Modelling of DNA Mismatch Repair with a reversible process calculus

Stefan Kuhn

School of Computer Science and Informatics, De Montfort University, Leicester, UK

Irek Ulidowski

School of Computing and Mathematical Sciences, University of Leicester, Leicester, UK

Abstract

We have demonstrated in previous work that the Calculus of Covalent Bonding (CCB) can be used to simulate higher-level biochemical processes. This is significant since CCB was originally devised to model lower level organic chemical reactions. In this paper we extend the use of the calculus to model an important gene repair pathway, namely DNA Mismatch Repair (MMR). This complex pathway involves four helper proteins and needs a distinction between the two chains in a DNA strand. In order to achieve this, we extend the calculus by allowing prefixing with collections of bonding sites.

Keywords: Reversible computation, Calculus of Covalent Bonding, DNA Mismatch Repair.

1. Introduction

We have previously introduced a Calculus of Covalent Bonding (CCB) [1, 2] to models the formation and breaking of covalent bonds. CCB contains the possibility to link forming and breaking of bonds, thus enabling integrated, local control of reversibility. This feature allows us to represent both standard reversibility as well as *out-of-causal order* reversibility [3]. In order to demon-

Email addresses: stefan.kuhn@dmu.ac.uk (Stefan Kuhn), iu3@leicester.ac.uk (Irek Ulidowski)

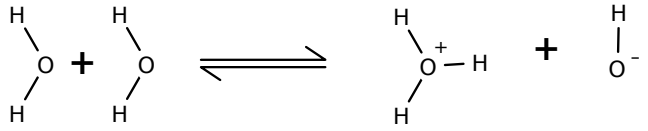

Figure 1: Autoprotolysis of water.

strate the basics of our calculus we show how to model the autoprotlysis of water, which transfers a proton between two water molecules. This reversible reaction is shown in Figure 1.

The main factor in the reaction is that oxygen is *nucleophilic*, so it attracts electrons. This makes the hydrogen atoms in water have a positive partial charge and the oxygen atoms a negative partial charge, attracting each other. Due to the nucleophilicity of oxygen, a covalent bond can form between them. This bond is formed out of electrons in the outer shell of the oxygen atom not used in bonds so far. Since a hydrogen atom cannot have more than one bond the creation of a new bond is compensated by breaking of the existing hydrogen-oxygen bond. These reactions are *concerted*, namely they happen together without a stable intermediate configuration. As a result we have reached the state where one oxygen has three covalent bonds to hydrogens and is positively charged, and the other oxygen bonds to only one hydrogen and is negatively charged. The reaction is reversible: the oxygen, which has lost a hydrogen, can pull back one of the hydrogens from the other water.

We now outline the modelling of the reaction in CCB. We model the hydrogen and oxygen atoms as processes H and O below, where h, o are actions representing the bonding capabilities of the atoms and n, p represent negative and positive charges respectively. H', O' are process constants.

$$\begin{aligned} H &\stackrel{def}{=} (h;p).H' \\ O &\stackrel{def}{=} (o,o,n).O' \end{aligned}$$

We use a general prefixing construct $(s;b).P$ where s is a sequence of actions or executed actions, and b is a *weak* action. Sometime the weak action is omitted (as in the definition of O). Informally, an action in s can take place in any

order, and b can happen if all actions in s have already taken place. Once b takes place, it must be accompanied by undoing immediately one of the actions in s (and thus possibly breaking a bond). We shall later extend this prefixing operator to permit multiple bonding sites as in $((s;b) \wr \cdots \wr (s';b')).P$.

We use a synchronisation function γ which tells us which actions can combine to produce bonds between atoms:

$$\gamma(h,o) = ho \qquad \gamma(n,p) = np \qquad \gamma(n,h) = nh \qquad \gamma(n,o) = no$$

Each water molecule is a structure consisting of two hydrogen atoms and one oxygen atom which are bonded appropriately.

$$((h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid (o_1[1],o_2[2],n).O_1') \setminus \{h_1,h_2,o_1,o_2\}$$

We have used subscripts to name the individual copies of atoms and actions. The system of two water molecules in Figure 1 is represented by placing them in parallel and restricting actions n and p. This is represented by the following process, where restrictions are moved outside using structural congruence laws:

$$\begin{aligned}&((h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid (o_1[1],o_2[2],n).O_1' \mid \\ &\mid (h_3[3];p).H_3' \mid (h_4[4];p).H_4' \mid (o_3[3],o_4[4],n).O_2') \setminus \{h_3,h_4,o_3,o_4,h_1,h_2,o_1,o_2,n,p\}\end{aligned}$$

Now actions n and p of different water molecules can combine, representing a transfer of a proton from one atom of oxygen to another oxygen. We show below the transfer from O_2 to O_1, using the *concerted actions* feature of CCB, where the actions to bond are indicated in bold blue and the bond to be broken is in bold red.

$$\begin{aligned}&((h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid (o_1[1],o_2[2],\boldsymbol{n}).O_1' \mid (\boldsymbol{h_3[3]};\boldsymbol{p}).H_3' \\ &\mid (h_4[4];p).H_4' \mid (\boldsymbol{o_3[3]},o_4[4],n).O_2') \setminus \{h_3,h_4,o_3,o_4,h_1,h_2,o_1,o_2,n,p\} \\ &\xrightarrow{\{np[5],\underline{h_3o_3}[3]\}} \\ &((h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid (o_1[1],o_2[2],\boldsymbol{n[5]}).O_1' \mid (\boldsymbol{h_3};\boldsymbol{p[5]}).H_3' \\ &\mid (h_4[4];p).H_4' \mid (\boldsymbol{o_3},o_4[4],n).O_2') \setminus \{h_3,h_4,o_3,o_4,h_1,h_2,o_1,o_2,n,p\}\end{aligned}$$

Since H_3 is weakly bonded to O_1 and its strong capability h_3 has become available, the bond 5 gets promoted to a stronger bond, releasing the capability p of H_3. This is done by the application of CCB's promotion rule:

$$\Rightarrow ((h_1[1];p).H_1' \mid (h_2[2];p).H_2' \mid (o_1[1],o_2[2],n[5]).O_1' \mid (h_3[5];p).H_3' \\ \mid (h_4[4];p).H_4' \mid (o_3,o_4[4],n).O_2') \setminus \{h_3,h_4,o_3,o_4,h_1,h_2,o_1,o_2,n,p\}$$

We have now arrived at the state on the right hand side in Figure 1. Oxygen O_1 is blocked, which represents it being fully bonded (and positively charged). Oxygen O_2 has a free n capability and can abstract any of the hydrogens from O_1. As a result the process can reverse to its original state or to equivalent states where different hydrogen atoms are bonded to O_1 and O_2.

In this paper, we model DNA mismatch repair. Since this requires modelling complex molecules as opposed to atoms, we extend the prefix operator of CCB to have multiple bonding sites. We also extend operational semantics of concerted actions which allows us to cover a fuller range of covalent reactions involving concerted actions. This builds upon our previous work on the modelling of BER and other biochemical reactions [4], and shows that the mechanisms in CCB are relevant outside its original application area.

Deoxyribonucleic acid (DNA) carries the essential information for the working of all known organisms. It is composed of two polynucleotide chains. Each nucleotide contains one of the four nucleobases, or simply bases, adenine (A), cytosine (C), guanine (G), and thymine (T). The bases can react with each other as follows: A and T produce a pair A-T, C and G produce a pair C-G, and other combinations are not normally possible. This property means that each of the two polynucleotide chains of a DNA contains the same biological information, but expressed differently. It also enables DNA replication. DNA can split into two chains and then each chain can be completed with the matching bases, so that each new DNA is an identical copy of the original strand. Replication in turn enables the multiplication of cells and growth.

In reality DNA can be imperfect. This could be due either to external factors, such as radiation or internal factors, such as free radicals produced by the body.

Another source of DNA errors are problems during replication. For example, it can happen that adenine in one chain faces guanine in the other chain when it should normally be matched by a thymine. Figure 2 shows such a mismatch on the right: The combination of the A/G bases is incorrect. Another error is an incorporation of a uracil base (U) in a DNA chain, which would normally not occur in DNA, is shown on the left in Figure 2. A repair mechanism for this error, called Base Excision Repair (BER), was modelled in [4] using our Calculus of Covalent Bonding (CCB).

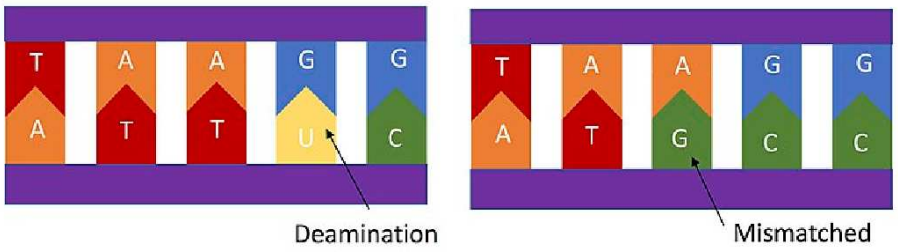

Figure 2: Two types of DNA damage. The right half shows a DNA mismatch. On the left, the incorporation of a Uracil base is shown. Part of Image by Chantelmao, licensed under CC-BY-SA-4.0.

DNA repair is crucial to maintain the quality of information in organisms. Whilst DNA replication errors are rare (they happen at a rate of about 1 per every 100,000 nucleotides), this translates to a large number of errors given the length of DNA: about 120,000 mistakes every time a human cell divides [5]. There are other sources of DNA damage, and also a multitude of repair mechanisms. We focus in this paper on a particular type of replication error called DNA Mismatch Repair (MMR) and its repair. Since incorrect DNA can give rise to cancer and other illnesses and generally to early ageing and death, DNA repair is a crucial part of life and its understanding is an important research topic. We shall model around 30 molecules involved in the MMR system, and represent the reactions of the mechanism as transitions of the MMR system. On the one hand, this is a continuation of the biochemical modelling in our previous work and, on the other hand, it provides an opportunity to further test

and evaluate CCB. A significant advantage of CCB is that it allows to represent out-of-causal order reversibility [3], where causes are seemingly undone before their effects are undone, which is common in biochemical reactions.

2. Related Work

This work has its origin in two strands of research. One is the simulation of chemical reactions and biological processes. The other is research on process calculi to model computation. These have been merged to better understand both areas of research.

In chemistry, speed of reactions and reaction rates have been modelled, with ordinary differential equations (ODEs) being an efficient tool [6]. Whilst such methods accurately show the dynamic behaviour, it was soon noted that they do not deal with the objects involved. In particular, any properties of the objects, which make reactions possible, are not considered. It became even more important when biochemical processes, which do not only involve small molecules, but also macromolecules, cells, and membranes, were modelled. This lack of understanding of the objects was raised in [7] and the use of computer science methods like the λ-calculus was suggested.

A connection between process calculi and chemistry has been made in [8]. Here, the Chemical Abstract Machine (CHAM) was introduced as a method to define the semantics of a calculus. As opposed to other semantics for process calculi, the algebraic terms ("molecules") can interact with each other no matter what order they are written in - this is similar to molecules being in a solution. CHAM also introduced membranes to group atoms and rules which simulate heating and cooling of solutions. The actual SOS and rewrite rules determining the reactions must be defined for a particular CHAM. Therefore, a CHAM can be used to execute processes of arbitrary process calculi. A CHAM can also model actual chemistry depending on the rules used. Starting with Regev et al. [9], process calculi, specifically the π-calculus, were used to model biochemical systems. There is an analogy between concurrent processes and

biological entities in natural systems, and between communication in computer systems and interactions between entities in biological systems. The modelling of biochemical systems is done by representing biological units as processes, and events of binding and unbinding as establishing and breaking a communication respectively. The stochastic π-calculus [10] was used to extend the model to include reaction rates [11]. This enables the simulation of the concentrations and of the overall state of the system over time.

Other work in the area include Bio-PEPA [12], which combines a process algebra with timing information and can represent reaction rates. In a Bio-PEPA simulation the entities modelled are species. Every interaction of molecules creates a new species, and there is no tracking of different "original" components. The transitions between the entities model the transformations between species. There is no tracking of mechanisms or modelling of the underlying causes. The focus is on the rates and the development of concentrations over time.

P Systems [13] was the first attempt to model membranes and compartments, with objects being able to move in and out of them. A P System contains several membranes, which can be inside one another. For every membrane a set of rules is defined, making P Systems a rule-based system. Objects are located inside membranes, can enter and leave membranes, and membranes can be dissolved. In [14] the usage of P Systems for modelling of distributed systems was demonstrated. [15] shows that even P systems with minimal parallelism can be universal. An example of a biological application is [16], modelling the sodium-potassium pump.

BioAmbients [17] is based on the Ambient calculus [18], which was originally devised for mobile computing outside the biological context and is an extension of the π-calculus. Here, similarly to P Systems, processes can be inside compartments. Brane Calculi ("brane" is short for membrane) [19], a family of process calculi, model membranes not only as compartments that define which interactions can happen, but as entities that can be transformed to gain new capabilities. Typical examples are viruses entering cells by interacting with their membranes. The Projective Brane Calculus [20] is a variant where interactions

are directed outward or inward from a membrane. Other formalims include the Language for Biochemical Systems (LBS) [21], the Calculus of Chemical Systems [22], and the Biochemical Abstract Machine (BIOCHAM) [23].

Biochemical reactions are in many cases reversible under certain conditions. The formalisms mentioned so far do not explicitly model reversibility as undoing of previously performed actions. Instead they use forward actions, which represent undoing of other actions. The first attempt at the modelling of undoing of forward computation in a process calculus was RCCS [24], a reversible calculus based on Milner's CCS. Reversibility is achieved by adding memories to the processes, where a record of past computation is stored. Another reversible calculus based on CCS is CCSK, introduced in [25, 26]. It uses keys as a particular form of memory. An extension of CCSK is the Calculus of Covelent Bonding (CCB), which was introduced in [1] and fully defined in [2]. CCB enables *locally controlled reversibility* by linking forming and breaking of bonds in the syntax of the calculus. CCB has been applied to chemical and biological modelling [4, 27]. Reversibility for the π-calculus was addressed in [28]. The κ-calculus [29] is based on graph-rewriting. Here, entities (e.g. proteins) are defined as having several sites, which can potentially bind to sites of other entities. The rewrite rules tell that a link can be formed or broken if the sites in the involved entities are in certain states. Reversibility in rule-based systems is discussed in [30]. Petri nets have also been successful in the modelling of biochemical reactions [31]. However, traditional Petri nets do not represent reversibility explicitly. Reversibility has only been introduced in Petri nets recently, as can be seen in [32, 33, 34, 35, 36].

Simple chemical reactions are modelled using CCB, P Systems, and Reversing Petri Nets in [27]. An important feature of those formalisms is that they can represent the so-called *out-of-causal order* reversibility [3], where effects can be seemingly undone before their causes are undone. In contrast, most other formalisms for reversible computation, for example RCSS and CCSK, represent *causal-consistent* and *backtracking* reversibility [37, 38], where effects can only be undone after all the causes (if any) are undone first.

3. A Calculus of Covalent Bonding

The Calculus of Covalent Bonding, or CCB for short, was defined in [2]. We recall here the main concepts and definitions in order to make the paper self-contained. We note that the syntax of this version of CCB is extended include a prefix that has collections of bonding sites (not just one site). As a result we also extend operational semantics of concerted actions which now allows us to cover a fuller range of covalent reactions involving concerted actions. Also, an informal new 'shortcut' SOS rule for three concerted actions is considered. We first introduce some preliminary notions and notations.

Let $\mathcal{A}$ be the set of (forward) action labels, ranged over by a, b, c, d, e, f, h and j. We partition $\mathcal{A}$ into the set of *strong actions*, written as $\mathcal{SA}$, and the set of *weak actions*, written as $\mathcal{WA}$. Communication between pairs of actions model creating bonds and undoing such communications represents breaking bonds. Most bonds result from communication between strong actions, but some bonds also involve a weak action. When a bond involving a weak action is created it causes breaking a neighbouring bond on strong actions[1]. Reverse (or past) action labels are members of $\underline{\mathcal{A}}$, with typical members $\underline{a}, \underline{b}, \underline{c}, \underline{d}, \underline{e}, \underline{f}, \underline{h}$ and $\underline{j}$, and represent undoing of actions. The set $\mathcal{P}(\mathcal{A} \cup \underline{\mathcal{A}})$ is ranged over by L.

Let $\mathcal{K}$ be an infinite set of *communication keys* (or *keys* for short) [25, 26], ranged over by k, l, m, n. The Cartesian product $\mathcal{A} \times \mathcal{K}$, denoted by $\mathcal{AK}$, represents past actions, which are written as $a[k]$ for $a \in \mathcal{A}$ and $k \in \mathcal{K}$. Correspondingly, we have the set $\underline{\mathcal{AK}}$ that represents undoing of past actions. Letters α, β denote actions which are either from $\mathcal{A}$ or $\mathcal{AK}$. It will be useful to consider sequences of actions or past actions, namely the elements of $(\mathcal{A} \cup \mathcal{AK})^*$, 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 ϵ. We use "α, s" and "s, s'" to denote a concatenation of elements, which can be strings or single actions.

[1] Fuller explanation follows Definition 1 on page 18.

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

Molecules may have several bonding *sites* and several bonds can be created or dissolved at each bonding site. A site and its potential bonds are modelled via $(s; b)$, where s is a sequence of actions, where actions model bonds, and b is a weak (bond) action. The action b can be omitted, in which case the construct is written as (s). When a molecule has several sites $(s; b), \ldots, (s'; b')$ we shall write them as $(s; b) \wr \ldots \wr (s'; b')$ by using the symbol "$\wr$" to indicate that sites bond independently of each other and that the order in which sites appear is irrelevant. Such expressions are called *collections of sites*, or just collections, and will be used to define the prefix operator. More formally, a site σ is either $(s; b)$ or (s). A collection μ is either σ or a composition of finitely many sites $\sigma \wr \ldots \wr \sigma$, where the order in which sites appear is irrelevant, and ϵ is the empty collection of sites. A site $(s; b)$ or (s) is fully bonded if all actions in s are bonded. We shall denote collections where all sites are fully bonded as η.

We now define the Calculus of Covalent Bonding. The syntax of CCB is given below where P and Q are process terms:

$$P ::= S \;\mid\; \mu.P \;\mid\; P \mid Q \;\mid\; P \backslash L$$

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

We have a general prefixing operator $\mu.P$. This operator extends the prefixing operators in [37], [3], and [1, 2]. If each site of μ has only one action, then it is the multi-action prefix from [37]. If the collection μ has only one site, then it is the general prefixing operator from [1, 2]. If its only site has no weak action, then the prefixing is written as $(s).P$ and is called the *simple prefix*. The simple prefix is the prefixing operator in [3]. 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 [39]. We omit trailing $\mathbf{0}$s so, for example, $(s).\mathbf{0}$ is written as (s). An interesting 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, which are strong actions, have taken place. Performing b then forces undoing one of the past actions in s (by the concert rules in Figures 6-7).

$P \mid Q$ 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 γ. As in the calculus ACP [40], the communication function is a partial function $\gamma : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is commutative and associative. The function γ is used in the operational semantics to define when two processes can interact. The use of a synchronization function instead of a τ-style communication as in CCS was motivated in the original CCB by the need to allow bonding between many different atoms. This can easily be captured by designing the atoms and the synchronizations separately. Processes P and Q in $P \mid Q$ can also perform a pair of concerted actions, which is a novel feature of our calculus. We also have the ACP-like restriction operator $\backslash L$, where L is a set of labels (subset of $\mathcal{P}(\mathcal{A} \cup \underline{\mathcal{A}})$). 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 $f : \mathcal{A} \to \mathcal{A}$, which could be easily added.

The set of *process terms* is ranged over by P, Q and R and is denoted by Proc. In the setting of CCB these terms are called *processes*. A context $C[\ ]$ is a process term containing a *hole*, represented by $[\ ]$. Formally, contexts are defined by the following syntax:

$$C ::= [\ ] \ \mid \ \mu.C \ \mid \ P \mid C \ \mid \ C \mid P \ \mid \ C \backslash L$$

The term $C[Q]$ denotes the result of filling the hole in the context $C[\ ]$ with Q.

We define the semantics of our calculus by a labelled transition system, LTS for short, which is a structure $(St, Lab, \rightarrow : \subseteq St \times Lab \times St)$ with St the set of

$$\frac{}{\mathsf{std}(\mathbf{0})} \quad \frac{\mathsf{std}(P)}{\mathsf{std}(S)}\ S \stackrel{def}{=} P \qquad\qquad \frac{}{\mathsf{fsh}[m](\mathbf{0})} \quad \frac{\mathsf{fsh}[m](P)}{\mathsf{fsh}[m](S)}\ S \stackrel{def}{=} P$$

$$\frac{\mathsf{k}(s) = \emptyset \quad \mathsf{std}(P)}{\mathsf{std}((s;b).P)} \qquad\qquad \frac{m \notin \mathsf{k}(s) \quad \mathsf{fsh}[m](P)}{\mathsf{fsh}[m]((s;b).P)}$$

$$\frac{\mathsf{k}(s) = \emptyset \quad \mathsf{std}(\mu.P)}{\mathsf{std}(((s;b) \wr \mu).P)} \qquad\qquad \frac{m \notin \mathsf{k}(s) \quad \mathsf{fsh}[m](\mu.P)}{\mathsf{fsh}[m](((s;b) \wr \mu).P)}$$

$$\frac{m \notin \mathsf{k}(s) \quad m \neq n \quad \mathsf{fsh}[m](P)}{\mathsf{fsh}[m]((s;b[n]).P)}$$

$$\frac{m \notin \mathsf{k}(s) \quad m \neq n \quad \mathsf{fsh}[m](\mu.P)}{\mathsf{fsh}[m](((s;b[n]) \wr \mu).P)}$$

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

Figure 3: Predicates std and fsh.

states, *Lab* the set of action labels and $\rightarrow : \subseteq St \times Lab \times St$ the labelled transition relation. For CCB, the set of states St is the set Proc. The action labels are the forward actions $\mathcal{AK}$, the reverse actions $\underline{\mathcal{A}}\mathcal{K}$ and the *concerted actions* $\mathcal{AK} \times \underline{\mathcal{A}}\mathcal{K}$. We shall use ω to denote a typical action label. The labelled transition relation will be defined in terms of SOS rules (Figures 4–8) and rewrite rules (Figure 9), where the rules in Figures 4–5 are influenced by [26]. Note that sequences s and t in Figures 4–7 are members of $(\mathcal{A} \cup \mathcal{AK})^*$ and $\mathcal{AK}^*$ respectively.

Next, we recall 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 4–8. Transitions above the horizontal bar in r are called *premises*. The set of premises is written as $pre(r)$. The transition below the bar in r is the *conclusion* and is written as $con(r)$.

We use two predicates $\mathsf{std}(P) : \mathsf{Proc}$ and $\mathsf{fsh}[m](P) : \mathcal{K} \times \mathsf{Proc}$ in our SOS rules. They are defined in Figure 3, and they use two auxiliary functions $\mathsf{k}(s) : (\mathcal{A} \cup \mathcal{AK})^* \rightarrow \mathcal{P}(\mathcal{K})$ and $\mathsf{keys}(P) : \mathsf{Proc} \rightarrow \mathcal{P}(\mathcal{K})$. The function $\mathsf{k}(\_)$ is defined as follows: $\mathsf{k}(\epsilon) = \emptyset$, $\mathsf{k}(\alpha : s) = \{l\} \cup \mathsf{k}(s)$ if $\alpha = a[l]$, for $a \in \mathcal{A}$ and $l \in \mathcal{K}$,

$$\mathsf{s}\ \frac{\mathsf{fsh}[k](s,s')}{(s,a,s';b) \xrightarrow{a[k]}_s (s,a[k],s';b)} \qquad \mathsf{c}\ \frac{\sigma \xrightarrow{a[k]}_s \sigma' \quad \mathsf{fsh}[k](\mu)}{\sigma \wr \mu \xrightarrow{a[k]}_s \sigma' \wr \mu}$$

$$\mathsf{act1}\ \frac{\mathsf{std}(P) \quad \mu \xrightarrow{a[k]}_s \mu'}{\mu.P \xrightarrow{a[k]} \mu'.P} \qquad \mathsf{act2}\ \frac{P \xrightarrow{a[k]} P' \quad \mathsf{fsh}[k](\eta)}{\eta.P \xrightarrow{a[k]} \eta.P'}$$

$$\mathsf{par}\ \frac{P \xrightarrow{a[k]} P' \quad \mathsf{fsh}[k](Q)}{P \mid Q \xrightarrow{a[k]} P' \mid Q} \qquad \mathsf{com}\ \frac{P \xrightarrow{a[k]} P' \quad Q \xrightarrow{d[k]} Q'}{P \mid Q \xrightarrow{c[k]} P' \mid Q'}$$

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

Figure 4: Forward SOS rules. We assume $\gamma(a,d) = c$.

and $\mathsf{k}(\alpha : s) = \mathsf{k}(s)$ if $\alpha \in \mathcal{A}$. The function $\mathsf{keys}(\_)$ is given by $\mathsf{keys}(\mathbf{0}) = \emptyset$, $\mathsf{keys}(S) = \mathsf{keys}(P)$ if $S \stackrel{def}{=} P$ and S is guarded in P (correspondingly for $\mathsf{std}(S)$ and $\mathsf{fsh}[m](S)$ in Figure 3), $\mathsf{keys}(((s;\beta) \wr \mu).P) = \mathsf{k}(s) \cup \mathsf{k}(\beta) \cup \mathsf{keys}(\mu.P)$, $\mathsf{keys}(P \mid Q) = \mathsf{keys}(P) \cup \mathsf{keys}(Q)$, and $\mathsf{keys}(P\backslash L) = \mathsf{keys}(P)$. Informally $\mathsf{keys}(P)$ associates with each term P the set of its keys. A process P is *standard*, written $\mathsf{std}(P)$, if it contains no past actions (hence no keys). A key n is *fresh* in Q, written $\mathsf{fsh}[n](Q)$, 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, and to sites σ, via the function $\mathsf{k}(\_)$. Moreover, correspondingly, $\mathsf{fsh}[n](\mu)$ if $\mathsf{fsh}[n](\sigma)$ for every site σ in μ.

Example 1. We illustrate how processes compute forwards using the new prefixing operator. Initially, we consider processes where collections have single sites only. Consider a standard process $(a;b).(c) \mid (a,d,c)$ and the communication function γ given by $\gamma(a,a) = a$ and $\gamma(c,c) = c$. We have

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

by the SOS rules act1 and com from Figure 4. This is because (c) is standard and the key 1 is fresh in ε. The next step of computation involves a communication

$$\textsf{rev s}\ \frac{}{(s, a[k], s'; \beta) \xrightarrow{\underline{a}[k]}_s (s, a, s'; \beta)} \qquad \textsf{rev c}\ \frac{\sigma \xrightarrow{\underline{a}[k]}_s \sigma' \quad \textsf{fsh}[k](\mu)}{\sigma \wr \mu \xrightarrow{\underline{a}[k]}_s \sigma' \wr \mu}$$

$$\textsf{rev act1}\ \frac{\textsf{std}(P) \quad \mu \xrightarrow{\underline{a}[k]}_s \mu'}{\mu.P \xrightarrow{\underline{a}[k]} \mu'.P} \qquad \textsf{rev act2}\ \frac{P \xrightarrow{\underline{a}[k]} P'}{\eta.P \xrightarrow{\underline{a}[k]} \eta.P'}$$

$$\textsf{rev par}\ \frac{P \xrightarrow{\underline{a}[k]} P' \quad \textsf{fsh}[k](Q)}{P \mid Q \xrightarrow{\underline{a}[k]} P' \mid Q} \qquad \textsf{rev com}\ \frac{P \xrightarrow{\underline{a}[k]} P' \quad Q \xrightarrow{\underline{d}[k]} Q'}{P \mid Q \xrightarrow{\underline{c}[k]} P' \mid Q'}$$

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

Figure 5: Reverse SOS rules. We assume $\gamma(a, d) = c$, and and β is either b or $b[l]$ for $b \in \mathcal{WA}$.

of the actions c, which we obtain by rules act2 and com:

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

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

$$(a[1]; b).(c[2]) \mid (a[1], d, c[2]) \xrightarrow{d[3]} (a[1]; b).(c[2]) \mid (a[1], d[3], c[2])$$

Formally, we use par, the structural congruence rule sc in Figure 8 and the reduction rule red1 in Figure 9.

275 The next example illustrates how some of the reverse SOS rules work. We only consider here processes with collections that consist of single sites.

Example 2. Consider $(a[1], b).(c).S$ where $S \stackrel{def}{=} (a, b).(c).S$. We have

$$(a[1], b).(c).S \xrightarrow{\underline{a}[1]} (a, b).(c).S$$

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

$$\mathsf{w}\ \frac{\mathsf{fsh}[l](t)}{(t;b) \xrightarrow{(b)[l]}_s (t;b[l])} \qquad \mathsf{cw}\ \frac{\sigma \xrightarrow{(b)[l]}_s \sigma' \quad \mathsf{fsh}[l](\mu)}{\sigma \wr \mu \xrightarrow{(b)[l]}_s \sigma' \wr \mu}$$

$$\mathsf{aux1}\ \frac{\mathsf{std}(P) \quad \mu \xrightarrow{(b)[k]}_s \mu'}{\mu.P \xrightarrow{(b)[k]} \mu'.P} \qquad \mathsf{aux2}\ \frac{P \xrightarrow{(b)[k]} P' \quad \mathsf{fsh}[k](\eta)}{\eta.P \xrightarrow{(b)[k]} \eta.P'}$$

$$\mathsf{concert2'}\ \frac{P \xrightarrow{(b)[k]} P' \xrightarrow{\underline{a}[l]} P'' \qquad Q \xrightarrow{\alpha[k]} Q' \xrightarrow{\underline{d}[l]} Q''}{P \mid Q \xrightarrow{\{e[k],\underline{f}[l]\}} P'' \mid Q''}$$

$$\mathsf{concert2}\ \frac{U \equiv P \mid Q \quad P \xrightarrow{(b)[k]} P' \xrightarrow{\underline{a}[l]} P'' \quad Q \xrightarrow{\alpha[k]} Q' \quad R \xrightarrow{\underline{d}[l]} R'}{U \mid R \xrightarrow{\{e[k],\underline{f}[l]\}} P'' \mid Q' \mid R'}$$

Figure 6: SOS rules for two concerted actions transitions. We assume α is c or (c) and $\gamma(b,c) = e$ for some $c \in \mathcal{A}$, $\gamma(a,d) = f$, and $\gamma(g,h) = j$. The symbol '$\equiv$' denotes syntactic congruence. Additionally, we assume that actions $(b)[k]$ and $\underline{a}[l]$ in concert2' and concert2 occur in the same site of P and correspondingly for $(c)[k]$, $\underline{d}[l]$ and Q in concert2'.

Figures 4–5 contain the usual SOS rules that define single action transitions of CCB. An auxiliary transition relation $\rightarrow_s$ is used there to define operational changes to sites and collections of sites.

Figures 6–7 contain the SOS rules that define the concerted actions transitions. The main rules are the concert rules in Figure 6 that define when a pair of concerted actions take place. We also have there four auxiliary rules w, cw, aux1 and aux2 which define only an auxiliary action transition relation needed in the concert rules. Note that transitions in aux1 and aux2 use the auxiliary labels $(b)[k]$ for all $b \in \mathcal{WA}$ and $k \in \mathcal{K}$. Also note that the three concert rules use *lookahead* [41].

The rules concert par in Figure 7 require that k is fresh in Q, correspondingly as in par. Moreover, we need to ensure that when we reverse h with the key l in P we do not leave out any actions with the key l in Q which make up a multi-action communication with the key l. Hence, we also include the premise $\mathsf{fsh}[l](Q)$ in concert par rules. Correspondingly for the key m in Q. The two rules

$$\textsf{concert act}\ \frac{P \xrightarrow{\{a[k],\underline{b}[l]\}} P' \quad \textsf{fsh}[k](\eta)}{\eta.P \xrightarrow{\{a[k],\underline{b}[l]\}} \eta.P'}$$

$$\textsf{concert2 par}\ \frac{P \xrightarrow{\{a[k],\underline{b}[l]\}} P' \quad \textsf{fsh}[k](Q) \quad \textsf{fsh}[l](Q)}{P \mid Q \xrightarrow{\{a[k],\underline{b}[l]\}} P' \mid Q}$$

$$\textsf{concert2 res}\ \frac{P \xrightarrow{\{a[k],\underline{b}[l]\}} P'}{P\backslash L \xrightarrow{\{a[k],\underline{b}[l]\}} P'\backslash L}\ \ a, \underline{b} \notin L \cup (L)$$

Figure 7: SOS rules for the prefix, parallel composition and restriction and concerted transitions.

$$\frac{P \Rightarrow Q \quad Q \xrightarrow{\omega} Q' \quad Q' \Rightarrow P'}{P \xrightarrow{\omega} P'}$$

Figure 8: Structural congruence rule sc when $\omega \in \mathcal{AK} \cup (\mathcal{AK} \times \underline{\mathcal{A}}\mathcal{K})$ and rev sc when $\omega \in \underline{\mathcal{A}}\mathcal{K}$.

concert act require, correspondingly as act, that k is fresh in t. Our operational semantics guarantees that if a standard process evolves to $(t; b).P$, where all actions in t are fully executed, and P reverses an action with the key l, then l is fresh in t. Hence, we do not include $\textsf{fsh}[l](t)$ in the premises of the concert act rules.

We now illustrate how concerted action transitions work.

Example 3. Consider the process $(a; b) \mid a \mid d$ with $\gamma(a, a) = c$ and $\gamma(b, d) = f$. After the initial synchronisation of actions a, which produces the transition $c[1]$, we can bond weak b with strong d producing f, and at the same time break the c bond. This is represented by a transition with a pair of concerted actions:

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

The transition is derived by rule concert2 in Figure 6 since $(a[1]; b) \xrightarrow{(b)[2]} (a[1]; b[2]) \xrightarrow{\underline{a}[1]} (a; b[2])$ by aux1 and rev act1, and since $a[1] \mid d \xrightarrow{f[2]} a[1] \mid d[2] \xrightarrow{\underline{a}[1]} a \mid d[2]$ by par and rev par.

$$
\begin{array}{lll}
\mathsf{red1}: & P \mid Q \Rightarrow Q \mid P & \\
\mathsf{red2}: & P \mid (Q \mid R) \Rightarrow (P \mid Q) \mid R & \\
\mathsf{red3}: & (P \mid Q) \mid R \Rightarrow P \mid (Q \mid R) & \\
\mathsf{red4}: & P \mid \mathbf{0} \Rightarrow P & \\
\mathsf{red5}: & (P \mid Q)\backslash L \Rightarrow P\backslash L \mid Q & \text{if } (\gamma(\mathsf{fn}(P) \times \mathsf{fn}(Q)) \cup \mathsf{fn}(Q)) \cap L = \emptyset \\
\mathsf{red6}: & P\backslash L \mid Q \Rightarrow (P \mid Q)\backslash L & \text{if } (\gamma(\mathsf{fn}(P) \times \mathsf{fn}(Q)) \cup \mathsf{fn}(Q)) \cap L = \emptyset \\
\mathsf{prom}: & ((s, a, s'; b[k]) \wr \mu).P \Rightarrow ((s, a[k], s'; b) \wr \mu).P & \text{if } a \in \mathcal{SA}, b \in \mathcal{WA} \\
\mathsf{move}: & ((s, a, s', b[k], s'') \wr \mu).P \Rightarrow ((s, a[k], s', b, s'') \wr \mu).P & \text{if } a \in \mathcal{SA}, b \in \mathcal{WA}
\end{array}
$$

Figure 9: Reduction rules. Sequences s, s', s'' are members of $(\mathcal{A} \cup \mathcal{AK})^*$.

The next example illustrates a creation of a bond between weak actions.

Example 4. Consider $(a[1]; b) \mid (a[1]; b)$ with $\gamma(a, a) = c$ and $\gamma(b, b) = d$. Recall that actions b here are weak. We have the following pair of concerted actions derived by concert2', where the bond created and the bond broken are between the same two processes:

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

This transition cannot be derived by concert2 because we only have two processes here and concert2 requires at least three separate processes. Note that concert2' can be used to derive

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

Overall, the transitions in Figures 4–8 are labelled with $a[k] \in \mathcal{AK}$, or with $\underline{c}[l] \in \underline{\mathcal{A}}\mathcal{K}$, or with concerted actions $(a[k], \underline{c}[l])$.

We also have the usual structural congruence rules sc and rev sc in Figure 8, which potentially combine reductions (defined below) with transitions.

Next, we introduce our reduction relation which is given by the reduction (rewrite) rules in Figure 9. The reduction relation is needed to define *promotion*

of actions. First we define *free names* of processes. The function $\mathsf{fn} : \mathsf{Proc} \to \mathcal{P}(\mathcal{K})$ is given as follows: $\mathsf{fn}(\mathbf{0}) = \emptyset$, $\mathsf{fn}(S) = \mathsf{fn}(P)$ if $S \stackrel{def}{=} P$, $\mathsf{fn}((\sigma \wr \mu).P) = \mathsf{f}(\sigma) \cup \mathsf{fn}(\mu.P)$, where $\mathsf{f}(\sigma)$ is the set of all action labels in σ, $\mathsf{fn}(\varepsilon.P) = \mathsf{fn}(P)$, $\mathsf{fn}(P \mid Q) = \mathsf{fn}(P) \cup \mathsf{fn}(Q)$, and $\mathsf{fn}(P \backslash L) = \mathsf{fn}(P) \backslash L$.

The 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 1. 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 9.

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

The rewrite rules in Figure 9 include prom and move which promote weak bonds (here b) to strong bonds (here a). Recall that the order of actions in s in expressions for sites (s) and $(s; b)$ is irrelevant. The rule prom applies to the full version of site expression (with the ; construct), and move applies only to the simple site expressions without ';'. These two rules 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 two 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 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 $\mathcal{A}$, $\mathcal{A}\mathcal{K}$, and the concerted actions labels in $\mathcal{A}\mathcal{K} \times \underline{\mathcal{A}}\mathcal{K}$. First, the

transition relation on collections sites, $\rightarrow_s$, is defined as the smallest relation derived by the rules s, c, $\mathsf{rev\ s}$, $\mathsf{rev\ c}$, w, and cw in Figures 4–6. Second, the transition relation $\rightarrow$ is defined as the smallest relation derived by the rules in Figures 4–8. In order to define our transition relation $\mapsto$ we introduce processes in *normal forms*: $\mathsf{nf}(P)$ is the set of P' such that $P \Rightarrow P'$ and P' cannot be reduced via prom or move.

Definition 2. We associate to Proc and $\mathcal{AK} \cup \underline{\mathcal{A}}\mathcal{K} \cup (\mathcal{AK} \times \underline{\mathcal{A}}\mathcal{K})$ a transition relation $\mapsto$ given by $P \stackrel{\omega}{\mapsto} Q$ if $P \stackrel{\omega}{\rightarrow} P'$ for some P' and $Q \in \mathsf{nf}(P')$.

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

Example 5. The transition $(a[1]; b) \mid a[1] \mid b \xrightarrow{\{d[2], \underline{c}[1]\}} (a; b[2]) \mid a \mid b[2]$ from Example 3 cannot give rise to the corresponding $\mapsto$ transition which is followed by a communication of actions a. This is because there is a prom redex $(a; b[2])$ in $(a; b[2]) \mid a \mid b[2]$. The rewrite of this redex takes priority: the bond 2 moves from the weak b to the strong a by prom. Since $(a; b[2]) \mid a \mid b[2] \Rightarrow (a[2]; b) \mid a \mid b[2]$ and the contractum cannot be reduced with prom or with move, we have

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

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

There are situations where creation of a bond between weak actions of two processes requires breaking of bonds with other processes.

Example 6. Consider $(a[1]; b) \mid (e[2]; b) \mid (a[1], e[2])$ with $\gamma(a, a) = c$, $\gamma(b, b) = d$, and $\gamma(e, e) = h$. The process cannot perform any concerted actions by $\mathsf{concert2'}$: Although $(a[1]; b) \xrightarrow{(b)[l]} \xrightarrow{\underline{a}[1]} (a; b[l])$, for any l different from 1 and 2, but $(e[2]; b) \mid (a[1], e[2])$ cannot perform the 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 $\mathsf{concert2'}$, respectively, and we notice

that we cannot undo a communication on a or e. However, this is precisely what concert2 allows, so we have these two different concerted actions transitions:

$$(a[1];b) \mid (e[2];b) \mid (a[1],e[2]) \xrightarrow{\{d[3],\underline{c}[1]\}} (a;b[3]) \mid (e[2];b[3]) \mid (a,e[2])$$

$$(a[1];b) \mid (e[2];b) \mid (a[1],e[2]) \xrightarrow{\{d[3],\underline{h}[2]\}} (a[1];b[3]) \mid (e;b[3]) \mid (a[1],e)$$

Promotion can be applied in both cases giving us the transitions

$$(a[1];b) \mid (e[2];b) \mid (a[1],e[2]) \xmapsto{\{d[3],\underline{c}[1]\}} (a[3];b) \mid (e[2];b[3]) \mid (a,e[2])$$

$$(a[1];b) \mid (e[2];b) \mid (a[1],e[2]) \xmapsto{\{d[3],\underline{h}[2]\}} (a[1];b[3]) \mid (e[3];b) \mid (a[1],e)$$

We notice that, following the first transition above, the $h[2]$ bond can be broken, and correspondingly the $c[1]$ bond can be broken after the second transition. These together with applying promotion gives us the following:

$$(a[3];b) \mid (e[2];b[3]) \mid (a,e[2]) \xmapsto{\underline{h}[2]} (a[3];b) \mid (e[3];b) \mid (a,e)$$

$$(a[1];b[3]) \mid (e[3];b) \mid (a[1],e) \xmapsto{\underline{c}[1]} (a[3];b) \mid (e[3];b) \mid (a,e)$$

Creation of the $d[3]$ bonds makes the $h[2]$ and $c[1]$ bonds unstable, so they break immediately after the corresponding concerted actions. Since creation and breaking of such bonds is almost instantaneous, we shall use additional "shortcut" SOS rules to capture such cascade of two concerted actions and a reversal of an unstable action, called concert3. They are not formally part of our operational semantics but could be used in some circumstances to represent two specific transitions as one. It allows us to derive

$$(a[1];b) \mid (e[2];b) \mid (a[1],e[2]) \xrightarrow{\{d[3],\underline{c}[1],\underline{h}[2]\}} (a;b[3]) \mid (e;b[3]) \mid (a,e)$$

which, when followed by application of promotion, gives us

$$(a[1];b) \mid (e[2];b) \mid (a[1],e[2]) \xrightarrow{\{d[3],\underline{c}[1],\underline{h}[2]\}} (a;b[3]) \mid (e;b[3]) \mid (a,e)$$

Here, creation of the bond d results in two other bonds c, h being broken. The rule concert3 and the related rules are in Figure 10.

$$\textsf{concert3}\ \frac{U \equiv P \mid Q \quad P \xrightarrow{(b)[k]} P' \xrightarrow{\underline{a}[l]} P'' \quad Q \xrightarrow{\alpha[k]} Q' \xrightarrow{\underline{g}[m]} Q'' \quad R \xrightarrow{\underline{d}[l]} R' \xrightarrow{\underline{h}[m]} R''}{U \mid R \xrightarrow{\{e[k],\underline{f}[l]\},\underline{j}[m]\}} P'' \mid Q'' \mid R''}$$

$$\textsf{concert3 act}\ \frac{P \xrightarrow{\{a[k],\underline{h}[l],\underline{j}[m]\}} P' \quad \textsf{fsh}[k](\eta)}{\eta.P \xrightarrow{\{a[k],\underline{h}[l],\underline{j}[m]\}} \eta.P'}$$

$$\textsf{concert3 par}\ \frac{P \xrightarrow{\{a[k],\underline{h}[l],\underline{j}[m]\}} P' \quad \textsf{fsh}[k](Q) \quad \textsf{fsh}[l](Q) \quad \textsf{fsh}[m](Q)}{P \mid Q \xrightarrow{\{a[k],\underline{h}[l],\underline{j}[m]\}} P' \mid Q}$$

$$\textsf{concert3 res}\ \frac{P \xrightarrow{\{a[k],\underline{h}[l],\underline{j}[m]\}} P'}{P\backslash L \xrightarrow{\{a[k],\underline{h}[l],\underline{j}[m]\}} P'\backslash L}\ a, \underline{h}, \underline{j} \notin L \cup (L)$$

Figure 10: Shortcut SOS rules for the three concerted actions transitions. We assume α is c or (c) and $\gamma(b, c) = e$ for some $c \in \mathcal{A}$, $\gamma(a, d) = f$, and $\gamma(g, h) = j$.

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

3.1. Properties of CCB

The current version of CCB is a minor extension of the original CCB from [2] by extending the prefix operator to use collections of sites. If collections have only one site, then both calculi have the same syntax. We are using additionally shortcut rules for concerted transitions that involve three actions, namely concert3, concert3 act, concert3 par, concert3 res, but they are just a shorthand for sequences of two standard transitions.

We have enlarged slightly the class of processes for which we can derive concerted actions and for that reason we have used concert2' and concert2 rules. All the properties that the original CCB possesses are also enjoyed by the current calculus. In particular, we can show, using the same arguments as in [2], that

this CCB satisfies several classical properties of reversible computation such as, for example, *Well-foundedness*, *Reverse Diamond* and *Forward Diamond* [26, 38]. Moreover, if we leave out weak actions (and do not use SOS rules that involve weak actions), then the resulting calculus satisfies the *Causal Consistency* property [24, 38]. This gives confidence that the core of CCB models reversibility correctly.

The purpose of CCB is however to capture the so called "out-of-causal order" behaviour which is so common in biochemical reactions. This is achieved via our concerted actions transitions, which allow us faithful modelling of complex sequences of reactions such as those present in MMR.

4. DNA Mismatch Repair

4.1. Description of DNA Mismatch Repair

We now return to the DNA mismatch repair, which we have briefly described in the introduction, and we give a fuller explanation of how it works. If two bases are incorporated into a DNA strand which do not match, namely are not A-T or C-G pairs, this constitutes a defect in the DNA strand. A repair similar to BER is not possible, since it is not clear which of the two bases is correct. We are looking at a specific way to handle this situation, called Methyl Mismatch Repair (MMR). This is the repair mechanism in bacteria, in particular E.coli, for which detailed studies has been done[2]. The mechanism relies on DNA strands being methylated sometime after their creation. Methylation means the attachment a methyl group (a carbon atom with three hydrogen atoms), in this case to the adenine base A. Specifically, this happens whenever there is a GATC sequence. Since the methylation happens with a delay after the duplication of a strand, the old strand is methylated and the new strand is not methylated for a short period of time. This enables the removal of the wrong base from the new strand.

[2]Paul L. Modrich, Tomas Lindahl and Aziz Sancar received the Nobel Price in Chemistry 2015 for their work on DNA repair. Modrich's Nobel Lecture about Methyl-directed Mismatch Repair in E. coli and humans [42] gives a good overview.

The actual repair involves the proteins MutS, MutL, MutH, and UvrD. MutS first binds to the mismatch and then recruits MutL and MutH. MutL can detect the methylated strand and can form a loop in the DNA. In this loop, the newer strand is now outside, whereas the old strand is covered by itself and MutL. Then MutH cleaves the unmethylated strand. This happens in some distance from the mismatch, due to the size of the proteins and the loop in the DNA. UvrD can then detect the cleavage and can move along the strand towards the site of the mismatch. It removes the outer strand when moving along. At the same time, the MutL, MutH, and MutS proteins are released and the loop in the DNA disappears. Finally UvrD moves to a position immediately beyond the mismatch, removing the wrong base together with its neighbours. UvrD then goes off the DNA and leaves the old, and correct, strand to be completed.

4.2. DNA Mismatch Repair System

Having briefly introduced all major components that are involved in this mismatch repair mechanism, we now define them in the syntax of CCB. Our intention is to reuse the components from [4] as much as possible. Specifically, we reuse the deoxyribose/phosphate groups, and the four bases A, T, C, and G. We will extend those components where needed, but we will keep the original actions. We need the following components for the MMR system: deoxyribose/phosphate groups (DP), the four bases A, T, G, C, the methyl group Me, and the proteins MutS, MutL, MutH, and UvrD. The reused components are as follows:

$$
\begin{aligned}
DP &\stackrel{def}{=} ((p3, p5; s) \wr (b, d)).DP' \\
A &\stackrel{def}{=} (b; i).(a, m; r).A' \\
T &\stackrel{def}{=} (b; i).(t; r).T' \\
G &\stackrel{def}{=} (b; i).(g; r).G' \\
C &\stackrel{def}{=} (b; i).(c; r).C'
\end{aligned}
$$

Notice that we have added new actions s, m, and r. The action m enables the methylation. It only exists in A since the methylation happens only there. In DP, we now use a collection with two sites. This is necessary to enable breaking

of a bond to $p3$ or $p5$, which are on one site, whilst there is still a bond to b on the other site.

We also need additional components, namely the methyl group Me, and the MutS, MutL, MutH, and UvrD proteins. They are defined as follows:

$$\begin{array}{lcl} \mathit{Me} & \stackrel{def}{=} & (m).(n).\mathit{Me}' \\ \mathit{MutS} & \stackrel{def}{=} & (k,k).(l).\mathit{MutS}' \\ \mathit{MutL} & \stackrel{def}{=} & (l).(n).(o).\mathit{MutL}' \\ \mathit{MutH} & \stackrel{def}{=} & (o).(w).\mathit{MutH}' \\ \mathit{UvrD} & \stackrel{def}{=} & ((u;r) \wr (v;s)).\mathit{UvrD}' \end{array}$$

Here s, i, and r are weak actions, all other actions, namely $p3$, $p5$, b, d, a, t, g, c, m, n, k, l, o, s, u, v and w, are strong. Notice that i is not used here, but we keep it to have this model as an extension of the BER model [4]. In *UvrD*, we again use a prefix with two sites. This will make it possible for *UvrD* to break two bonds (one from each site) as it "moves" along a DNA strand.

The synchronisation function for our system is as follows:

$\gamma(p3,p5) = p$	$\gamma(b,b) = bb$	$\gamma(a,t) = at$	$\gamma(g,c) = gc$
$\gamma(m,m) = mm$	$\gamma(k,a) = ka$	$\gamma(k,g) = kg$	$\gamma(k,c) = kc$
$\gamma(k,t) = kt$	$\gamma(l,l) = ll$	$\gamma(n,n) = nn$	$\gamma(o,o) = oo$
$\gamma(r,r) = rr$	$\gamma(t,u) = tu$	$\gamma(p,s) = ps$	$\gamma(w,s) = ws$
$\gamma(s,p3) = sp3$	$\gamma(v,p3) = vp3$	$\gamma(w,p3) = wp3$	$\gamma(u,r) = ur$
$\gamma(s,s) = ss$	$\gamma(s,v) = sv$	$\gamma(u,c) = uc$	

The deoxyribose/phosphate groups and the bases can combine to form a DNA strand. There can be DNA mismatches in such strands. Note that we do not model how they happen, we just assume a DNA strand as in Figure 11 with one DNA mismatch. The mismatch here is on the A_2 and G_3 pair. The A and G bases cannot bond to each other–it is the case in our model as well as in reality–and this gives the opportunity for the repair. There is a CTAG sequence (respectively a GATC sequence) on the top left (respectively bottom left) chain of the DNA in Figure 11. This is where the methylation happens. In our case, we have a recently duplicated strand, where the older chain is the upper one

(A_1 is methylated there) and the newer chain is the lower one (where A is not methylated). The methylation is done by proteins which are not modelled here.

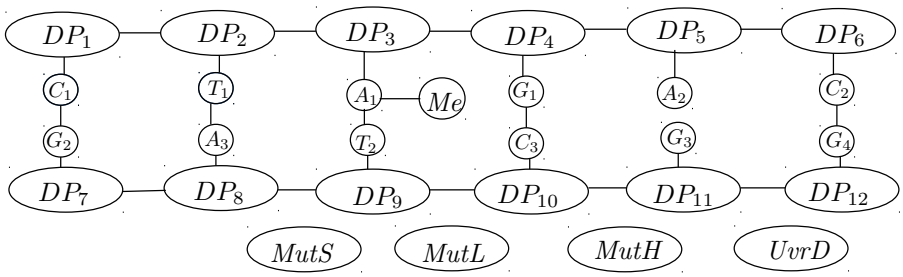

Figure 11: A six base pair DNA fragment, with a DNA mismatch and methylation of an upper strand. The proteins *MutL*, *MutS*, *MutH*, and *UvrD* are not currently bonded to the DNA.

The system shown in Figure 11 is modelled in CCB as follows:

$$
\begin{aligned}
&(DP_1 \mid DP_2 \mid DP_3 \mid DP_4 \mid DP_5 \mid DP_6 \mid \\
&C_1 \mid T_1 \mid A_1 \mid G_1 \mid A_2 \mid C_2 \mid \\
&G_2 \mid A_3 \mid T_2 \mid C_3 \mid G_3 \mid G_4 \mid \\
&DP_7 \mid DP_8 \mid DP_9 \mid DP_{10} \mid DP_{11} \mid DP_{12} \mid \\
&Me \mid MutS \mid MutL \mid MutH \mid UvrD) \\
&\backslash \{p3, p5, s, i, r, b, b, d, b, a, t, g, c, m, n, k, l, o, s, u, v, w\}
\end{aligned}
$$

We leave out the restriction from now on for ease of reading. We mark copies of processes DP and the bases, and their actions, using subscripts in order to make them unique. Pairs of keys show on which actions the initial bonds have been created. The MMR system in full detail is given below, where henceforth

we leave out the outermost restriction.

$((p3_1, p5_1[1]; s_1) \wr (b_1[10], d_1)).DP'_1 \mid ((p3_2[1], p5_2[2]; s_2) \wr (b_2[8], d_2)).DP'_2 \mid$
$((p3_3[2], p5_3[3]; s_3) \wr (b_3[6], d_3)).DP'_3 \mid ((p3_4[3], p5_4[4]; s_4) \wr (b_4[9], d_4)).DP'_4 \mid$
$((p3_5[4], p5_5[5]; s_5) \wr (b_5[7], d_5)).DP'_5 \mid ((p3_6[5], p5_6; s_6) \wr (b_6[11], d_6)).DP'_6 \mid$
$(b_{10}[10]; i_{10}).(c_1[41]; r_{10}).C'_1 \mid (b_4[8]; i_4).(t_1[23] : r_4).T'_1 \mid (b_1[6]; i_1).(a_1[24], m_1[40]; r_1).A'_1 \mid$
$(b_6[9]; i_6).(g_1[25]; r_6).G'_1 \mid (b_2[7]; i_2).(a_2, m_2; r_2).A'_2 \mid (b_{11}[11]; i_{11}).(c_2[27]; r_{11}).C'_2 \mid$
$(b_7[17]; i_7).(g_2[41]; r_7).G'_2 \mid (b_3[18]; i_3).(a_3[23], m_3; r_3).A'_3 \mid (b_5[19]; i_5).(t_2[24]; r_5).T'_2 \mid$
$(b_{12}[20]; i_{12}).(c_3[25]; r_{12}).C'_3 \mid (b_8[21]; i_8).(g_3; r_8).G'_3 \mid (b_9[22]; i_9).(g_4[27]; r_9).G'_4 \mid$
$((p3_7, p5_7[12]; s_7) \wr (b_7[17], d_7)).DP'_7 \mid ((p3_8[12], p5_8[13]; s_8) \wr (b_8[18], d_8)).DP'_8 \mid$
$((p3_9[13], p5_9[14]; s_9) \wr (b_9[19], d_9)).DP'_9 \mid ((p3_{10}[14], p5_{10}[15]; s_{10}) \wr (b_{10}[20], d_{10})).DP'_{10} \mid$
$((p3_{11}[15], p5_{11}[16]; s_{11}) \wr (b_{11}[21], d_{11})).DP'_{11} \mid ((p3_{12}[16], p5_{12}; s_{12}) \wr (b_{12}[22], d_{12})).DP'_{12} \mid$
$(m[40]).(n).Me' \mid (k, k).(l).MutS' \mid (l).(n).(o).MutL' \mid$
$(o).(w).MutH' \mid ((u; r) \wr (v; s)).UvrD'$

DP_1 is connected to DP_2 via the bond on $p5_1$ and $p3_2$ with the key 1. Similarly, DP_1 is bonded to C_1 via the bond on actions b with the key 10. Also, Me is bonded with A_1 on m with the key 40. Correspondingly, there are other initial bonds with appropriate keys in the MMR system as shown in Figure 11. It should be noted that the two processes above show the same system, and so does Figure 11. The two notations above show different levels of detail, with the first only giving the process names and the second their definitions. Figure 11 shows the same level of detail as the second process. We do not say how this state emerged and how the bonds were formed.

A mismatch in the DNA section in Figure 11 is between A_2 and G_3. Methylation is on the old upper half of the strand, which means that a methyl group Me is attached to A_1. Note that there is no methyl group on A_2. This is because the methylation only happens on CTAG sequences via a specific protein, which we do not model here. No bases A are methylated on the new lower half, even if they are part of CTAG sequences, since not enough time has passed at this

stage for this to happen.

Before we show the main reactions of the MMR mechanism, we shall shorten the presentation of our MMR system so that the reactions are more easily readable and can be quickly checked. Since DP_1, DP_2, DP_3, DP_4, DP_5, DP_6, and DP_7, and the bases C_1, T_1, C_2, G_2, A_3, and G_4 (read from top to bottom and from left to right in Figure 11) never directly participate in the reactions, we will leave them out from now on. The remaining part of the syntax of the MMR system is much smaller:

$$(b_1[6];i_1).(a_1[24],m_1[40];r_1).A_1' \mid (b_6[9];i_6).(g_1[25];r_6).G_1' \mid (b_2[7];i_2).(a_2,m_2;r_2).A_2' \mid$$
$$(b_5[19];i_5).(t_2[24];r_5).T_2' \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C_3' \mid (b_8[21];i_8).(g_3;r_8).G_3' \mid$$
$$((p3_8[12],p5_8[13];s_8) \wr (b_8[18],d_8)).DP_8' \mid$$
$$((p3_9[13],p5_9[14];s_9) \wr (b_9[19],d_9)).DP_9' \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP_{10}' \mid$$
$$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP_{11}' \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP_{12}' \mid$$
$$(m[40]).(n).Me' \mid (k,k).(l).MutS' \mid (l).(n).(o).MutL' \mid$$
$$(o).(w).MutH' \mid ((u;r) \wr (v;s)).UvrD'$$

Note that leaving out some components results in appearance of several "incomplete" bonds, for example, between DP_8 and DP_7, which is left out, with the key 12.

4.3. Reactions in DNA Mismatch Repair

We now present the main reactions of the MMR system, thus illustrating the benefits of concerted actions transitions.

We start with the protein *MutS* bonding to processes A_2 and G_3 via the a_2 and g_3 actions respectively. This is the recognition of the mismatch. The transitions for creating the two bonds of *MutS* with A_2 and G_3 are as follows, where first we highlight the participating actions in bold blue font and then we highlight the actions and keys in bold black font after the bonds are created:

$(b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$

$(b_2[7];i_2).(\boldsymbol{a_2},m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$

$(b_8[21];i_8).(\boldsymbol{g_3};r_8).G'_3 \mid ((p3_8[12],p5_8[13];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9[13],p5_9[14];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$

$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(n).Me' \mid (\boldsymbol{k},\boldsymbol{k}).(l).MutS' \mid (l).(n).(o).MutL' \mid$

$(o).(w).MutH' \mid ((u;r) \wr (v;s)).UvrD'$

$\xrightarrow{ka[28]}\xrightarrow{kg[29]}$

$(b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$

$(b_2[7];i_2).(\boldsymbol{a_2[28]},m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$

$(b_8[21];i_8).(\boldsymbol{g_3[29]};r_8).G'_3 \mid ((p3_8[12],p5_8[13];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9[13],p5_9[14];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$

$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(n).Me' \mid (\boldsymbol{k[28]},\boldsymbol{k[29]}).(\boldsymbol{l}).MutS' \mid (\boldsymbol{l}).(n).(o).MutL' \mid$

$(o).(w).MutH' \mid ((u;r) \wr (v;s)).UvrD'$

Now it is possible for $MutL$ to bond to $MutS$ on l with key 30 (see Figure 12). Again, the participating actions l are in bold blue font before the transition and then they are displayed in bold black font:

$\xrightarrow{ll[30]}$ $(b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$

$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$

$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],p5_8[13];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9[13],p5_9[14];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$

$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(\boldsymbol{n}).Me' \mid (k[28],k[29]).(\boldsymbol{l[30]}).MutS' \mid (\boldsymbol{l[30]}).(\boldsymbol{n}).(o).MutL' \mid$

$(o).(w).MutH' \mid ((u;r) \wr (v;s)).UvrD'$

MutL then bonds with *Me* on n, which means it detects which of the strands is methylated, and therefore the correct one:

$\xrightarrow{nn[31]} (b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$
$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{11}[20];i_{11}).(c_3[25];r_{12}).C'_3 \mid$
$(b_8[21];i_8).(g_2[29];r_8).G'_3 \mid ((p3_8[12],p5_8[13];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$
$((p3_9[13],p5_9[14];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$
$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$
$(m[40]).(\boldsymbol{n[31]}).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(\boldsymbol{n[31]}).(\boldsymbol{o}).MutL' \mid$
$(\boldsymbol{o}).(w).MutH' \mid ((u;r) \wr (v;s)).UvrD'$

Finally, *MutL* recruits *MutH*. Henceforth, we shall use bold red font to highlight actions and keys of bonds that will be next broken. Once the bonds are broken, their actions will be shown in bold black font.

$\xrightarrow{oo[32]} (b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$
$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$
$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],\boldsymbol{p5_8[13]};\boldsymbol{s_8}) \wr (b_8[18],d_8)).DP'_8 \mid$
$((\boldsymbol{p3_9[13]},p5_9[14];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$
$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$
$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(\boldsymbol{o[32]}).MutL' \mid$
$(\boldsymbol{o[32]}).(\boldsymbol{w}).MutH' \wr ((u;r) \mid (v;s)).UvrD'$

The system at this stage is shown in Figure 12.

Next, *MutH* is ready to cleave the DNA at the right location. This is done by bonding to DP_8 with the key 33 and, at the same time, breaking the bond between DP_8 and DP_9 (with the key 13). These two simultaneous reactions are represented by the following concerted actions transition.

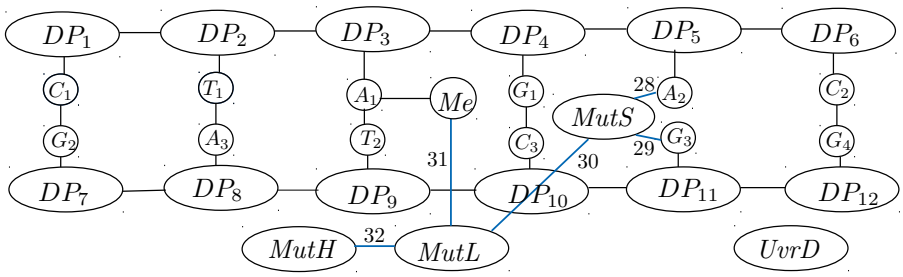

Figure 12: A six base pair DNA fragment, with a DNA mismatch and methylation of one strand. The proteins *MutL*, *MutS*, and *MutH* are bonded to the strand and ready to recruit *UvrD*, which at this point is not involved yet.

$\xrightarrow{\{ws[33],\underline{p}[13]\}} (b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$
$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$
$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],\boldsymbol{p5_8};\boldsymbol{s_8}[\mathbf{33}]) \wr (b_8[18],d_8)).DP'_8 \mid$
$((\boldsymbol{p3_9},p5_9[14];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[14],p5_{10}[15];s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$
$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$
$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$
$(o[32]).(\boldsymbol{w}[\mathbf{33}]).MutH' \mid ((u;r) \wr (v;s)).UvrD'$

Next, *promotion* happens, which moves the bond with the key 33 from a weak action s_8 to a strong action $p5_8$ in one of the sites of DP_8:

$\overset{\text{prom}}{\Rightarrow} (b_1[6];i_1).(a_1[24],m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$
$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[24];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{10}).C'_3 \mid$
$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],\boldsymbol{p5_8}[\mathbf{33}];\boldsymbol{s_8}) \wr (b_8[18],d_8)).DP'_8 \mid$
$((p3_9,\boldsymbol{p5_9}[\mathbf{14}];s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((\boldsymbol{p3_{10}}[\mathbf{14}],p5_{10}[15];\boldsymbol{s_{10}}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$
$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$
$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$
$(o[32]).(w[33]).MutH' \mid ((u;r) \wr (\boldsymbol{v};s)).UvrD'$

These two transitions give us the corresponding $\overset{\{ws[33],\underline{p}[13]\}}{\longmapsto}$ by Definition 2.

Note that *MutH* bonding with DP_8 on s_8 (key 33) leads to breaking the bond between two deoxyribose/phosphate groups DP_8 and DP_9 (key13). The situation after this step is shown in Figure 13, where the new bond (key 33) is shown as a blue line. The broken bond (key 13) is indicated by a red dotted line, and henceforth, we shall use this representation for broken bonds. Such dotted lines will be left out in subsequent figures to indicate that the relevant bonds no longer exist. This has now the right DNA strand cleaved.

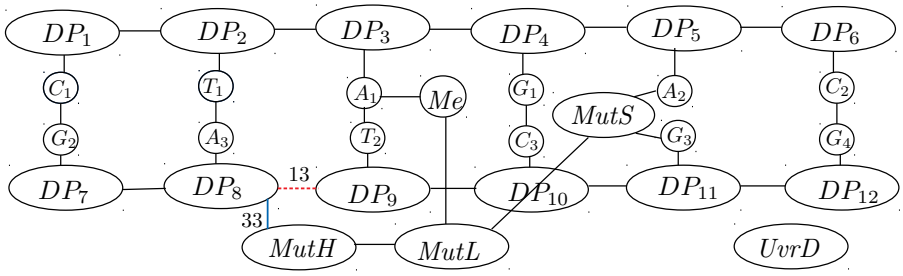

Figure 13: The mismatch has been detected by *MutS*, and *MutH* and *MutL* have been recruited. *MutL* has detected the methylation, and *MutH* has cleaved the strand containing the wrong base by bonding to DP_8 (blue line) and thus breaking the bond between DP_8 and DP_9 (red dotted line).

Next, the protein *UvrD* gets involved in the repair. It uses the cleavage to remove pairs of deoxyribose/phosphate groups and bases. Firstly, *UvrD* bonds to DP_{10} (key 34) and thus breaks the bond between DP_{10} and its left neighbour DP_9 (key 14). Then the bond 34 is promoted from weak s_{10} to strong $p3_{10}$:

$\xrightarrow{\{sv[34],\underline{p}[14]\}} (b_1[6];i_1).(\boldsymbol{a_1}\mathbf{[24]}, m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$

$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(\boldsymbol{t_2}\mathbf{[24]};\boldsymbol{r_5}).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$

$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],p5_8[33];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9,\boldsymbol{p5_9};s_9) \wr (b_9[19],d_9)) \mid .DP'_9 \mid ((\boldsymbol{p3_{10}}\mathbf{[34]},p5_{10}[15];\boldsymbol{s_{10}}) \wr (b_{10}[20],d_{10})).DP'_{10}$

$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$

$(o[32]).(w[33]).MutH' \mid ((\boldsymbol{u};r) \wr (\boldsymbol{v}\mathbf{[34]};s)).UvrD'$

Secondly, *UvrD* combines with T_2 (key 35) thus breaking the bond between T_2 and its paired base A_1 (key 24). This concerted actions transition is followed immediately by an application of prom (by Definition 2), which moves a bond from weak action r_5 to strong actions t_2.

$\xrightarrow{\{ur[35],\underline{at}[24]\}} (b_1[6];i_1).(\boldsymbol{a_1},m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$

$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(\boldsymbol{t_2}\mathbf{[35]};\boldsymbol{r_5}).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$

$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],p5_8[33];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9,p5_9;s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((p3_{10}[34],p5_{10}[15];\boldsymbol{s_{10}}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$

$((p3_{11}[15],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$

$(o[32]).(w[33]).MutH' \mid ((\boldsymbol{u}\mathbf{[35]};r) \wr (v[34];s)).UvrD'$

Note that *UvrD* does not attach to DP_9 initially, but to DP_{10}. If it had been attached to DP_9, it would have had no connection to the DNA strand once the bonds of DP_9 and T_2 to the rest of DNA strand have been broken.

The removal of the first base and deoxyribose/phosphate group is shown in Figure 14. The blue lines show the bonds that have been created and the red dotted lines indicate the bonds that have been broken. The two pairs of concerted actions transitions have the keys (34,14) and (35,24) respectively.

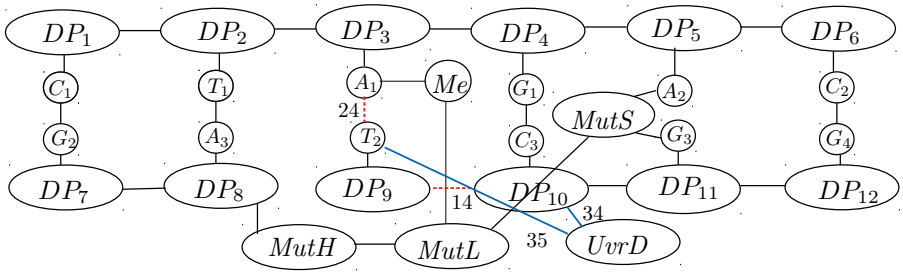

Figure 14: *UvrD* has now bonded to T_2 and DP_{10} (blue lines), thus breaking the bonds of the DP_9-T_2 group to the rest of the DNA (red dotted lines).

UvrD can now "walk" along the lower chain of the strand towards the right end. Concurrently, the bonds between the deoxyribose/phosphate groups are also broken as *UvrD* progresses. We write these reactions as two separate sequences of transitions for clarity, with the first transition in full, and note that

this is the place where could have used the shortcut concert3 SOS rule.

$(b_1[6];i_1).(a_1,m_1[40];r_1).A'_1 \mid (b_6[9];i_6).(g_1[25];r_6).G'_1 \mid$

$(b_2[7];i_2).(a_2[28],m_2;r_2).A'_2 \mid (b_5[19];i_5).(t_2[35];r_5).T'_2 \mid (b_{12}[20];i_{12}).(c_3[25];r_{12}).C'_3 \mid$

$(b_8[21];i_8).(g_3[29];r_8).G'_3 \mid ((p3_8[12],p5_8[33];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9,p5_9;s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((\boldsymbol{p3_{10}[34]},\boldsymbol{p5_{10}[15]};s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$

$((\boldsymbol{p3_{11}[15]},p5_{11}[16];\boldsymbol{s_{11}}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$

$(o[32]).(w[33]).MutH' \mid ((u[35];r) \wr (\boldsymbol{v[34]};\boldsymbol{s})).UvrD'$

$\xrightarrow{\{ss[36],\underline{vp}3[34]}\!\!\!\!\rightarrow \xrightarrow{\underline{p}[15]}$

$(b_1[6];i_1).(a_1,m_1[40];r).A'_1 \mid (b_6[9];i_6).(\boldsymbol{g_1[25]};r_3).G'_1 \mid$

$(b_2[7];i_2).(a_2[28],m_2;r).A'_2 \mid (b_5[19];i_5).(\boldsymbol{t_2[35]};r_2).T'_2 \mid (b_{11}[20];i_{11}).(\boldsymbol{c_3[25]};\boldsymbol{r_8}).C'_3 \mid$

$(b_8[21];i_8).(g_2[29];r_4).G'_3 \mid ((p3_8[12],p5_8[33];s_8) \wr (b_8[18],d_8)).DP'_8 \mid$

$((p3_9,p5_9;s_9) \wr (b_9[19],d_9)).DP'_9 \mid ((\boldsymbol{p3_{10}},\boldsymbol{p5_{10}};s_{10}) \wr (b_{10}[20],d_{10})).DP'_{10} \mid$

$((\boldsymbol{p3_{11}[36]},p5_{11}[16];\boldsymbol{s_{11}}) \wr (b_{11}[21],d_{11})).DP'_{11} \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP'_{12} \mid$

$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$

$(o[32]).(w[33]).MutH' \mid ((\boldsymbol{u[35]};\boldsymbol{r}) \wr (\boldsymbol{v[36]};\boldsymbol{s})).UvrD'$

Notice that promotion has moved a bond from s_{11} to $p3_{11}$ and from s to v. Instead of using these transitions we could have applied concert3, we could have used this sequence of transitions from the source to the target process: $\xrightarrow{\{ss[36],\underline{vp}3[34],\underline{p}[15]\}} \stackrel{\text{prom}}{\Rightarrow} \stackrel{\text{prom}}{\Rightarrow}$.

Next, *UvrD* combines with C_3 thus breaking bonds with keys 35 and 25.

$\xrightarrow{\{rr[37],\underline{tu}[35]\}} \xrightarrow{\underline{cg}[25]} (b_1[6];i_1).(a_1,m_1[40];r).A_1' \mid (b_6[9];i_6).(\boldsymbol{g_1};r_3).G_1' \mid$

$(b_2[7];i_2).(a_2[28],m_2;r).A_2' \mid (b_5[19];i_5).(\boldsymbol{t_2};r_2).T_2' \mid (b_{11}[20];i_{11}).(\boldsymbol{c_3[37]};\boldsymbol{r_8}).C_3' \mid$

$(b_8[21];i_8).(g_2[29];r_4).G_3' \mid ((p3_8[12],p5_8[33];s_8) \wr (b_8[18],d_8)).DP_8' \mid$

$((p3_9,p5_9;s_9) \wr (b_9[19],d_9)).DP_9' \mid ((p3_{10},p5_{10};s_{10}) \wr (b_{10}[20],d_{10})).DP_{10}' \mid$

$((p3_{11}[36],p5_{11}[16];s_{11}) \wr (b_{11}[21],d_{11})).DP_{11}' \mid ((p3_{12}[16],p5_{12};s_{12}) \wr (b_{12}[22],d_{12})).DP_{12}' \mid$

$(m[40]).(n[31]).Me' \mid (k[28],k[29]).(l[30]).MutS' \mid (l[30]).(n[31]).(o[32]).MutL' \mid$

$(o[32]).(w[33]).MutH' \mid ((\boldsymbol{u[37]};\boldsymbol{r}) \wr (v[36];s)).UvrD'$

Promotion moves bonds from r_8 to c_3 and from r to u below. Correspondingly as above, we could use shortcut rule concert3 and write these reactions using this sequence: $\xrightarrow{\{rr[37],\underline{tu}[35],\underline{cg}[25]\}} \Rightarrow \Rightarrow$.

The resulting system is shown in Figure 15. Here, the first base T_2 and deoxyribose/phosphate group DP_9 have been removed (shown in grey) and *UvrD* has moved to the next base C_3 and deoxyribose/phosphate group DP_{10}, again to remove them. The two concerted actions transitions that represent these reactions are with the keys (36,34,15) and (37,35,25) respectively.

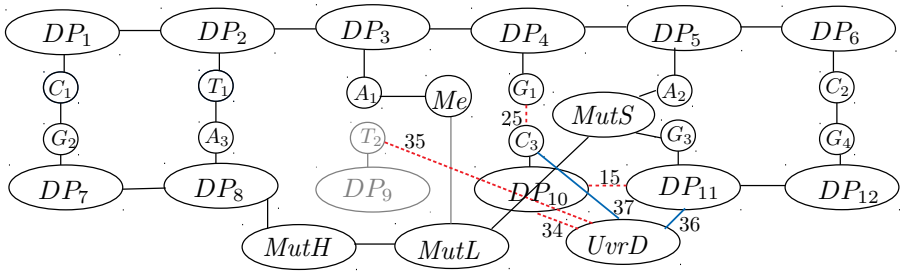

Figure 15: *UvrD* has now bonded to C_3 and DP_{11}, thus breaking the bonds of the DP_{10}-C_3 group with the rest of the DNA (via two concerted actions). The T_2-DP_9 group is no longer attached to the DNA (shown in grey).

This process continues until the offending base G_3 has been removed. We do not show concerted actions transitions that model this since they correspond very closely to the last two transitions. Figure 16 shows the result of these

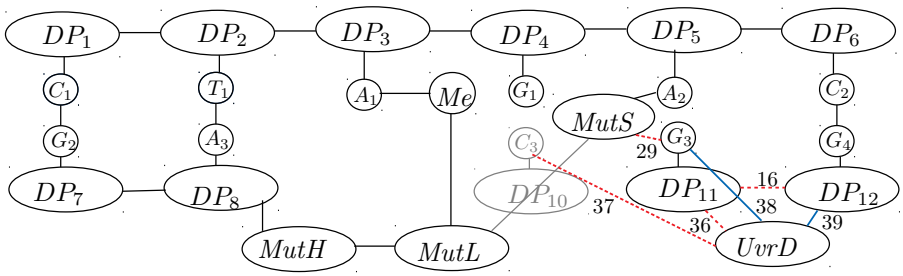

Figure 16: *UvrD* has now bonded to G_3 (the offending base) and DP_{12}, thus breaking the bonds of the G_3-DP_{11} group to the rest of the DNA (via two concerted actions). The C_3-DP_{10} group is now removed (shown in grey). Next, the group G_3-DP_{11} will be removed.

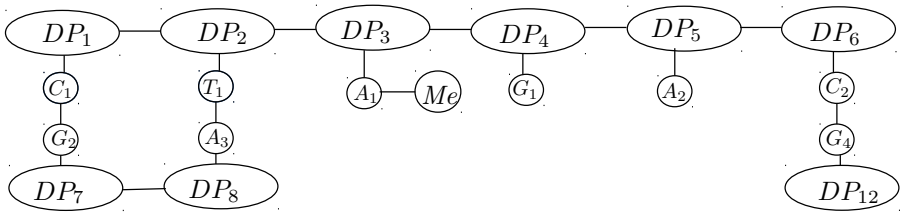

Figure 17: A DNA fragment with a gap of three bases in one strand produced by MMR.

two transitions which remove of the offending base. Once this is done and the *MutH*, *MutL*, and *MutS* proteins have detached we get the situation shown in Figure 17.

The resulting system is now ready for other proteins to replace the bases and deoxyribose/phosphate groups. Notice that the remaining strand contains the necessary information for this, in particular that only a T base can bond with A_1 and A_2 and that a C must bond with G_1. Note that the methyl group stays attached to A_1, since it is still an old DNA strand. Also the appropriate groups in the new, lower half will be methylated, so that both strands are marked as old material after the next DNA duplication process.

4.4. Evaluation

We have shown that CCB can be used to model the MMR gene repair mechanism. This is significant since the calculus was designed for simpler lower level systems and reactions. We have been able to reuse the main features of

CCB, in particular, the concerted actions. Out-of-causal order reversibility plays a significant role in our example, in particular for the modelling of movement of the proteins along the DNA chain.

There is one aspect of our reactions which we model adequately but which we intend to improve in the future. This is the breaking of two bonds as a result of one bond on weak actions being formed as, for example, in these transitions $\xrightarrow{\{rr[37],\underline{tu}[35],\underline{cg}[25]\}}\stackrel{\mathsf{prom}}{\Rightarrow}\stackrel{\mathsf{prom}}{\Rightarrow}$ derived with the shortcut concert3 rule. Currently, we represent this by two consecutive transitions $\xrightarrow{\{rr[37],\underline{tu}[35]\}}\xrightarrow{\underline{cg}[25]}$, but note that these transitions can be interleaved by other transitions, which weakens the modelling of the two bonds being broken immediately after the weak bond is created. In short, our concert rules, which were developed for covalent reactions, are not general enough.

Our calculus permits the modelling of certain reactions which normally do not happen in reality, and this property is inherited from the original CCB. This is mainly due to non-consideration of aspects of spatial configuration of proteins and bases. We give several examples of reactions where do not model the spacial aspect adequately:

- We have not modelled the formation of the loop in the DNA, which is crucial for cleaving the correct strand. In reality, *MutL* forms a loop in the strand, which hides the methylated strand, making sure in this way that *MutH* cleaves the un-methylated (meaning new) strand.

- In our model, *MutL* can bond to any methyl group and a base pair any distance apart. In reality, *MutL* has a fixed size and it can only react with parts of the DNA it can physically reach.

- Similarly, *UvrD* can bond to any deoxyribose/phosphate group, not just the neighbouring one. Moreover, when *UvrD* bonds on weak s, there is no way to differentiate between $p3$ or $p5$ of *DP* being broken in our model.

We have checked that all reactions given in this section are possible in our model by using the simulation software tool [4], which has been extended to

cover the extended prefix operator.

5. Conclusion

In this paper, we have used a Calculus of Covalent Bonding to model an important gene repair pathway, namely the DNA Mismatch Repair (MMR) mechanism. Since the molecules that we have considered, and reactions between them, are more complex than simple compounds of atoms (as in our previous applications), we have extended the calculus with a prefix operator that can represent multiple bonding sites. We have discovered that we can model adequately creation and breaking of bonds (that appear in the out-of-causal order fashion) involving two bonds (as is the case in covalent reactions). However, when a bond is created on two weak actions, this sometime requires instantaneous breaking of two bonds, as, for example, when *UvrD* bonds with DP_{11}. This we cannot model directly in CCB. Instead, we use two transitions but can also use a shorthand notation via our concert3 rule. We intend to re-develop the operational semantics for the extended CCB so that both two action and three action concerted transitions are allowed. This will also involve adding a suitable choice operator and producing a fuller set of reduction rules especially for the restriction operator.

The MMR mechanism is a complex pathway. We have modelled it with twelve deoxyribose/phosphate groups, twelve bases and four helper proteins, all composed in parallel. We then have shown successfully a sequence of transitions (representing bonding reactions and cascades of concerted reactions with either two or three reactions each) that represent the MMR mechanism. As stated above, we have used a shortcut rule concert3, which only is an approximation of how such reactions happen in reality. We have not been able to consider spatial aspects of such reactions in the current version of CCB, which is left to future research. Reaction rates is another useful addition to CCB and its simulator that we would be interested to develop in the future.

References

[1] S. Kuhn, I. Ulidowski, A calculus for local reversibility, in: S. Devitt, I. Lanese (Eds.), Reversible Computation 2016, Vol. 9720 of LNCS, Springer, 2016, pp. 20–35.

[2] S. Kuhn, I. Ulidowski, Local reversibility in a calculus of covalent bonding, Science of Computer Programming 151 (2018) 18–47.

[3] I. Phillips, I. Ulidowski, S. Yuen, A reversible process calculus and the modelling of the ERK signalling pathway, in: R. Glück, T. Yokoyama (Eds.), Reversible Computation, Vol. 7581 of LNCS, Springer, 2013, pp. 218–232.

[4] S. Kuhn, Simulation of base excision repair in the calculus of covalent bonding, in: J. Kari, I. Ulidowski (Eds.), Reversible Computation 2018, Vol. 11106 of LNCS, Springer, 2018, pp. 123–129.

[5] L. A. Pray, DNA replication and causes of mutation, Nature Education 1 (1) (2008) 214.

[6] D. J. Higham, Modeling and simulating chemical reactions, SIAM Review 50 (2) (2008) 347–368.

[7] W. Fontana, L. Buss, The barrier of objects: From dynamical systems to bounded organizations, IIASA Working Paper, International Institute for Applied Systems Analysis, IIASA, Austria (March 1996).

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

[9] A. Regev, W. Silverman, E. Shapiro, Representing biomolecular processes with computer process algebra: π-calculus programs of signal transduction pathways, in: R. B. Altmann, A. K. Dunker, L. Hunker (Eds.), Pacific Symposium on Biocomputing 2001, World Scientific, Singapore, 2001, pp. 459–470.

[10] C. Priami, Stochastic π-calculus, The Computer Journal 38 (7) (1995) 578–589.

[11] C. Priami, A. Regev, E. Shapiro, W. Silverman, Application of a stochastic name-passing calculus to representation and simulation of molecular processes, Information Processing Letters 80 (1) (2001) 25–31.

[12] F. Ciocchetta, J. Hillston, Bio-PEPA: A framework for the modelling and analysis of biological systems, Theoretical Computer Science 410 (33-34) (2009) 3065–3084.

[13] G. Păun, Computing with membranes, Journal of Computer and System Sciences 61 (1) (2000) 108–143.

[14] G. Ciobanu, Distributed algorithms over communicating membrane systems, Biosystems 70 (2) (2003) 123–133.

[15] G. Ciobanu, L. Pan, G. Păun, M. J. Pérez-Jiménez, P Systems with minimal parallelism, Theoretical Computer Science 378 (1) (2007) 117–130.

[16] D. Besozzi, G. Ciobanu, A P System description of the sodium-potassium pump, in: G. Mauri, G. Păun, M. J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, Springer, 2005, pp. 210–223.

[17] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, E. Shapiro, BioAmbients: an abstraction for biological compartments, Theoretical Computer Science 325 (1) (2004) 141–167.

[18] L. Cardelli, A. D. Gordon, Mobile ambients, Theoretical Computer Science 240 (1) (2000) 177–213.

[19] L. Cardelli, Brane calculi, in: V. Danos, V. Schachter (Eds.), Computational Methods in Systems Biology, Vol. 3082 of LNCS, Springer, 2005, pp. 257–278.

[20] V. Danos, S. Pradalier, Projective brane calculus, in: V. Danos, V. Schachter (Eds.), Computational Methods in Systems Biology 2004, Vol. 3082 of LNCS, Springer, 2005, pp. 134–148.

[21] M. Pedersen, G. D. Plotkin, A language for biochemical systems: design and formal specification, in: C. Priami, R. Breitling, D. Gilbert, M. Heiner, A. M. Uhrmacher (Eds.), Transactions on Computational Systems Biology XII, Vol. 5945 of LNCS, Springer, 2010, pp. 77–145.

[22] G. D. Plotkin, A Calculus of Chemical Systems, Springer, 2013, pp. 445–465.

[23] F. Fages, S. Soliman, N. Chabrier-Rivier, Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM, Journal of Biological Physics and Chemistry 4 (2004) 64–73.

[24] V. Danos, J. Krivine, Reversible communicating systems, in: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory, Vol. 3170 of LNCS, Springer, 2004, pp. 292–307.

[25] I. Phillips, I. Ulidowski, Reversing algebraic process calculi, in: L. Aceto, A. Ingólfsdóttir (Eds.), International Conference on Foundations of Software Science and Computation Structures 2006, Vol. 3921 of LNCS, Springer, 2006, pp. 246–260.

[26] I. Phillips, I. Ulidowski, Reversing algebraic process calculi, The Journal of Logic and Algebraic Programming 73 (1) (2007) 70–96.

[27] S. Kuhn, B. Aman, G. Ciobanu, A. Philippou, K. Psara, I. Ulidowski, Reversibility in chemical reactions, in: I. Ulidowski, I. Lanese, U. P. Schultz, C. Ferreira (Eds.), Reversible Computation: Extending Horizons of Computing. Selected Results of the COST Action IC1405, Vol. 12070 of LNCS, Springer, 2020, pp. 151–176.

[28] I. Lanese, C. A. Mezzina, J.-B. Stefani, Reversing Higher-Order Pi, in: P. Gastin, F. Laroussinie (Eds.), CONCUR 2010 - Concurrency Theory, Vol. 6269 of LNCS, Springer, 2010, pp. 478–493.

[29] V. Danos, C. Laneve, Formal molecular biology, Theoretical Computer Science 325 (1) (2004) 69–110.

[30] B. Aman, G. Ciobanu, Reversible computation in nature inspired rule-based systems, Journal of Membrane Computing 2 (4) (2020) 246–254.

[31] M. Heiner, D. Gilbert, R. Donaldson, Petri nets for systems and synthetic biology, in: M. Bernardo, P. Degano, G. Zavattaro (Eds.), Formal Methods for Computational Systems Biology, Springer, 2008, pp. 215–264.

[32] A. Philippou, K. Psara, Reversible computation in Petri nets, in: J. Kari, I. Ulidowski (Eds.), Reversible Computation 2018, Vol. 11106 of LNCS, Springer, 2018, pp. 84–101.

[33] K. Barylska, A. Gogolinska, L. Mikulski, A. Philippou, M. Piatkowski, K. Psara, Reversing computations modelled by coloured Petri nets, in: W. M. P. van der Aalst, R. Bergenthum, J. Carmona (Eds.), International Workshop on Algorithms & Theories for the Analysis of Event Data 2018, Vol. 2115 of CEUR Workshop Proceedings, CEUR-WS.org, 2018, pp. 91–111.

[34] H. C. Melgratti, C. A. Mezzina, I. Ulidowski, Reversing place transition nets, Logical Methods in Computer Science 16 (4).

[35] H. C. Melgratti, C. A. Mezzina, I. Phillips, G. M. Pinna, I. Ulidowski, Reversible occurrence nets and causal reversible prime event structures, in: I. Lanese, M. Rawski (Eds.), Reversible Computation 2020, Vol. 12227 of LNCS, Springer, 2020, pp. 35–53.

[36] A. Philippou, K. Psara, Reversible computation in cyclic Petri nets, CoRR abs/2010.04000.

[37] V. Danos, J. Krivine, Formal molecular biology done in CCS-R, Electron. Notes Theor. Comput. Sci. 180 (3) (2007) 31–49.

[38] I. Lanese, I. C. C. Phillips, I. Ulidowski, An axiomatic approach to reversible computation, in: J. Goubault-Larrecq, B. König (Eds.), Foundations of Software Science and Computation Structures 2020, Vol. 12077 of LNCS, Springer, 2020, pp. 442–461.

[39] R. Milner, A Calculus of Communicating Systems, 1st Edition, Vol. 92 of LNCS, Springer, 1980.

[40] W. Fokkink, Introduction to Process Algebra, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2000.

[41] I. Ulidowski, Equivalences on observable processes, in: Proceedings of LICS 1992, IEEE Computer Society, 1992, pp. 148–159.

[42] P. Modrich, Mechanisms in E. coli and Human Mismatch Repair (Nobel Lecture), Angewandte Chemie International Edition in English 55 (30) (2016) 8490–8501.