sigma'' \) must start with at least one reverse transition. Since \( \sigma_1 \) and \( \sigma_2 \) are coinitial and cofinal \( t_1; \sigma' \) and \( t_2; \sigma'' \) are also coinitial and cofinal. We notice that the assumption of Proposition 8 is valid for \( t_1; \sigma' \) and \( t_2; \sigma'' \), hence there is a forward trace \( \sigma''' \) shorter than \( t_2; \sigma'' \) such that \( \sigma''' \approx t_2; \sigma'' \). So the trace \( \sigma^*; \sigma''' \) is shorter than \( \sigma_2 = \sigma^*; t_2; \sigma'' \), and the result follows by induction.
2. $t_1$ and $t_2$ are forward: Assume $t_1 \equiv R \xrightarrow{\mu[k]} R'$ and $t_2 \equiv R \xrightarrow{\nu[l]} R''$ for some $R, R', R''$. Transitions $t_1$ and $t_2$ must be concurrent according to Proposition [3]. Since $\sigma_1$ and $\sigma_2$ are cofinal there must be transitions $t'_1 \equiv P \xrightarrow{\nu[l]} P'$ and $t'_2 \equiv Q \xrightarrow{\mu[k]} Q'$ for some $P, P', Q, Q'$ at some later stage in $\sigma_1$ and $\sigma_2$. We look at the case of $t'_1 \equiv P \xrightarrow{\nu[l]} P'$ in $\sigma_1$ here, the other case follows in the corresponding way. All transitions from $t_1$ to $t'_1$ are concurrent since we can do $\mu[k]$ either as the first transition $R \xrightarrow{\mu[k]} R'$ of $\sigma_1$ or as the last transition in the sub-trace $t_2; t'; t'_2$. Hence we can rearrange the transitions by the technique used in the proof of Proposition [2] so that $R \xrightarrow{\nu[l]} R'''$ comes first. This transition is identical to $t_2$. So we have moved the first non-matching pair of transitions in $\sigma_1$ and $\sigma_2$ to the right, and the result follows by induction.

3. $t'_1$ and $t'_2$ are reverse: Transitions $t'_1 \equiv R \xrightarrow{\mu[k]} R'$ and $t'_2 \equiv R \xrightarrow{\nu[l]} R''$, for some $R, R'$ and $R''$, are undoing different actions. Since $\sigma_1$ and $\sigma_2$ are cofinal there is either a transition $t'_1 \equiv P \xrightarrow{\mu[k]} P''$, for some $P$ and $P''$ in $\sigma_2$, or a transition $t'_1 \equiv P \xrightarrow{\mu[k]} P'$, for some $P$ and $P'$ in $\sigma_1$, redoing the action undone in $t'_1$. For $t'_2$ there is either a transition $t'_2 \equiv S \xrightarrow{\nu[l]} S''$, for some $S'$ and $S''$ in $\sigma_1$, or a transition $t'_2 \equiv S \xrightarrow{\nu[l]} S'$, for some $S$ and $S'$ in $\sigma_2$, redoing the action undone in $t'_2$. We treat these cases separately:

(a) $t'_1 \equiv P \xrightarrow{\mu[k]} P''$ in $\sigma_2$ or $t'_2 \equiv S \xrightarrow{\nu[l]} S''$ in $\sigma_1$: We consider $t'_1 \equiv P \xrightarrow{\mu[k]} P''$ in $\sigma_2$, the other case follows correspondingly. The traces $t'_1$ and $t'_2$ are concurrent according to Proposition [2]. All transitions from $t'_2$ to $t'_1$ must be concurrent since from $R$ we can do $\mu[k]$ either as the first transition as in $\sigma_1$ or as the last one in this sub-trace. So we can rearrange the transitions by a similar technique of “swaps” as used in the proof of Proposition [2] so that $R \xrightarrow{\mu[k]} R'''$ comes first. This transition is identical to $t'_1$, and we have moved the first non-matching pair of transitions in $\sigma_1$ and $\sigma_2$ to the right, hence the case follows by induction.

(b) $t'_1 \equiv P \xrightarrow{\mu[k]} P'$ in $\sigma_1$ and $t'_2 \equiv S \xrightarrow{\nu[l]} S''$ in $\sigma_2$ and $P'' \xrightarrow{\mu[k]} P'''$ not in $\sigma_2$ and $S'' \xrightarrow{\nu[l]} S'''$ not in $\sigma_1$: We consider $t'_1 \equiv P \xrightarrow{\mu[k]} P'$ in $\sigma_1$ only, the case $t'_2 \equiv S \xrightarrow{\nu[l]} S'$ in $\sigma_2$ follows correspondingly. We show that the transitions from $t'_1$ up to $t'_1$ are concurrent. In fact, we show that $t''_1$ and $t''$, the next transition on the way to $t'_1$, are concurrent. If they are concurrent we can swap them thus moving $t'_1$ to the right towards $t'_1$. If we follow this approach we will eventually have $t''_1$ immediately to the left of $t'_1$, and we can remove $t''_1; t'_1$ according to Definition [8]. What remains to be done is to prove that $t''_1$ and $t''$ are concurrent. There are two cases: If $t''$ is forward then we have a situation as in the proof of Proposition [8] where we show that the transitions
are concurrent. The second case is for $t''$ being reverse. Assume $t'' \equiv R' \xrightarrow{\alpha[n]} R''$. Hence by Proposition 1.3 we have $R' \xrightarrow{\mu[k]} R$. So we have $R' \xrightarrow{\alpha[n]} R''$ and $R' \xrightarrow{\mu[k]} R$. This matches the assumptions of Proposition 1.3 (where $R'$ is $P$). If we show that $n \notin \text{cau}(R, k)$ then they are concurrent. The transitions $R_1 \xrightarrow{\mu[k]} R' \xrightarrow{\alpha[n]} R''$ show that $\mu[k]$ is undone before $\alpha[n]$, so $n$ cannot be one of the causes of $k$ in $R$. Hence the transitions $t_1''$ and $t''$ are concurrent, and the result follows by induction.

One of the consequences of causal consistency for CCBs 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 6.

5. The hydration of formaldehyde in water

In this section we describe in detail the hydration of formaldehyde in water. 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 industrial processes and is therefore produced in large quantities. At room temperature, formaldehyde forms a colourless and smelly gas. Formaldehyde reacts with water molecules to form methanediol. The reaction is shown in Figure 9. It is reversible but it is much faster towards the methanediol, so the equilibrium contains mostly methanediol. It should be noted that other reactions are taking place in a solution of formaldehyde in water, however we consider only the interaction of formaldehyde with water molecules. The carbon atom of formaldehyde is not shown in Figure 9 in line with a common convention. It resides at the point where the lines from the oxygen and the hydrogens meet; we follow this convention in all other reaction diagrams. We use single lines to represent single bonds in all our reactions, and we use double lines for double bonds, where two electron pairs are shared between atoms.

![Figure 9: Hydration of formaldehyde in water into methanediol](image)

\[ H\cdot O \quad + \quad H\cdot C\cdot H \quad \xleftrightarrow{\text{reversible}} \quad H\cdot O\cdot H \]

$^1$Detailed description of the reaction is given in [4]. The main path is on page 143, the base-catalysed reaction is on page 713, and the acid-catalysed reaction in on page 343 of [4].
5.1. The most common mechanism of the reaction

A closer look at the reaction in Figure 9 shows that the most common sequence of intermediate reactions leading to methanediol is the one given in Figure 10. We now describe carefully its steps. The oxygen atom in one of the water molecules attacks the central carbon in the formaldehyde to form the intermediate 2 in Figure 10. Then one of the hydrogens of the positively charged oxygen is taken away by another water molecule in a proton transfer. This gives the intermediate 3. Finally, a proton is donated to the other oxygen in the intermediate 3 to give the final configuration 4.

In order to model this reaction we need to understand what it is that makes it happen. The main factor is that the oxygen in the water is nucleophilic: it tends to connect to another atomic nucleus. In the formaldehyde the oxygen attracts electrons towards itself, so the carbon has a positive charge. Now the oxygen in the water is attracted by this carbon and, due to its nucleophilicity, it forms a bond to the carbon. This bond is formed out of the electrons of one of the electrons in the oxygen so far not involved in bonding, so-called lone electron pairs. 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. So we have a continuous change between two tetracoordinate species through a pentacoordinate intermediate. We have now reached the intermediate 2 in Figure 10.

Looking at the intermediate 2, we can see that this has one oxygen which is negatively charged, because it has three lone pairs and a surplus of one electron. The other oxygen is positively charged, since it has three bonds and only one lone pair, therefore is missing an electron. The intermediate 2 reacts then with a water molecule. The water molecule is nucleophilic and has lone pairs for a bonding. It therefore can abstract one of the hydrogens on the positively charged oxygen. This leads to the intermediate 3 and a H$_3$O molecule, a water with an additional hydrogen and a positively charged oxygen.

Finally, a hydrogen can be re-donated to the negatively charged oxygen. Note that the re-donated hydrogen may not be the one which was originally attached to the oxygen. This transfer is possible since the oxygen in the H$_3$O is electron-deficient and the negative oxygen is rich in electrons. We then get the substrate 4 which contains the final product methanediol, and water.
5.2. Other paths through the reaction

There are other paths through the reaction of formaldehyde and water. We assume that we now have a mixture of three water molecules and one formaldehyde molecule. This shall allow us to explore all other water-formaldehyde interactions. Two water molecules can interact to form H₃O and HO. This is known as autoprotolysis of water and is described in detail in [16]. These two molecules can be considered an acid and a base respectively. Both bases and acids can interact with formaldehyde and thus produce methanediol by different mechanisms than those in the previous section.

![Figure 11: Acid-catalysed hydration of formaldehyde in water.](image)

The acid-catalysed reaction is described in Figure 11. Here the H₃O molecule, which was formed during the autoprotolysis serves as the acid as it easily donates a proton. This proton can be donated to the oxygen in the formaldehyde, since this is slightly negatively charged, as we have seen. We then get the intermediate 6 in Figure 11. The charge on the oxygen can “move” to the carbon by the electrons forming the bonds a lone pair on the oxygen. The real charge distribution is somewhere in between. We shall assume that the intermediates 6 and 7 are somewhat different structures, and we shall model the transition from 6 to 8 as a pair of concerted actions. Once the intermediate 8 is reached one of the protons from the oxygen in the intermediate 8 can be abstracted by one of the water molecules, giving H₃O and making it a catalytic process.

![Figure 12: Base-catalysed hydration of formaldehyde in water.](image)

The base-catalysed reaction in Figure 12 starts with a water and a HO⁻ molecule, the base, which tends to accept protons very strongly. It can therefore

---

2 There are slightly different definitions of acids and bases in the literature, we use the general Bronsted-Lowry acid-base theory, which defines a base as a proton acceptor and an acid as a proton donor.
interact with the carbon in the formaldehyde, which is similar to the water attacking the carbon. One of the bonds of the carbon-oxygen double bond is broken and the oxygen becomes negatively charged. This gives the intermediate 11 which is in fact the same as the intermediate 3 in Figure 10. Then one of the protons of the water can be abstracted, and we obtain the methanediol and an HO molecule. Although this is the same structure as the original acid, it is made up from different atoms. The process is considered catalytic since the HO molecule is retrieved at the end of the process.

![Figure 13: Other compounds possible in the reaction of formaldehyde with water.](image)

There are other ways that several molecules of water can react with formaldehyde to produce methanediol and other byproducts. For example the base, HO, could directly interact with intermediate 7 to form methanediol and two water molecules. Also, the two compounds given in Figure 13 can be created as intermediate products in the reaction.

All possible reactions of a system of one formaldehyde and three water molecules are shown in Figure 14. Although the direction of reactions are denoted with arrows, all reactions are reversible; the arrows merely indicate the direction to the final product. We write FA for formaldehyde, W for water and MD for methanediol. Many of the intermediate compounds are denoted by iX where X ∈ {2, 3, 6, 7, 8, 13, 14}. Figure 14 includes all intermediates from Figure 11 and Figure 12. The compounds in Figure 14 have only one path leading to them, and we could only move away from them by reverting these actions. This is because these compounds are the “extreme” states where there is either no hydrogen on an oxygen or both oxygens are fully saturated with two hydrogens. The resonance between the intermediates i6 and i7 in Figure 11 is represented by a dashed arrow.

6. A CCB model of the hydration of formaldehyde in water

We are now ready to model our reaction in CCB. Figure 15 shows in more detail the three main paths through our reaction that we described in the last section; these paths have been highlighted in bold in Figure 14.

The initial state of the reaction is modelled by the composition of one formaldehyde FA three waters, written as FA | W | W | W. The final products are methanediol MD and two waters, written as MD | W | W. The path via the intermediate molecules 2 and 3 in Figure 14 is via the nodes denoted by i2 and
Figure 14: All possible reactions in a system of formaldehyde and three water molecules. The reactions displayed in more detail in Figure 15 are denoted here by bold arrows.

i3 in Figure 15: The FA | W | HO | H₂O node is the result of an autoprotolysis. The base-catalysed and acid-catalysed reactions are put into the same diagram, again the intermediates i6 and i8 correspond to molecules in Figure 11. As we can see in Figure 15, the reactions are driven completely by concerted actions.

We model the formaldehyde molecule CH₂O and the three water molecules H₂O as appropriate compositions of their atoms, namely hydrogen, oxygen and carbon. We use our general prefixing operator to define these atoms:

\[
H \overset{\text{def}}{=} (h; p).H' \\
O \overset{\text{def}}{=} (o, o, n).O' \\
C \overset{\text{def}}{=} (c, c, c, c; p).C'
\]

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 can have up to three bonds. Normally is has two bonds, however, if a suitable reaction partner is close, an additional weak bond is available. We use therefore the simple prefixing operator to model it, with one weak action n standing for the potential for a negative partial charge. A partial charge means that a part or parts of a molecule have an electric charge, even though the molecule as a whole
is neutrally charged. These uneven charge distributions enable many reactions by allowing another molecule to attack a partially charged part. The hydrogen has one strong bond, and we use additionally a weak action \( p \) to represent that it can become positively charged. Processes \( H', O', \) and \( C' \) represent unspecified further behaviour of the respective atoms.

Since our reaction involves multiple copies of \( H \) and \( O \) we shall adopt a subscript notation to denote distinct copies of the same process. Both action labels and process names will be subscripted. For example, \( H_1 \overset{\text{def}}{=} (h_1; p).H'_1 \) and \( H_2 \overset{\text{def}}{=} (h_2; p).H'_2 \) are two copies of hydrogen \( H \). We shall abstract away these subscripts in Section 6.5, where we introduce informally chemical equivalence. Also, the copies of actions \( c \) of carbon will also be subscripted.

The synchronisation function \( \gamma \) is defined on subscripted actions as follows:

\[
\begin{align*}
\gamma(c_i, h_j) &= c_i h_j & i \in \{1, \ldots, 4\}, j \in \{1, \ldots, 8\} \\
\gamma(h_i, o_j) &= h_i o_j & i \in \{1, \ldots, 8\}, j \in \{1, \ldots, 6\} \\
\gamma(c_i, o_j) &= c_i o_j & i \in \{1, \ldots, 4\}, j \in \{1, \ldots, 6\} \\
\gamma(c_i, n) &= c_i n & i \in \{1, \ldots, 4\} \\
\gamma(h_i, n) &= h_i n & i \in \{1, \ldots, 8\} \\
\gamma(n, p) &= np
\end{align*}
\]

Now we are ready to model \( \text{H}_2\text{O} \) and \( \text{CH}_2\text{O} \) molecules. A water molecule is modelled simply as a composition of two copies of the hydrogen process with one copy of oxygen; see below. The restriction of actions \( h_i \) and \( o_i \) for \( i \in \{3, 4\} \), ensures that actions such as \( h_3[5] \) cannot be undone alone but only together with their partners \( o_3[5] \). This happens via undoing of \( h_3 o_3[5] \) bond. Correspondingly, when any of these actions is fresh they can only happen together with their partners (as prescribed by the communication function \( \gamma \)), and not alone. We also restrict \( np \) to stop self-bonding of water’s hydrogens with its oxygen.

\[
((h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2) \setminus \{h_3, h_4, o_3, o_4, np\}
\]
The formaldehyde molecule is modelled by
\[
((c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2 \mid (o_1[3], o_2[4], n).O'_1) \\
\setminus \{c_1, c_2, c_3, c_4, h_1, h_2, o_1, o_2, np, c_1h_1, c_2h_2\} \\
\mid ((h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2) \setminus \{h_3, h_4, o_3, o_4, np\} \\
\mid ((h_5[7]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[7], o_6[8], n).O'_3) \setminus \{h_5, h_6, o_5, o_6, np\} \setminus \{n, p\}
\]

We restrict the appropriate versions of the \(c_i, o_j\) and \(h_k\) actions, for all \(i, j\) and \(k\), meaning that we do not allow the creation of new bonds (involving these actions) between different molecules as single acts of synchronisation. Such new bonds will be created via concerted actions and (almost) always via the weak \(np\) bonds which will then get promoted to strong bonds.

We also restrict \(c_ih_k\) for \(i \in \{1, 2\}\). It prevents breaking any of the bonds between \(C_1\) and its hydrogens \(H_1\) and \(H_2\). 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, so this models the reality closely. Secondly, it also prevents \(O_2\) or \(O_3\) from abstracting \(H_1\) or \(H_2\) from the carbon. Note that \(h_i, o_j\) and \(n\) are not restricted in FA and in W: this allows us to break bonds involving these actions as a part of concerted actions.

The four molecules of the reaction are now composed in parallel:
\[
(CH_2O \mid H_2O \mid H_2O \mid H_2O) \setminus \{n, p\}
\]

We restrict actions \(n\) and \(p\), so that they cannot happen separately from each other but only together within this system of processes.

The main path through the reaction require only two copies of water so we start with the following initial state, where keys \(1, \ldots, 8\) specify the existing initially bonds of formaldehyde and the two waters.
\[
(((c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2 \mid (o_1[3], o_2[4], n).O'_1) \\
\setminus \{c_1, c_2, c_3, c_4, h_1, h_2, o_1, o_2, np, c_1h_1, c_2h_2\} \\
\mid ((h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2) \setminus \{h_3, h_4, o_3, o_4, np\} \\
\mid ((h_5[7]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[7], o_6[8], n).O'_3) \setminus \{h_5, h_6, o_5, o_6, np\} \setminus \{n, p\}
\]

In order to simplify the display of transitions or rewrites of this process we omit the four restrictions and write the initial process simply as
\[
(c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2 \mid (o_1[3], o_2[4], n).O'_1 \\
\mid (h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2 \\
\mid (h_5[7]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[7], o_6[8], n).O'_3
\]
remembering that the restrictions are still there.

6.1. The main path through the reaction

We first consider the reaction steps in Figure 10 following the path via \(i3\) | \(H_3O\) | \(W\) and \(i2\) | \(W\) | \(W\) in Figures 14 and 15. The first step is the \(n, p\) reaction between \(C_1\) and \(O_2\) or \(O_3\). Note that there are other \(n, p\) reactions that
are allowed by our model. We could have $O_2$ getting a hydrogen from $O_3$, or vice versa, which is the autoprotolysis of water which we have mentioned before. We shall discuss this further later on. Also $O_1$ could pull one of the hydrogens from one of the water molecules: again we shall discuss it later.

Returning to the first step of our reaction, we have $O_2$ bonding with $C$ with the key 11. This is followed immediately by breaking of the bond 3 or 4 by the rule concert. Note that breaking of 1 or 2 is not possible because of the restriction of $c_1h_1$ and $c_2h_2$. We break the bond 4 and get a transition by concerted actions: we create the bond $np[11]$ and break the bond $c_3o_2[4]$. Henceforth we shall write the name of the target of a transition below the transition, using the names that appear in Figures 14–15. Here, for example, the compound resulting from this transition is $i2 \mid W$ but since in general we have an extra water $W$, we write $i2 \mid W \mid W$.

\[
\frac{\{np[11], c_3o_2[4]\}}{(c_1[1], c_2[2], c_3[3], c_4[11]; p[11]), C^\prime \mid (h_1[1]; p), H_4^\prime \mid (h_2[2]; p), H_5^\prime \mid (o_3[3], o_2, n).O_1^\prime \mid (h_3[5]; p), H_6^\prime \mid (h_4[6]; p), H_7^\prime \mid (o_3[5], o_4[6], n[11]).O_2^\prime \mid (h_5[7]; p), H_8^\prime \mid (o_3[7], o_6[8], n).O_3^\prime}{i2 \mid W \mid W}
\]

Now the prom rewrite rule must be applied before we derive the next concerted transition: we promote the weak bond 11 of the carbon to a stronger bond on $c_4$, which has become available:

\[
\Rightarrow (c_1[1], c_2[2], c_3[3], c_4[11]; p).C^\prime \mid (h_1[1]; p), H_4^\prime \mid (h_2[2]; p), H_5^\prime \mid (o_3[3], o_2, n).O_1^\prime \mid (h_3[5]; p), H_6^\prime \mid (h_4[6]; p), H_7^\prime \mid (o_3[5], o_4[6], n[11]).O_2^\prime \mid (h_5[7]; p), H_8^\prime \mid (o_3[7], o_6[8], n).O_3^\prime \mid \text{prom}) \mid W
\]

The next step is to form a bond between $O_3$ and either $H_3$ or $H_4$. We bond with $H_3$ with the key 12 and break the bond 5, producing a pair of concerted actions. We then move a weak bond 11 on $n$ in $O_2$ to a stronger bond on $o_3$ (which has become available) by rewrite rule move-r. We also promote the weak bond 12 in $H_3$ to a strong bond on $h_3$ by rewrite rule prom, giving overall this transition:

\[
\frac{\{np[12], h_3o_3[5]\}}{(c_1[1], c_2[2], c_3[3], c_4[11]; p).C^\prime \mid (h_1[1]; p), H_4^\prime \mid (h_2[2]; p), H_5^\prime \mid (o_3[3], o_2, n).O_1^\prime \mid (h_3[12]; p), H_6^\prime \mid (h_4[6]; p), H_7^\prime \mid (o_3[5], o_4[6], n).O_2^\prime \mid (h_5[7]; p), H_8^\prime \mid (h_6[8]; p), H_9^\prime \mid (o_3[7], o_6[8], n[12]).O_3^\prime}{i3 \mid H_3O \mid W}
\]

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 12), we obtain the
following:

\[
\begin{align*}
\{n,p[13],h_5,o_5[12]\} & \rightarrow (c_1[1], c_2[2], c_3[3], c_4[11]; p).C' | (h_1[1]; p).H'_1 | (h_2[2]; p).H'_2 | \\
& \quad (o_1[3], o_2, n[13]).O'_1 | (h_3; p[13]).H'_3 | (h_4[6]; p).H'_4 | (o_3[11], o_4[6], n).O'_2 \\
& \quad | (h_5[7]; p).H'_5 | (h_6[8]; p).H'_6 | (o_5[7], o_6[8], n).O'_3
\end{align*}
\]

and promoting and moving the bond 13 in \(H_3\) and \(O_1\), respectively, to strong bonds we obtain the final product (methanediol \(\text{CH}_2(\text{OH})_2\)) and two waters:

\[
\Rightarrow \quad (c_1[1], c_2[2], c_3[3], c_4[11]; p).C' | (h_1[1]; p).H'_1 | (h_2[2]; p).H'_2 \\
& \quad | (o_1[3], o_2[13], n).O'_1 | (h_3[13]; p).H'_3 | (h_4[6]; p).H'_4 | (o_3[11], o_4[6], n).O'_2 \\
& \quad | (h_5[7]; p).H'_5 | (h_6[8]; p).H'_6 | (o_5[7], o_6[8], n).O'_3
\]

MD | W | W

Note that the \(n, p\) actions are ready again and all the existing bonds are on strong actions. So we now can reverse the reaction by \(O_3\) abstracting a hydrogen from \(H_4\) or \(H_5\), and all the way to the initial state. Moreover, let us inspect the bonds 4, 5 and 7 which are broken during the reaction. The bonds were formed during the initial bonding of the atoms. They are broken as a result of application of our new general prefixing operator. This operator enables the reaction to work forwards without external control. Indeed the original order of the formation of the bonds is completely irrelevant for the reaction to work.

6.2. The base-catalysed path

We now consider the base-catalysed path described in Figure 12. The path is via FA | W | HO | \(H_5\)O and i3 | \(H_3\)O | W to MD | W | W in Figures 14-15. The path needs three water molecules. We shall show the application of the promotion or move rules without explaining them in detail. We start with the initial system (remembering that restrictions are not shown):

\[
\begin{align*}
(c_1[1], c_2[2], c_3[3], c_4[4]; p).C' | (h_1[1]; p).H'_1 | (h_2[2]; p).H'_2 | (o_1[3], o_2[4], n).O'_1 \\
& \quad | (h_3[5]; p).H'_3 | (h_4[6]; p).H'_4 | (o_3[5], o_4[6], n).O'_2 \\
& \quad | (h_5[7]; p).H'_5 | (h_6[8]; p).H'_6 | (o_5[7], o_6[8], n).O'_3 \\
& \quad | (h_7[9]; p).H'_7 | (h_8[10]; p).H'_8 | (o_7[9], o_8[10], n).O'_4
\end{align*}
\]

FA | W | W | W
6.3. The acid-catalysed path

Next we consider the acid-catalysed path described in Figure 11. The path is via $i_6 \mid W \mid HO \mid W$ and $i_8 \mid HO \mid W$ in Figures 14, 15. The path uses three
water molecules as in the base-catalysed path. We shall apply the promotion or move rules as needed. We start with the initial system:

\[
(c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[7]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[7], o_6[8], n[12]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\frac{\{np[13], h_3.o_6[7]\}}{\rightarrow} (c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[12], o_6[8], n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\Rightarrow (c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[12], o_6[8], n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\frac{\{np[11], h_3.o_6[4]\}}{\rightarrow} (c_1[1], c_2[2], c_3[3], c_4[11]; p[11]).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[12], o_6[8], n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\Rightarrow (c_1[1], c_2[2], c_3[3], c_4[11]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[12], o_6[8], n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\frac{\{np[14], h_3.o_6[8]\}}{\rightarrow} (c_1[1], c_2[2], c_3[3], c_4[11]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p[14]).H'_6 \mid (o_5[12], o_6, n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n[14]).O'_4

i6 | W | HO | H\textsubscript{3}O

The last rewrite is by rule move-r. The reaction continues as follows:

\[
\frac{\{np[11], h_3.o_6[4]\}}{\rightarrow} (c_1[1], c_2[2], c_3[3], c_4[11]; p[11]).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[12], o_6[8], n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\Rightarrow (c_1[1], c_2[2], c_3[3], c_4[11]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p).H'_6 \mid (o_5[12], o_6[8], n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4

\[
\frac{\{np[14], h_3.o_6[8]\}}{\rightarrow} (c_1[1], c_2[2], c_3[3], c_4[11]; p).C' \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2
\]

| (o_1[3], o_2[4], n[13]).O'_1 \mid (h_3[12]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3, o_4[6], n).O'_2
| (h_5[13]; p).H'_5 \mid (h_6[8]; p[14]).H'_6 \mid (o_5[12], o_6, n[11]).O'_3
| (h_7[9]; p).H'_7 \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n[14]).O'_4

i8 | HO | W
move-r

\[ (c_1[1], c_2[2], c_3[3], c_4[11] ; p).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \]

\[ (o_1[3], o_2[13], n).O_1' \mid (h_3[12]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3, o_4[6], n).O_2' \]

\[ (h_5[13]; p).H_5' \mid (h_6[14]; p).H_6' \mid (o_5[12], o_6[11], n).O_3' \]

\[ (h_7[9]; p).H_7' \mid (h_8[10]; p).H_8' \mid (o_7[9], o_8[10], n[14]).O_4' \]

\[ \text{MD} \mid \text{HO} \mid \text{H}_3\text{O} \]

Finally we reach the final state of a methanediol and two waters:

\[ \frac{(np[5], h_5[14])}{(c_1[1], c_2[2], c_3[3], c_4[11] ; p).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \}

\[ (o_1[3], o_2[13], n).O_1' \mid (h_3[12]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3, o_4[6], n[5]).O_2' \]

\[ (h_5[13]; p).H_5' \mid (h_6[14]; p).H_6' \mid (o_5[12], o_6[11], n).O_3' \]

\[ (h_7[9]; p).H_7' \mid (h_8[5]; p).H_8' \mid (o_7[9], o_8[10], n[14]).O_4' \]

\[ \text{MD} \mid \text{W} \mid \text{W} \]

We return to the transition from the intermediate 6 to the intermediate 8 in Figure [11]. Once the hydrogen is bound to the oxygen in the compound 6 we have a so-called resonance, where the positive charge can be on the oxygen or the carbon (or in between) and the structure resonates between the intermediate 6 and 7 (indicated by the dashed arrow in Figure [11]). We can model this movement between the two intermediates as follows: starting from i6 \mid W \mid HO \mid W we break bond 4 without forming a new bond at the same time, and perform a move-r rewrite on O1:

\[ \frac{c_4[2]}{c_4[4]} (c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \]

\[ (o_1[3], o_2[13], n).O_1' \mid (h_3[12]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3, o_4[6], n).O_2' \]

\[ (h_5[13]; p).H_5' \mid (h_6[8]; p).H_6' \mid (o_5[12], o_6[8], n).O_3' \]

\[ (h_7[9]; p).H_7' \mid (h_8[10]; p).H_8' \mid (o_7[9], o_8[10], n).O_4' \]

\[ \text{i7} \mid \text{W} \mid \text{HO} \mid \text{W} \]

The movement from i7 to i6 is gotten by creating the bond on c4 and n of O1 with the key 4:

\[ \frac{c_4[n]}{c_4[4]} (c_1[1], c_2[2], c_3[3], c_4[4]; p).C' \mid (h_1[1]; p).H_1' \mid (h_2[2]; p).H_2' \]

\[ (o_1[3], o_2[13], n[4]).O_1' \mid (h_3[12]; p).H_3' \mid (h_4[6]; p).H_4' \mid (o_3, o_4[6], n).O_2' \]

\[ (h_5[13]; p).H_5' \mid (h_6[8]; p).H_6' \mid (o_5[12], o_6[8], n).O_3' \]

\[ (h_7[9]; p).H_7' \mid (h_8[10]; p).H_8' \mid (o_7[9], o_8[10], n).O_4' \]
Note that the last compound is equivalent chemically to i6 | W | HO | W, and it is also identical syntactically to i6 | W | HO | W except the keys 4 and 13 are swapped in O1.

We have noted before that the main path and the base-catalysed path form a diamond: we can get from FA | W | W | W to i3 | H3O | W either via i2 | W | W or via FA | W | HO | H3O. We underline that the resulting methanediol processes are syntactically identical. However, there is no such tight correspondence between methanediol in the main path and methanediol in the acid-catalysed path. In the main path we bind a H2O to the carbon, and then use another H2O as a proton shuttle to move a hydrogen. The H2O which is used as a proton shuttle is unchanged. In the acid-catalysed path we bind a hydrogen to the formaldehyde first, bind a H2O to it and then remove one of these hydrogens and put it back on a water. Hence, the two methanediol processes are not identical but they represent chemically the same compound. We shall explain later how this equivalence might be formally defined.

6.4. Other paths

We now discuss the remaining reactions in Figure [14]

The compounds 13 and 14 in Figure [13] are of particular interest since they have only one path leading to and out of them. Firstly we can get from i3 | H3O | W to i13 | H3O | H3O as follows:

\[
\begin{align*}
& (c_{1}[1], c_{2}[2], c_{3}[3], c_{4}[11]; p).C' | \ (h_{1}[1]; p).H_1' | \ (h_{2}[2]; p).H_2' | \ (o_{1}[3], o_{2}, n).O_1' \\
& | \ (h_{3}[12]; p).H_3' | \ (h_{4}[6]; p).H_4' | \ (o_{3}[11], o_{4}[6], n).O_2' \\
& | \ (h_{5}[7]; p).H_5' | \ (h_{6}[8]; p).H_6' | \ (o_{5}[7], o_{6}[8], n[12]).O_3' \\
& | \ (h_{7}[9]; p).H_7' | \ (h_{8}[10]; p).H_8' | \ (o_{7}[9], o_{8}[10], n).O_4' \\
& \text{i3 | H3O | W}
\end{align*}
\]

\[
\text{\text{np[i13,H4O6]}} \rightarrow \left\{ (c_{1}[1], c_{2}[2], c_{3}[3], c_{4}[11]; p).C' | \ (h_{1}[1]; p).H_1' | \ (h_{2}[2]; p).H_2' \\
| \ (o_{1}[3], o_{2}, n).O_1' | \ (h_{3}[12]; p).H_3' | \ (h_{4}[p][13]).H_4' | \ (o_{3}[11], o_{4}, n).O_2' \\
| \ (h_{5}[7]; p).H_5' | \ (h_{6}[8]; p).H_6' | \ (o_{5}[7], o_{6}[8], n[12]).O_3' \\
| \ (h_{7}[9]; p).H_7' | \ (h_{8}[10]; p).H_8' | \ (o_{7}[9], o_{8}[10], n[13]).O_4' \Rightarrow \right.
\]

\[
\begin{align*}
& \Rightarrow (c_{1}[1], c_{2}[2], c_{3}[3], c_{4}[11]; p).C' | \ (h_{1}[1]; p).H_1' | \ (h_{2}[2]; p).H_2' \\
& | \ (o_{1}[3], o_{2}, n).O_1' | \ (h_{3}[12]; p).H_3' | \ (h_{4}[13]; p).H_4' | \ (o_{3}[11], o_{4}, n).O_2' \\
& | \ (h_{5}[7]; p).H_5' | \ (h_{6}[8]; p).H_6' | \ (o_{5}[7], o_{6}[8], n[12]).O_3' \\
& | \ (h_{7}[9]; p).H_7' | \ (h_{8}[10]; p).H_8' | \ (o_{7}[9], o_{8}[10], n[13]).O_4' \\
& \text{i13 | H3O | H3O}
\end{align*}
\]

We can also reverse back to i3 | H3O | W by performing concerted actions fol-
lowed by a promotion and a move:

\[
\begin{align*}
&\{np[6], h[n][13]\} \rightarrow \{c[1], c[2], c[3], c[4][11]; p\}.C' \mid (h[1][1]; p).H'_4 \mid (h[2][2]; p).H'_2 \\
&| (a[3], o[2], n).O'_1 \mid (h[3][12]; p).H'_3 \mid (h[4][6]; p).H'_4 \mid (o[3][11], o[4][6], n).O'_2 \\
&| (h[5][7]; p).H'_5 \mid (h[6][8]; p).H'_6 \mid (o[3][7], o[6][8], n[12]).O'_3 \\
&| (h[7][9]; p).H'_7 \mid (h[8][10]; p).H'_8 \mid (o[7][9], o[8][10], n).O'_4
\end{align*}
\]

\[
\Rightarrow (c[1], c[2][2], c[3][3], c[4][11]; p).C' \mid (h[1][1]; p).H'_4 \mid (h[2][2]; p).H'_2 \\
&| (a[3], o[2], n).O'_1 \mid (h[3][12]; p).H'_3 \mid (h[4][6]; p).H'_4 \mid (o[3][11], o[4][6], n).O'_2 \\
&| (h[5][7]; p).H'_5 \mid (h[6][8]; p).H'_6 \mid (o[3][7], o[6][8], n[12]).O'_3 \\
&| (h[7][9]; p).H'_7 \mid (h[8][10]; p).H'_8 \mid (o[7][9], o[8][10], n).O'_4
\]

Furthermore can also get from i8 | HO | W to i14 | HO | HO:

\[
\begin{align*}
&\{np[14], h[n][9]\} \rightarrow (c[1], c[2][2], c[3][3], c[4][11]; p).C' \mid (h[1][1]; p).H'_4 \mid (h[2][2]; p).H'_2 \\
&| (a[3], o[2][13], n[14]).O'_1 \mid (h[3][12]; p).H'_3 \mid (h[4][6]; p).H'_4 \mid (o[3][11], o[4][6], n).O'_2 \\
&| (h[5][13]; p).H'_5 \mid (h[6][8]; p).H'_6 \mid (o[3][12], o[6][8], n[11]).O'_3 \\
&| (h[7][14]; p).H'_7 \mid (h[8][10]; p).H'_8 \mid (o[7][9], o[8][10], n).O'_4
\end{align*}
\]

\[
\Rightarrow (c[1], c[2][2], c[3][3], c[4][11]; p).C' \mid (h[1][1]; p).H'_4 \mid (h[2][2]; p).H'_2 \\
&| (a[3], o[2][13], n[14]).O'_1 \mid (h[3][12]; p).H'_3 \mid (h[4][6]; p).H'_4 \mid (o[3][11], o[4][6], n).O'_2 \\
&| (h[5][13]; p).H'_5 \mid (h[6][8]; p).H'_6 \mid (o[3][12], o[6][8], n[11]).O'_3 \\
&| (h[7][14]; p).H'_7 \mid (h[8][10]; p).H'_8 \mid (o[7][9], o[8][10], n).O'_4
\]

\[
i3 \mid H_3O \mid W
\]

\[
i8 \mid HO \mid W
\]

And we can reverse back to i8 | HO | W in a very much the same way as from i13 | H_3O | i3; H_3O | W.

The reaction from i6 | W | HO | W to i6 | HO | HO | H_3O and its inverse is the autoprotolysis of water, and work independently of the rest. The corresponding applies to the reaction from i7 | HO | HO | H_3O to i7 | W | HO | W and its reversal.

The step from i6 | HO | HO | H_3O to i7 | HO | HO | H_3O involves the intermediate i6 and i8 only (as in Figure 11) and no other compounds, so the reaction is as described before.
The reaction from i2 | W | W to i2 | HO | H3OW is also an autoprotolysis of water.

There are seven more reactions possible, all via concerted actions. We only describe one of them, namely from FA | W | W to i6 | W | HO | W:

\[
\begin{align*}
(c_1[1], c_2[2], c_3[3], c_4[4]; p).C' & \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2 \\
| (o_1[3], o_2[4], n).O'_1 & \mid (h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2 \\
| (h_5[7]; p).H'_5 & \mid (h_6[8]; p).H'_6 \mid (o_5[7], o_6[8], n).O'_3 \\
| (h_7[9]; p).H'_7 & \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4
\end{align*}
\]

\[
\begin{align*}
\{np[13], h_{so[7]}\} \rightarrow (c_1[1], c_2[2], c_3[3], c_4[4]; p).C' & \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2 \\
| (o_1[3], o_2[4], n[13]).O'_1 & \mid (h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2 \\
| (h_5[13]; p).H'_5 & \mid (h_6[8]; p).H'_6 \mid (o_5, o_6[8], n).O'_3 \\
| (h_7[9]; p).H'_7 & \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4
\end{align*}
\]

\[
\Rightarrow (c_1[1], c_2[2], c_3[3], c_4[4]; p).C' & \mid (h_1[1]; p).H'_1 \mid (h_2[2]; p).H'_2 \\
| (o_1[3], o_2[4], n[13]).O'_1 & \mid (h_3[5]; p).H'_3 \mid (h_4[6]; p).H'_4 \mid (o_3[5], o_4[6], n).O'_2 \\
| (h_5[13]; p).H'_5 & \mid (h_6[8]; p).H'_6 \mid (o_5, o_6[8], n).O'_3 \\
| (h_7[9]; p).H'_7 & \mid (h_8[10]; p).H'_8 \mid (o_7[9], o_8[10], n).O'_4
\]

6.5. Chemical equivalence

We have seen that the system of FA with three W evolves to MA with two W via different sequences of concerted actions. Some paths end up at the same process, others lead to somewhat different syntactically processes. We have explained that these different processes represent the same chemical compounds, and we have explained that the difference originates as a side effect of using subscripted copies of atoms, where both the action names and process names are uniformly subscripted, and of using keys. Now we present informally our notion of chemical equivalence of systems of subscripted copies of processes. This essentially amounts to performing alpha conversion on subscripts and keys in processes. We leave the precise formulation for future research. Here, we only demonstrate how the methanediol obtained via the main path is equivalent chemically to the methanediol gotten via the acid-catalysed path by performing a number of atom swaps or key changes.

It is clear that a copy \( H_3 \) of hydrogen can be replaced in a compound by \( H_5 \) without changing the chemical meaning of the compound provided that \( H_5 \) does not appear in the compound in the first place. Also, it is irrelevant what precise keys are used in a process as we can replace a key by another fresh key without changing the meaning of the process. Hence, using alpha conversion, we can define a swap operation of subscripted versions of the same process and
We take the second MD and we swap the H bonded to O on the key 5. Finally we swap this. This is identical to the MD obtained via the main path if we replace also all the keys 12 and 5 by the keys 7 and 8, respectively, which are fresh.

and the MD process produced via acid-catalysed path:

\[ \text{We take the second MD and we swap } H \text{ with } H_5 \text{. This results in } H_3 \text{ becoming bonded to } O_3 \text{ with key 13, and } H_5 \text{ becoming bonded to } O_3 \text{ with key 12. Note that the same result is obtained by simply swapping keys 12 and 13 in } H_3 \text{ with } H_5 \text{. This gives us:} \]

\[ c_1[1], c_2[2], c_3[3], c_4[11]; p).C' | (h_1[1]; p).H'_1 | (h_2[2]; p).H'_2 | (o_1[3], o_2[13]), n).O'_1 \\
| (h_3[13]; p).H'_3 | (h_4[6]; p).H'_4 | (o_3[11], o_4[6]), n).O'_2 \\
| (h_5[7]; p).H'_5 | (h_6[8]; p).H'_6 | (o_5[7], o_6[8]), n).O'_3 \\
| (h_7[9]; p).H'_7 | (h_8[10]; p).H'_8 | (o_7[9],o_8[10]), n).O'_4 \]

Next, we swap O_3 (together with H_5 which is bonded to O_3) with O_2 (and H_4 which is bonded to O_2). This is done by swapping the key 11 on o_6 of O_3 with the key 5 on o_5 of O_2. This gives us

\[ c_1[1], c_2[2], c_3[3], c_4[11]; p).C' | (h_1[1]; p).H'_1 | (h_2[2]; p).H'_2 | (o_1[3], o_2[13]), n).O'_1 \\
| (h_3[13]; p).H'_3 | (h_4[6]; p).H'_4 | (o_3[11], o_4[6]), n).O'_2 \\
| (h_5[12]; p).H'_5 | (h_6[10]; p).H'_6 | (o_5[12], o_6[11]), n).O'_3 \\
| (h_7[9]; p).H'_7 | (h_8[5]; p).H'_8 | (o_7[9],o_8[10]), n).O'_4 \]

Finally we swap H_6 and H_8, which is the same as swapping keys 5 and 10 in H_6 and H_8, producing the following:

\[ c_1[1], c_2[2], c_3[3], c_4[11]; p).C' | (h_1[1]; p).H'_1 | (h_2[2]; p).H'_2 | (o_1[3], o_2[13]), n).O'_1 \\
| (h_3[13]; p).H'_3 | (h_4[6]; p).H'_4 | (o_3[11], o_4[6]), n).O'_2 \\
| (h_5[12]; p).H'_5 | (h_6[5]; p).H'_6 | (o_5[12], o_6[5]), n).O'_3 \\
| (h_7[9]; p).H'_7 | (h_8[10]; p).H'_8 | (o_7[9],o_8[10]), n).O'_4 \]

This is identical to the MD obtained obtained via the main path if we replace also all the keys 12 and 5 by the keys 7 and 8, respectively, which are fresh.
6.6. Evaluation

In this subsection we evaluate the suitability of CCB in the modelling of covalent reactions as exemplified by the hydration of formaldehyde in water. We first consider if all chemically valid interactions between the compounds of the reaction can be represented well in our calculus. We have seen in Figure 11 the resonance between the intermediates i6 and i7. We are able to model appropriately the movement between i6 and i7 by transitions that break a bond or create a bond on its own. What is less clear is why this bond break should happen at this point and not elsewhere. We are unable to represent the movement from i7 to i8, however, we model appropriately the evolution from i6 to i8 via a concerted actions transition. Apart from the reaction steps described above, all other steps can be represented well in our calculus.

The other way of assessing the suitability of CCB for this type of reactions is to ask if our calculus enables transitions which do not occur in reality. We notice that some bonds can break on their own as, for example, in water molecules. Although this does not happen in reality an alternative process exists which produces the same products H and HO. This is the autoprotolysis of water which we have described earlier and in [16]. So the breaking of water molecules could be treated as harmless. Also a bond between carbon and oxygen in methanediol can break on its own, without being accompanied by a new bond formation. This is a consequence of allowing the break of the double bond in the compound i6 on its own. The difference between these two cases is not properly modelled in our calculus. However, if the carbon-oxygen bond in MD breaks the carbon could bond immediately afterwards to the water, and we would get the compound i8, which is a valid intermediate, so this break of the carbon-oxygen bond does not cause a problem. Our model also allows the interaction of the methanediol with a water molecule in the final system MD | W | W. Since the carbon C has the action p ready, and the oxygen O3 in the water has the action n ready, and n, p is a part of our synchronisation function, a bond between C and O3 can be created. The reason why this type of interaction does not occur in reality is that the four groups around the carbon shield it from any interaction (as opposed to the three groups on the formaldehyde, of which two are relatively small hydrogens). This is a steric effect, the effect due to atoms occupying space and preventing other atoms from moving. Our calculus does not model spacial arrangement of atoms well enough; we aim to address this in the future.

Finally we discuss how CCB compares to other calculi used for the modelling biochemical reactions. We start with the process calculi based on CCS such as CCS-R [7, 8] or CCSK [25, 26]. One of the advantages of CCB are the concerted actions transitions. This permits us to represent faithfully that in covalent reactions some compounds are unstable and cannot exist independently of the reaction. A double bonded hydrogen is an example: it exists only “during” concerted actions transition. Other calculi cannot represent this point; a double

---

3We have used a similar mechanism of C interacting with O to get from the compound 1 to the compound 2 in Figure 10.
bonded hydrogen is a valid compound in those calculi and two transitions are used instead of our concerted transition.

Graph-based formalisms such as the κ-calculus use rewrite rules to encode which bonds can be created, which bonds can be broken, and in which states. So the designer of the rules influences what reactions can occur and how they occur. In CCB, however, the processes contain all the information needed to drive reactions, of course in accordance with the given SOS, without any external control or guidance.

7. Conclusion

We have introduced a reversible process calculus CCB with a novel prefixing operator which is inspired by the mechanism of covalent bonding. This mechanism 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. We have also proposed rewrite rules that model promotion of weak bonds to strong bonds. Our prefixing operator 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 via concerted actions transitions and promotion rules. We have presented a detailed CCB model of 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.

Acknowledgements

We are grateful to the referees for their detailed and helpful comments and suggestions. The authors acknowledge partial support of COST Action IC1405 on Reversible Computation - extending horizons of computing. The second author acknowledges the support by the University of Leicester in granting him Academic Study Leave, and thanks Nagoya University for support during the study leave.

References


Appendix

**Proposition 4.** Let $P$ be consistent. If $P \xrightarrow{\mu[k],\nu[l]} P'$ then $k \notin \text{keys}(P)$, $l \in \text{keys}(P)$ and $\text{keys}(P') = (\text{keys}(P) \cup \{k\}) \setminus \{l\}$ for all $P'$.

**Proof.** By induction on the depth of the inference tree of $P \xrightarrow{\mu[k],\nu[l]} Q$.

1. Base case follows for processes with the inference tree of depth 0.
2. Inductive hypothesis: We assume that Proposition 4 holds for all sub-processes $R$ of $P$ and all $\nu[l]$ and all $\mu[k]$, namely if $R$ is a consistent process and $R \xrightarrow{\mu[k],\nu[l]} R'$ for some $R'$ then $k \notin \text{keys}(R)$, $l \in \text{keys}(R)$ and $\text{keys}(R') = (\text{keys}(R) \cup \{k\}) \setminus \{l\}$.
3. Induction step: We show that Proposition 4 holds for $P$. For this we consider cases depending on the structure of $P$:

   (a) $P \equiv R | Q$: There are two cases here:
      
      i. The transition is by the rule concert in Figure 3. Assume $P$ is consistent and $P \xrightarrow{\mu[k],\nu[l]} P'$ for some $P'$. Since $P \xrightarrow{\mu[k],\nu[l]} P'$, by rule concert, we have $R \xrightarrow{\mu[k]} R', Q \xrightarrow{\nu[l]} Q'$ or $Q \xrightarrow{\nu_2[l]} Q', R' \xrightarrow{\nu_1[l]} R'', Q' \xrightarrow{\nu_2[l]} Q''$, $\gamma(\nu_1, \nu_2) = \nu$ and $\gamma(\mu[k], \nu_2) = \mu$ must hold without loss of generality. Also, $P' \equiv R'' | Q'$. We can calculate $\text{keys}(P)$: $\text{keys}(P) = \text{keys}(R) \cup \text{keys}(Q)$. Since $R$ and $Q$ can both perform a transition with $k$ we know, by argument similar to that used in the proof of Proposition 4.1, that $k \notin \text{keys}(R)$ and $k \notin \text{keys}(Q)$, and therefore $k \notin \text{keys}(P)$ as required. Since $R'$ and $Q'$ can perform undoing of actions with the key $l$, we know that $l \in \text{keys}(R')$ and $l \in \text{keys}(Q')$ by Proposition 4.2. Transitions of $R$ and $Q$ to $R'$ and $Q'$ respectively do not involve $l$, so $l \in \text{keys}(R)$ and $l \in \text{keys}(Q)$ most hold. Since $\text{keys}(P) = \text{keys}(R \cup Q)$ we deduce that $l \in \text{keys}(P)$ as required. Because of Proposition 4.1 and Proposition 4.2 we can calculate $\text{keys}(Q')$: $\text{keys}(Q'') = \text{keys}(Q') \setminus \{l\} = (\text{keys}(Q) \cup \{k\}) \setminus \{l\}$, $\text{keys}(R')$: $\text{keys}(R'') = \text{keys}(R') \setminus \{l\} = (\text{keys}(R) \cup \{k\}) \setminus \{l\}$, and $\text{keys}(P')$: $\text{keys}(P') = \text{keys}(R'') \cup \text{keys}(Q'') = (\text{keys}(R) \cup \{k\}) \setminus \{l, \text{keys}(Q) \cup \{k\}\} \setminus \{l\} = (\text{keys}(R) \cup \text{keys}(Q) \cup \{k\}) \setminus \{l\} = \text{keys}(P) \cup \{k\} \setminus \{l\}$ as required.

   ii. The transition is by concert par rule in Figure 3. We assume without loss of generality $R \xrightarrow{\mu[k],\nu[l]} R'$ and fsh$[l](Q)$, and $P' \equiv R' | Q$. By the inductive hypothesis we have $k \notin \text{keys}(R)$, $l \in \text{keys}(R)$ and $\text{keys}(R') = (\text{keys}(R) \cup \{k\}) \setminus \{l\}$. Since $\text{keys}(P) = \text{keys}(R | Q) = \text{keys}(R) \cup \text{keys}(Q) \cup \text{fsh}[k](Q)$ and $k \notin \text{keys}(R)$ we deduce that $k \notin \text{keys}(P)$. Also since $l \in \text{keys}(R)$ and $\text{keys}(P) = \text{keys}(R | Q) = \text{keys}(R) \cup \text{keys}(Q)$ and we deduce that $l \in \text{keys}(P)$. Hence, we have $\text{keys}(P') = \text{keys}(R') \cup \text{keys}(Q) = (\text{keys}(R) \cup \text{keys}(Q) \cup \{k\}) \setminus \{l\} \cup \text{keys}(Q)$. Since $l \notin \text{keys}(Q)$ by concert par, we have
(keys(R) ∪ \{k\}) \setminus \{l\} \cup \text{keys}(Q) = (\text{keys}(R \parallel Q) \cup \{k\}) \setminus \{l\} = 
(\text{keys}(P) \cup \{k\}) \setminus \{l\} \text{ as required.}

(b) \ P \equiv (t; b).R: The transition is by concert act rule in Figure 3. Assume
\ P \text{ is consistent and } P \xrightarrow{\mu | k | \omega | \nu} P'. \text{ We deduce that } P' \equiv (t; b).R' \text{ for }
\text{some } R'. \text{ Recall that } t \text{ denotes a sequence of past actions, namely an }
\text{element from } A\mathcal{K}^+. \text{ The premises of concert act ensure that } \text{fsh}[k](t) \text{ and }
\text{and } P \xrightarrow{\mu | k | \omega | \nu} R'. \text{ The process } R \text{ is consistent since } P \text{ is consistent.}
\text{Since } R \text{ is consistent and } R \xrightarrow{\mu | k | \omega | \nu} R' \text{ then, by the inductive hypothesis, } k \notin \text{keys}(R), l \in \text{keys}(R) \text{ and }
\text{keys}(R') = (\text{keys}(R) \cup \{k\}) \setminus \{l\}. \text{ Since } k \notin \text{keys}(R) \text{ and fsh}[k](t) \text{ we obtain } k \notin \text{keys}(t; b).R = \text{keys}(P) \text{ as required. Since } l \in \text{keys}(R) \text{ we obtain } l \in \text{keys}(t; b).R = \text{keys}(P) \text{ as required. It is clear by the rules act2 and concert act that }
\text{since } l \in \text{keys}(R) \text{ the key } l \text{ is not among the keys in } t \text{ in } (t; b).R. \text{ We now calculate } \text{keys}(P') = \text{keys}(P') = \text{keys}(t; b).R' = k(t) \cup k(b) \cup \text{keys}(R') = k(t) \cup k(b) \cup \text{keys}(R) \cup \{k\} \setminus \{l\} = (k(t) \cup k(b) \cup \text{keys}(R) \cup \{k\}) \setminus \{l\} = (\text{keys}(t; b).R \cup \{k\}) \setminus \{l\} = (\text{keys}(P) \cup \{k\}) \setminus \{l\} \text{ as required.}

(c) \ P \equiv R \setminus L: The transition must be by concert res from Figure 3.
\text{Assume } P \text{ is consistent and } P \xrightarrow{\mu | k | \omega | \nu} P'. \text{ Since } P \xrightarrow{\mu | k | \omega | \nu} P' \text{ by rule concert res we obtain } R \xrightarrow{\mu | k | \omega | \nu} R' \text{ and } \mu, \nu \notin L \cup
\{L\}. \text{ Since } k \notin \text{keys}(R) \text{ by the inductive hypothesis it follows that }
k \notin \text{keys}(R \setminus L) \text{ since restriction does not change the keys of the }
\text{process by definition of keys. Since } l \in \text{keys}(R) \text{ by the inductive hypothesis it follows that } l \in \text{keys}(R \setminus L). \text{ Also } \text{keys}(P) = \text{keys}(R) \text{ and }
\text{keys}(P') = \text{keys}(R') \text{ according to the definition of keys. Since by }
\text{the inductive hypothesis } \text{keys}(R') = \text{keys}(R) \cup \{k\} \text{ we can calculate }
\text{keys}(P') = \text{keys}(R' \setminus L) = \text{keys}(R') = (\text{keys}(R) \cup \{k\}) \setminus \{l\} = 
(\text{keys}(P) \cup \{k\}) \setminus \{l\} \text{ as required.}

(d) \ There are no concerted transitions for } P \equiv (s; c).R \text{ and } P \equiv S \text{ with } S \equiv R.

\textbf{Proposition 6.} \text{If } t_1 \equiv P \xrightarrow{\mu | k} P' \text{ and } t_2 \equiv P \xrightarrow{\nu | l} P'', \text{ then either } t_1 \text{ and } t_2 \text{ are concurrent or } k \in \text{cau}(P'', l).

\textbf{PROOF.} \text{By induction on the depth of inference trees for transition of } P.

1. \text{Base case: Obvious.}
2. \text{Inductive hypothesis: We assume that for all subprocesses } R \text{ of } P \text{ and all }
\theta | m, \theta | n, \text{ if } R \text{ is a consistent process, } t_1 \equiv R \xrightarrow{\theta | m} R' \text{ and } t_2 \equiv R \xrightarrow{\theta | n} R'' \text{ then either } t_1 \text{ and } t_2 \text{ are concurrent or } m \in \text{cau}(P'', n).
3. \text{Induction step: We show Proposition 6 for } P. \text{ By Definition 5 there is }
either an } M \text{ such that } M \neq P, P' \xrightarrow{\nu | l} M \text{ and } P'' \xrightarrow{\mu | k} M, \text{ or } k \in \text{cau}(P'', l) \text{ holds. We consider cases based on the structure of } P.
(a) \( P \equiv (s; b).R \) with \( s \) containing at least one past action and at least one fresh action. The transitions \( t_1 \) and \( t_2 \) are by rev \text{act1} respectively \text{act1}. Since there is nothing in the rules giving any of them precedence, the transitions are concurrent as required. With \( s' \) being the sequence obtained from \( s \) by removing actions \( \nu \) and \( \mu[k] \) we have \( t_1 \equiv (s', \nu, \mu[k]; b).R \xrightarrow{\mu[k]} (s', \nu, \mu; b).R \) and \( t_2 \equiv (s', \nu, \mu[k]; b).R \xrightarrow{\nu[l]} (s', \nu[l], \mu[k]; b).R \). We now deduce \( M \equiv (s', \nu[l], \mu; b).R \) and the properties required for \( \nu \) and \( \mu[k] \) being concurrent hold: namely
\[
(s', \nu, \mu; b).R \xrightarrow{\nu[l]} M \quad \text{and} \quad (s', \nu[l], \mu[k]; b).R \xrightarrow{\mu[k]} M.
\]

(b) \( P \equiv (s; b).R \) where \( s \) contains only fresh actions: We cannot have the required transition \( t_2 \). Hence Proposition 6 is vacuously valid.

(c) \( P \equiv (s; b).R \) where \( s \) contains only past actions: There is \( R \xrightarrow{\nu[l]} R' \), and a \( \mu[k] \in s \) so that \( t_2 \equiv (s; b).R \xrightarrow{\nu[l]} (s; b).R' \) and \( t_1 \equiv (s; b).R \xrightarrow{\mu[k]} (s'; b).R \) for some \( s' \). These transitions are not concurrent since doing any action in \( R \) prevents undoing of actions in \( s \) and vice versa. Hence \( k \in \text{cau}(P'', l) \) holds with \( P'' \equiv (s; b).R' \).

(d) \( P \equiv Q \mid R \): There are three cases:

i. \( t_1 \) by rule rev \text{par} and \( t_2 \) by rule \text{par}. There are two sub-cases:

A. Transitions in the same subprocess: Assume without loss of generality \( Q \xrightarrow{\mu[k]} Q' \) and \( Q \xrightarrow{\nu[l]} Q'' \). By the inductive hypothesis these transitions are either concurrent or \( k \in \text{cau}(Q'', l) \).

- concurrent transitions: By the inductive hypothesis there is an \( N \) so that \( Q' \xrightarrow{\nu[l]} N \) and \( Q'' \xrightarrow{\mu[k]} N \). We can conclude by using rule rev \text{par} that \( Q' \mid R \xrightarrow{\nu[l]} N \mid R \) and \( Q'' \mid R \xrightarrow{\mu[k]} N \mid R \). Letting \( M \equiv N \mid R \) we get the required result.

- \( k \in \text{cau}(Q'', l) \): Since \( \text{cau}(P \mid Q, k) = \text{cau}(P, k) \cup \text{cau}(Q, k) \) according to Definition 7 we get \( \text{cau}(Q'' \mid R, l) = \text{cau}(Q'', l) \cup \text{cau}(R, l) \). If \( k \in \text{cau}(Q'', l) \) then \( k \in \text{cau}(Q'', l) \cup \text{cau}(R, l) \) must be true as well and, hence, \( k \in \text{cau}(Q'' \mid R) \). Since \( P'' \equiv Q'' \mid R \) it follows that \( k \in \text{cau}(P'', l) \) as required.

B. Transitions in different subprocesses: Assume without loss of generality that \( Q \xrightarrow{\mu[k]} Q' \) and \( R \xrightarrow{\nu[l]} R' \). These transitions are concurrent. By rule rev \text{par} \( Q \mid R \xrightarrow{\mu[k]} Q' \mid R \xrightarrow{\nu[l]} Q' \mid R' \) and \( Q \mid R \xrightarrow{\nu[l]} Q \mid R' \xrightarrow{\mu[k]} Q' \mid R' \) are valid. These form the diamond required for concurrent transitions with \( M \equiv Q' \mid R' \).

ii. \( P \xrightarrow{\mu[k]} P' \) by rule rev \text{com} and \( P \xrightarrow{\nu[l]} P'' \) by rule \text{par}: Without loss of generality this covers all cases with one \text{par} and one rev \text{com} or one rev \text{par} and one \text{com} transition. We assume that \( P \xrightarrow{\nu[l]} P' \)
happens by rule \( \text{rev com} \), where \( \gamma(\mu_1, \mu_2) = \mu \), and that \( P \xrightarrow{\nu[l]} P'' \) happens by rule \( \text{par} \). We also assume that \( \nu \) happens in \( Q \), so that \( Q \xrightarrow{\nu[l]} Q' \), and that \( Q \xrightarrow{\mu_1[k]} Q'' \) and \( R \xrightarrow{\mu_2[k]} R' \). We know that \( \text{fish}[l](R) \) because of the preconditions of the \( \text{par} \) rule and that \( l \neq k \) because \( \text{fish}[l](R) \) and \( R \xrightarrow{\mu_2[k]} R'' \), a transition which could not happen if \( l = k \), since according to Proposition 1.2 a key cannot be fresh for a reverse transition to happen with this key. By the inductive hypothesis transition \( Q \xrightarrow{\nu[l]} Q' \) and \( Q \xrightarrow{\mu_1[k]} Q'' \) are either concurrent or \( k \in \text{cau}(Q', l) \).

A. concurrent transitions: By the inductive hypothesis there is an \( N \) so that \( Q' \xrightarrow{\mu_1[k]} N \) and \( Q'' \xrightarrow{\nu[l]} N \). Using the \( \text{rev com} \) rule we can deduce that \( P \xrightarrow{\mu[k]} Q'' \mid R', \ P \xrightarrow{\nu[l]} Q' \mid R, \ Q'' \mid R' \xrightarrow{\nu[l]} N \mid R' \) and \( Q' \mid R' \xrightarrow{\nu[l]} N \mid R' \). Letting \( M \equiv N \mid R' \) we obtain the result.

B. \( k \in \text{cau}(Q', l) \): \( k \in \text{cau}(Q', l) \) implies \( k \in (\text{cau}(Q', l) \cup \text{cau}(R, l)) \) according to Definition 7 which is \( k \in \text{cau}(Q' \mid R, l) \). Since \( P'' = Q' \mid R \) we have \( k \in \text{cau}(P'', l) \) as required.

iii. \( P \xrightarrow{\mu[k]} P' \) by \( \text{rev com} \) and \( P \xrightarrow{\nu[l]} P'' \) by \( \text{com} \): Without loss of generality this covers all cases with one \( \text{com} \) and one \( \text{rev com} \) transition. We assume that \( \gamma(\mu_1, \mu_2) = \mu \) and \( \gamma(\nu_1, \nu_2) = \nu \).

Also \( Q \xrightarrow{\mu_1[k]} Q', \ Q \xrightarrow{\nu_1[l]} Q'' \), \( R \xrightarrow{\mu_2[k]} R' \) and \( R \xrightarrow{\nu_2[l]} R'' \). By the inductive hypothesis the transitions \( \mu_1 \) and \( \nu_1 \) must be either concurrent or \( k \in \text{cau}(Q'', l) \). For transitions \( \mu_2 \) and \( \nu_2 \) the same holds. So we distinguish three cases:

A. two concurrent transitions: Since \( Q \xrightarrow{\mu_1[k]} Q' \) and \( Q \xrightarrow{\nu_1[l]} Q'' \) by the inductive hypothesis it follows that there is an \( N \) so that \( Q' \xrightarrow{\nu_1[l]} N, \ Q'' \xrightarrow{\mu_1[k]} N \) and \( R \xrightarrow{\mu_2[k]} R' \) and \( R \xrightarrow{\nu_2[l]} R'' \) there is an \( N' \) so that \( R' \xrightarrow{\nu_2[l]} N' \) and \( R'' \xrightarrow{\mu_2[k]} N' \). By \( \text{rev par} \) and \( \text{par} \) it follows that \( P \xrightarrow{\mu[k]} Q' \mid R' \xrightarrow{\nu[l]} N \mid N' \) and \( P \xrightarrow{\nu[l]} Q'' \mid R'' \xrightarrow{\mu[k]} N \mid N' \). Letting \( M \equiv N \mid N' \) gives the result.

B. \( k \in \text{cau}(Q'', l) \) and \( k \in \text{cau}(R'', l) \): Since \( P'' = Q'' \mid R'' \) we can calculate \( \text{cau}(P'', l) = \text{cau}(Q'' \mid R'', l) = \text{cau}(Q'', l) \cup \text{cau}(R'', l) \). Since \( k \in \text{cau}(Q'', l) \) and \( k \in \text{cau}(R'', l) \) it follows that \( k \in \text{cau}(Q'', l) \cup \text{cau}(R'', l) \) and that \( k \in \text{cau}(P'', l) \) as required.

C. \( k \in \text{cau}(Q'', l) \), and \( R \xrightarrow{\mu_2[k]} R' \) and \( R \xrightarrow{\nu_2[l]} R'' \) are concurrent: Since \( P'' = Q'' \mid R'' \) we get \( \text{cau}(P'', l) = \text{cau}(Q'' \mid R'', l) = \text{cau}(Q'', l) \cup \text{cau}(R'', l) \). Since \( k \in \text{cau}(Q'', l) \) it follows that
\[ k \in \text{cau}(Q^n, l) \cup \text{cau}(R^n, l) \] and that \( k \in \text{cau}(P^n, l) \) as required. This also applies when \( k \in \text{cau}(R^n, l) \), so \( Q \mu[k] \rightarrow Q' \) and \( Q \nu[l] \rightarrow Q'' \) are concurrent.

(c) \( P \equiv R \setminus L \): The transitions \( R \setminus L \xrightarrow{\mu[k]} R' \setminus L \) and \( R \setminus L \xrightarrow{\nu[l]} R'' \setminus L \) are by rule rev res in Figure 1 respectively res in Figure 1. We assume without loss of generality that \( R \xrightarrow{\mu[k]} R' \), \( R \xrightarrow{\nu[l]} R'' \) and \( \mu, \nu \notin L \).

By the inductive hypothesis \( R \xrightarrow{\mu[k]} R' \) and \( R \xrightarrow{\nu[l]} R'' \) are either concurrent or \( k \in \text{cau}(R^n, l) \).

i. concurrent transitions: By the inductive hypothesis there is an \( N \) so that \( R' \xrightarrow{\mu[l]} N \) and \( R'' \xrightarrow{\nu[k]} N \). Consider \( M \equiv N \setminus L \).

Since \( \mu, \nu \notin L \) by rule rev res respectively res we deduce \( P \xrightarrow{\mu[k]} R' \setminus L \xrightarrow{\mu[l]} M \) and \( P \xrightarrow{\nu[l]} R'' \setminus L \xrightarrow{\nu[k]} M \).

ii. \( k \in \text{cau}(R^n, l) \): Since \( \text{cau}(P \setminus L, k) = \text{cau}(P, k) \) according to Definition 7 we calculate \( \text{cau}(R^n \setminus L, l) = \text{cau}(R^n, l) \). If \( k \in \text{cau}(R^n, l) \) it follows that \( k \in \text{cau}(R^n \setminus L, l) \) as required.

(f) \( P \equiv S \) with \( S \overset{\text{def}}{=} R \): similar to the \( P \equiv R \setminus L \) case.

**Proposition 7 (Rearrangement).** If \( \sigma \) is a trace then there exist forward traces \( \sigma_1 \) and \( \sigma_2 \) such that \( \sigma \preceq \sigma_1; \sigma_2 \).

**Proof.** We give a constructive proof. We show that any trace can be transformed to the form required by Proposition 7. Any trace must be either \( \sigma \), \( \sigma^* \) or \( \sigma_1; \sigma_2 \) or it must be of the form \( \sigma_1^*; \sigma_2^*; t_1; t_2^*; \sigma_3 \) where \( \sigma \), \( \sigma_1 \) and \( \sigma_2 \) are forward traces and \( \sigma_3 \) is composed of any number of forward and reverse transitions. In other words this means that we can identify the earliest pair of forward-reverse transitions. The instructions for the transformation to the required form are in the algorithm in Figure 10.

We show that the algorithm terminates in all cases with the required result. Executing the inner while loop (lines 9 to 13) once decreases the length of the resulting \( \sigma_2 \) by 1 and increases the length of the resulting \( \sigma_3 \) by 1.

After the inner while loop terminates we have \( \text{length}(\sigma_2^*) = \text{length}(\sigma_2) \).

The trace \( \sigma_1^*; t^* \) forms a new reverse only trace whose length is increased by 1 compared to \( \sigma_1^* \).

Executing the outer while loop once decreases the length of the sequence \( t_2^*; \sigma_3 \), which is the “unprocessed” part of \( \sigma_1^*; \sigma_2^*; t_1; t_2^*; \sigma_3 \), and changes the structure of \( \sigma_{\text{input}} \) so it is no longer as required by Proposition 7. The sequence \( \sigma_1^*; \sigma_2; t_1 \) is of the form required by Proposition 7 and its length is increased by the outer while loop. This is because at the end of the outer while loop we have \( \sigma_1^*; t^*; \sigma_2^*; \sigma_3 \). The part \( \sigma_1^*; t^*; \sigma_2^* \) of this trace is in the correct form and its length is increased by one, since \( t^* \) is added, \( \sigma_1^* \) is unchanged, and \( \text{length}(\sigma_2^*) = \text{length}(\sigma_2) \). The sequence \( \sigma_3 \) is the “unprocessed” part, which has been shortened by one transition, in comparison to \( t_2^*; \sigma_3 \) since \( \sigma_3 \) is unchanged.
Hence the algorithm terminates once \( \sigma_3 \) has been completely processed and the final \( \sigma_{\text{input}} \) has the required form with \( \sigma_3 \) being empty.

**Proposition 8 (Shortening).** If \( \sigma_1, \sigma_2 \) are coinitial and cofinal traces, with \( \sigma_2 \) forward, then there exists a forward trace \( \sigma'_1 \) of length at most that of \( \sigma_1 \) such that \( \sigma'_1 \succeq \sigma_2 \).

**Proof.** By induction on the length of \( \sigma_1 \). If \( \sigma_1 \) is a forward trace then the proposition holds. If not, then by Proposition 7 we assume \( \sigma_1 \) to be \( \sigma' \sigma' \) for some forward sequences \( \sigma \) and \( \sigma' \). There is only one sub-trace \( t_1; t_2 \) in \( \sigma'' \sigma' \) where the first transition \( t_1 \) is reverse and the second transition \( t_2 \) is forward. We assume \( t_1^* \equiv P \xrightarrow{\mu[k]} P' \) and \( t_2 \equiv P' \xrightarrow{\nu[l]} P'' \). There is a transition \( t' \equiv R \xrightarrow{\mu[k]} R' \) in \( \sigma_1 \) for some \( R, R' \), otherwise \( \sigma_1 \) could not be cofinal with \( \sigma_2 \).

Proposition 3 implies that there is a transition \( t_1 \equiv P' \xrightarrow{\mu[k]} P \). Transitions \( t_1 \) and \( t_2 \) are either concurrent, in conflict, or \( k \in \text{cau}(P, k) \) or \( k \in \text{cau}(P'', l) \). The possibility of \( t_1 \) and \( t_2 \) being in conflict is excluded since we have \( t_1 \) and \( t_2 \) in a valid trace, namely \( P' \xrightarrow{\mu[k]} P'' \xrightarrow{\nu[l]} R \xrightarrow{\mu[k]} R' \). Also \( P' \xrightarrow{\mu[k]} P' \) is impossible since we perform \( \nu[l] \) before \( \mu[k] \) in the trace \( P \xrightarrow{\mu[k]} P' \xrightarrow{\nu[l]} P'' \xrightarrow{\nu[l]} R \xrightarrow{\mu[k]} R' \).

Finally, \( l \in \text{cau}(P, k) \) cannot hold because otherwise, since \( P' \xrightarrow{\mu[k]} P \), some \( \alpha[l] \) transition must have happened in \( P' \) before \( P' \xrightarrow{\mu[k]} P \). This contradicts \( P' \xrightarrow{\nu[l]} P'' \) (key \( l \) is not fresh in \( P' \) due to the \( \alpha[l] \) action, hence \( P' \xrightarrow{\nu[l]} P'' \) is not possible for any \( P'' \)). Hence \( l \notin \text{cau}(P, k) \) and, overall, the only possibility is that \( t_1 \) and \( t_2 \) are concurrent.

The transitions \( t_1^* \) and \( t_2 \) form a diamond with two transitions \( t_3 \equiv P \xrightarrow{\nu[l]} P''' \) and \( t_4 \equiv P''' \xrightarrow{\mu[k]} P'' \) for some \( P''' \) according to Proposition 3. The trace \( t_3; t_4 \) can replace \( t_1; t_2 \) in \( \sigma'' \sigma' \) since the traces are coinitial and cofinal. We can repeat this process of “moving” an equivalent version of \( t_1^* \) to the right (by following the steps described above) until the resulting \( \sigma^* \) (with the label \( \mu[k] \)) is directly to the left of \( t' \equiv R \xrightarrow{\mu[k]} R' \). These transitions are then \( Q \xrightarrow{\mu[k]} R \xrightarrow{\mu[k]} R' \) for some \( Q \) (where \( Q = R \)). Using Definition 8 we can remove them. The resulting trace is shorter than \( \sigma_1 \) and we can repeat the process until the trace is forwards only.
Let $\sigma_{\text{input}}$ be our trace

while ($\sigma_{\text{input}} \neq \sigma \land \sigma_{\text{input}} \neq \sigma^* \land \sigma_{\text{input}} \neq \sigma_1^\bullet ; \sigma_2$ for all forward traces $\sigma, \sigma_1, \sigma_2$)

$\sigma_{\text{input}}$ is of the form $\sigma_1^\bullet ; \sigma_2 ; t_1 ; t_2^* ; \sigma_3$ for some (possibly new) $\sigma_1, \sigma_2, t_1, t_2, \sigma_3$, where $\sigma_1, \sigma_2, \sigma_3$ are forward traces, $t_1, t_2$ are forward transitions (any of $\sigma_1, \sigma_2, \sigma_3$ could be empty sequences $\epsilon$)

Let $\text{length}(\sigma_1) = n$, $\text{length}(\sigma_2) = k$, $\text{length}(\sigma_3) = l$. Distance from start to the pair $t_1 ; t_2^*$ is $n + k$

Let $t_1 \equiv P \xrightarrow{\mu[m]} P', t_2^* \equiv P' \xrightarrow{\nu[n]} P''$ (so $t_2 \equiv P'' \xrightarrow{\nu[n]} P'$)

if ($\mu[m] = \nu[n]$) then

Since $\mu[m] = \nu[k]$ we get $P \equiv P''$ and, by Definition 8, transitions $t_1 ; t_2^*$ are replaced by $\epsilon$ in $\sigma_{\text{input}}$

$\sigma_{\text{input}}$ is now $\sigma_1^\bullet ; \sigma_2 ; \sigma_3$

else

while ($\sigma_2 \neq \epsilon$)

$\sigma_{\text{input}}$ is $\sigma_1^\bullet ; \sigma_2 ; t_1 ; t_2^* ; \sigma_3$ for some (possibly new) $\sigma_2, t_1, t_2$ and $\sigma_3$, with $\sigma_1$ and $\sigma_2$ being forward traces and $t_1$ and $t_2$ being forward transitions

According to Proposition 13 there must be a transition $t_1^* \equiv P' \xrightarrow{\mu[m]} P$. $t_1^*$ and $t_2^*$ form a diamond with two transitions $t_3^* \equiv P \xrightarrow{\mu[m]} M$ and $t_4^* \equiv P'' \xrightarrow{\mu[m]} M$ for some $M$ according to Proposition 2. According to Proposition 13 there must be a transition $t_4 \equiv M \xrightarrow{\mu[m]} P''$. The trace $t_3^* ; t_4$ replaces $t_1 ; t_2^*$ in $\sigma_{\text{input}}$ since the traces are coinitial and cofinal

$\sigma_{\text{input}}$ is now $\sigma_1^\bullet ; \sigma_2 ; t_3^* ; t_4 ; \sigma_3$

end while (at this point $\sigma_2$ is $\epsilon$)

$\sigma_{\text{input}}$ is now $\sigma_1^\bullet ; t_4^* ; \sigma_3$ for some $t, \sigma_2'$ where $\sigma_2'$ is forwards only

end if

end while (at this point $\sigma_{\text{input}} = \sigma \lor \sigma_{\text{input}} = \sigma^* \lor \sigma_{\text{input}} = \sigma_1^\bullet ; \sigma_2$, for some forward traces $\sigma, \sigma_1$ and $\sigma_2$)

Figure 16: Rearrangement algorithm.