Scientific Annals of Computer Science vol. 21, 2011, pp. 175–198

Reversibility in Massive Concurrent Systems

Luca Cardelli[1], Cosimo Laneve[2]

Abstract

We introduce *reversible structures*, an algebra for massive concurrent systems, where terms retain bits of causal dependencies that allow one to reverse computation histories. We then study the implementation of (weak coherent) reversible structures in *three-domains* DNA *strands*, which is the natural model that has inspired reversible structures. We finally provide schemas for modeling significant synchronization patterns of process algebra into reversible structures and discuss the encoding of asynchronous Reversible CCS.

Keywords: Reversible algebra, DNA strands, computational histories, synchronization patterns, encodings.

1 Introduction

Reversing a (forward) computation history means undoing the history. In concurrent systems, undoing the history is not performed in a deterministic way but in a causally consistent fashion, where states that are reached during a backward computation are states that could have been reached during the computation history by just performing independent actions in a different order. In RCCS [5], Danos and Krivine achieve this by attaching a memory m to each process P, in the monitored process construct $m : P$. Memories in RCCS are stacks of information needed for processes to backtrack. Alternatively, Phillips and Ulidowski propose a technique for reversing process calculi without using memories [16]. In this technique, the structure of processes is not destroyed and the progress is noted by underlining the actions that have been performed. In order to tag the communicating processes, they generate unique identifiers on-the-fly during the communications.

[1]Microsoft Research, Cambridge. Email: luca@microsoft.com
[2]Università di Bologna. Email: laneve@cs.unibo.it

These foundational studies of reversible and concurrent computations have been largely stimulated by areas such as chemical and biological systems – called *massive concurrent systems* in the following – where operations are reversible, and only an appropriate injection of energy and/or a change of entropy can move the computational system in a desired direction.

However there is a mismatch between chemical and biological systems and the above concurrent formalisms. In the latter ones, reversibility means *desynchronizing processes that actually interacted in the past* while, in massive concurrent systems, reversibility means *reversibility of configurations*. In order to make massive concurrent systems reversible with the process calculus meaning, one has to remember the position and momentum of each molecule, which is precisely contrary to the well-mixing assumption of biochemical soups, namely that the probability of collision between two molecules is independent of their position (*cf.* Gillespie's algorithm [8]).

We introduce an algebra for massive concurrent systems, called *reversible structures*, where terms retain bits of causal dependencies that allow one to reverse computation histories. These bits permit to trace effects of interactions, but not to the point of being able to identify the precise molecule that caused an effect. For example, it is not possible to determine the signal that causes a reduction among the many several of the same population. It is worth to remark that, in reversible structures with populations of species that are singletons (called *coherent reversible structures* in [4]), causality has a meaning that is consistent with that of standard process calculi [5, 16]. While these latter structures are not currently realizable, they may become realizable in the future if we learn how to control individual molecules.

Reversible structures may implement significant CCS-style interaction patterns (Cardelli already noticed this by studying a class of reversible systems – the DNA chemical systems [2, 3]). Consider for example a binary operator that takes two input molecules and produces one unrelated output molecule when (and only when) both inputs are present. It is too difficult to engineer the input machinery in order to account for any possible pattern of interaction, and to produce the output molecule out of their own structure. This operator is therefore implemented by an artifact that binds the two inputs one after the other and then releases the output out of its own structure. Of course, if the second input never comes, the structure must release the first input, because the first input may be legitimately used by some other operator. This means that the binding of the first input must be reversible, and the natural reversibility of our structures is exploited to

Figure 1: A DNA domain

achieve correctness.

These remarks allow us to draw a precise measure of the expressive power of reversible structures. In fact we compile a sub-calculus of RCCS [5], its asynchronous fragment, into reversible structure and demonstrate a strong correspondence between causalities in the two formalisms.

Structure of the paper. In Section 2 we overview the model that inspired our algebra: DNA three-domains strands and their dynamics. In Section 3 we define reversible structures and discuss a few properties of these structures. In Section 4 we study the encoding of reversible structures in DNA circuits. In Section 5 we model standard synchronization patterns in our formalism. In Section 6 we study the compilation of asynchronous RCCS [5] in reversible structures. We conclude in Section 7 by discussing the theoretical results in [4] and by outlining some future work.

This paper contains an introduction to reversible structures by discussing the motivations that led to their definition and by establishing a precise relationship between reversible structures and reversible process calculi. The purpose of the companion paper [4] is to provide a detailed presentation of the theory developed to date and to discuss algorithmic issues of decision problems.

2 Three-domains DNA strands and causality

There are many ways of computing with DNA structures. They all use the Watson-Crick complement that we briefly discuss.

DNA strands are sequences of bases (Adenine, Cytosine, Guanine, and Thymine). There are subsequences of them, called *domains*, that are *independent* of each other and cannot hybridize from any other domain except the sequence consisting of complementary bases (Adenine is complementary to Thymine, Cytosine is complementary to Guanine). In Figure 1 we illustrate a strand of three domains, that have different names because the corresponding sequences of bases are different. Single strands have an orientation; double strands are composed of two single strands with opposite orientations, where

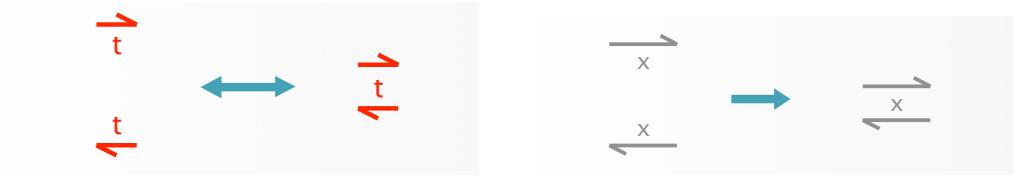


Figure 2: DNA domain hybridizations

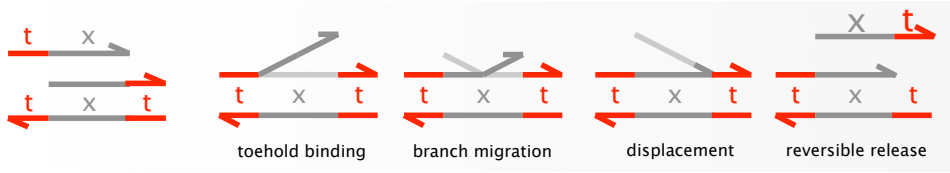


Figure 3: Reversible branch migration

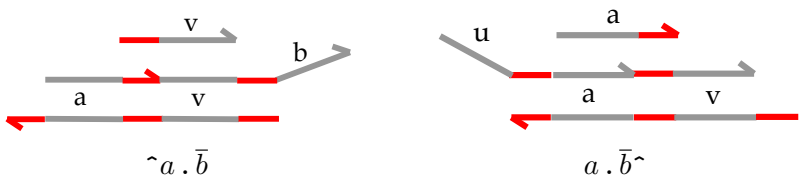
the bottom strand is the complement of the top strand. The "short" domains, called *toeholds* and depicted in red in the following figures, hybridize (bind) *reversibly* to their complements, while the "long" domains hybridize irreversibly; the exact critical length depends on physical condition. Figure 2 illustrates the hybridizations where distinct letters indicate domains that do not hybridize with each other.

An additional fundamental mechanism, called *toehold mediated branch migration* [18], allows displacements of strands composed of long- and short-domains in a reversible way, as illustrated in the leftmost picture of Figure 3. In the first reaction of Figure 3, a toehold t initiates a binding between a double strand and a single strand. After the (reversible) binding of the toehold, the x domain of the single strand gradually replaces the top x strand of the double strand by *branch migration*. The branching point between the two top x domains performs a random walk that eventually leads to the displacement of the x strand. If the toehold matches but the branch migration region does not match, then the signal will eventually break off from the gate, as if nothing had happened. The last reaction unbinds the rightmost toehold of the double strand, thus obtaining a single strand $x{:}t$. The whole process may be reverted by binding the toehold of the single strand to the rightmost toehold of the double strand.

In this paper we consider a somewhat different subset of DNA strands (the reasons are given in the following discussion about causal dependencies), which is a refined version of the "see-saw gates" model of Quian and

Winfree [17]. Our DNA system consists of signals and gates. A signal is encoded as a single-stranded DNA sequence consisting of three contiguous domains as illustrated in Figure 1. The first domain is a "history" domain arising from previous interactions: it can be in principle arbitrarily long and is not part of the signal identity. That is, two signals are equal provided they are equal in the second and third domains, ignoring the history domain. The second domain is a toehold: it initiates the signal processing. The third long-domain is a branch migration domain: it stabilizes the interaction initiated by toehold binding.

Branch migration, like toehold binding, is also a reversible reaction: it is a bidirectional random walk, zipping and unzipping complementary DNA strands. Although there is no directionality, one can arrange that when branch migration randomly reaches one end of a region, it causes another strand to detach. If that detached strand has a toehold to go back, it can then start reversible branch migration again, and we have a reversible overall reaction: this whole process is called toehold exchange. As an example we discuss the simplest DNA gate: a *reversible signal transducer*. A transducer $\hat{}a\,.\,\overline{b}$ takes an input signal a (a fixed input) and produces an output signal $\overline{b}$ becoming $a\,.\,\overline{b}\hat{}$. The expected reduction semantics is $\overline{a} \mid \hat{}a\,.\,\overline{b} \quad \leftrightarrow \quad a\,.\,\overline{b}\hat{} \mid \overline{b}$. We model $\hat{}a\,.\,\overline{b}$ and $a\,.\,\overline{b}\hat{}$ with the structures



$\hat{}a\,.\,\overline{b}$ $\qquad$ $a\,.\,\overline{b}\hat{}$

and we discuss the behaviour of $\hat{}a\,.\,\overline{b}$ when a signal with name a binds to it (we assume toeholds are always complementary). In Figure 4, leftmost picture, the toehold of the signal binds to the complementary toehold of the gate. Then branch migrations starts, where the domain a of the signal and the upper strand a of the gate compete for the lower strand. When branch migration (by a random walk) reaches the right end, the only thing holding the a domain is the second toehold of the gate, which can hence detach – see the picture in the middle of Figure 4. Still, this toehold can reattach, and the branch migration may then reach the left end causing the signal a with history u to detach. Therefore, this toehold exchange is fully reversible. It is also possible, however, that the single strand labelled v attaches to the gate while a is detached, see the rightmost picture of Figure 4. This will displace

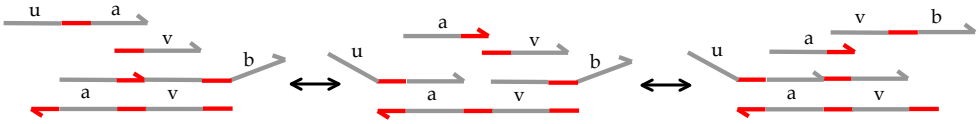


Figure 4: The dynamics of a transducer: the leftmost solution represents $\overline{a} \mid \hat{}a \,.\, \overline{b}$, the rightmost one represents $a \,.\, \overline{b}\hat{} \mid \overline{b}$

a signal b leaving unbound the rightmost lower toehold of the gate. Because of this, the signal may bind again to the gate, thus reverting the behaviour.

Three-domains DNA strands carry bits of information that often allow one to reverse computations in a causally consistent fashion. To illustrate the question, consider the solution (written in process algebraic form for simplicity)

$$\overline{a} \mid \overline{b} \mid \hat{}a \,.\, \overline{c} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e}$$

and the computation

$$\begin{array}{lrll} \overline{a} \mid \overline{b} \mid \hat{}a \,.\, \overline{c} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e} & \rightarrow & \overline{b} \mid \overline{c} \mid a \,.\, \overline{c}\hat{} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e} & (1) \\ & \rightarrow & \overline{b} \mid \overline{d} \mid a \,.\, \overline{c}\hat{} \mid \hat{}b \,.\, \overline{c} \mid c \,.\, \overline{d}\hat{} \mid \hat{}c \,.\, \overline{e} & (2) \\ & \rightarrow & \overline{c} \mid \overline{d} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid c \,.\, \overline{d}\hat{} \mid \hat{}c \,.\, \overline{e} & (3) \\ & \rightarrow & \overline{d} \mid \overline{e} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid c \,.\, \overline{d}\hat{} \mid c \,.\, \overline{e}\hat{} & (4) \end{array}$$

where the c signal of the transducer $\hat{}a \,.\, \overline{c}$ has triggered the transducer $\hat{}c \,.\, \overline{d}$ and the c signal of the transducer $\hat{}b \,.\, \overline{c}$ has triggered the transducer $\hat{}c \,.\, \overline{e}$. This computation may be reversed in different causally consistent ways. One of its is to reverse the reductions (2) and (4) because independent (they concern different terms):

$$\begin{array}{lrl} \overline{d} \mid \overline{e} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid c \,.\, \overline{d}\hat{} \mid c \,.\, \overline{e}\hat{} & \rightarrow & \overline{c} \mid \overline{d} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid c \,.\, \overline{d}\hat{} \mid \hat{}c \,.\, \overline{e} \\ & \rightarrow & \overline{c} \mid \overline{c} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e} \end{array}$$

In this last solution $\overline{c} \mid \overline{c} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e}$ it is not possible to determine the c that caused either the signal $\overline{d}$ or $\overline{e}$ because biological systems are massively concurrent. Therefore one ends with identifying the above computation with

$$\begin{array}{lrl} \overline{a} \mid \overline{b} \mid \hat{}a \,.\, \overline{c} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e} & \rightarrow & \overline{b} \mid \overline{c} \mid a \,.\, \overline{c}\hat{} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e} \\ & \rightarrow & \overline{b} \mid \overline{e} \mid a \,.\, \overline{c}\hat{} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid c \,.\, \overline{e}\hat{} \\ & \rightarrow & \overline{c} \mid \overline{e} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid \hat{}c \,.\, \overline{d} \mid c \,.\, \overline{e}\hat{} \\ & \rightarrow & \overline{d} \mid \overline{e} \mid a \,.\, \overline{c}\hat{} \mid b \,.\, \overline{c}\hat{} \mid c \,.\, \overline{d}\hat{} \mid c \,.\, \overline{e}\hat{} \end{array}$$

It is worth noticing that such identities may become troublesome as soon as different causal dependencies produce different visible effects, such as different colors of the solutions.

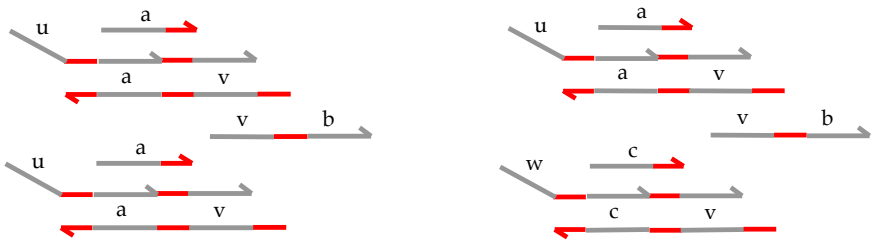
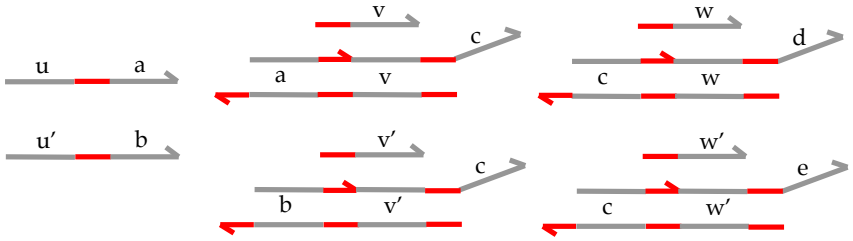


Figure 5: Causality problems

A standard solution to this problem is to record the strands that synchronized and the literature reports several techniques to achieve this [11, 1, 5, 16, 9]. Our three-domain structures implement the technique proposed by Lévy [11] that, in the above example, amounts to using signals c with different histories according to whether they are produced by the transducer $\hat{}a \,.\, \overline{c}$ or $\hat{}b \,.\, \overline{c}$. In facts, the reader may remark that there is no mixing of causal dependencies in the solution



that encodes $\overline{a} \mid \overline{b} \mid \hat{}a \,.\, \overline{c} \mid \hat{}b \,.\, \overline{c} \mid \hat{}c \,.\, \overline{d} \mid \hat{}c \,.\, \overline{e}$ in three domain structures.

However, mixing of causalities is still possible in three-domain DNA strands because of massive concurrency and because of bad designs. In Figure 5, left picture, it is not possible to determine if the signal b has been produced by the lower or the upper gate, because they belong to the same species. As we said, this kind of confusion is unavoidable in massive concurrent systems. In the right picture of Figure 5, again, the signal b may bind either to the lower or the upper gate, even if they do not belong to the same species. In this case the situation is worse because the designer has used the same history id in two different gates. We notice that these solutions may be banned by a simple static verifier enforcing that different species retain different history ids (called weak coherence, see Section 3).

3 The algebra of reversible structures

In this section we define a process algebra, called *reversible structures*, for the three-domains DNA strands discussed in the previous section. We use three disjoint infinite sets: *names* $\mathcal{N}$, ranged over by $a, b, c, \cdots$, *co-names* $\overline{\mathcal{N}}$, ranged over $\overline{a}, \overline{b}, \overline{c}, \cdots$, and a countable set of *id*s, ranged over u, v, w, $\cdots$. Names and co-names are ranged over by $\alpha, \alpha', \cdots$ and $\overline{\overline{\alpha}} = \alpha$. Names and ids are ranged over x, x', $\cdots$. The following notations for *sequences of actions* will be used:

- sequences of $\mathcal{N}$ are ranged over by $\mathtt{A}$, $\mathtt{B}$, $\cdots$;
- sequences of elements $u{:}\overline{a}$ are ranged over by $\overline{\mathtt{A}}$, $\overline{\mathtt{B}}$, $\cdots$;
- sequences of elements $u{:}a$ are ranged over by $\mathtt{A}^{\perp}$, $\mathtt{B}^{\perp}$, $\cdots$;

The empty sequence is represented by ε; the length of a sequence is given by the function $\mathit{length}(\cdot)$.

The syntax of *reversible structures* includes *gates* g and *structures* $\mathtt{S}$ and consists of the rules:

$$
\begin{array}{rcll}
g & ::= & \mathtt{A}^{\perp}\,.\,\hat{}\,\mathtt{B}\,.\,\overline{\mathtt{C}} & (\mathit{length}(\mathtt{A}^{\perp}\,.\,\mathtt{B}) > 0) \\
 & | & \mathtt{A}^{\perp}\,.\,\overline{\mathtt{B}}\,.\,\hat{}\,\overline{\mathtt{C}} & (\mathit{length}(\mathtt{A}^{\perp}) > 0)
\end{array}
$$

$$
\begin{array}{rcll}
\mathtt{S} & ::= & & \\
 & & \mathbf{0} & \text{(null)} \\
 & | & u{:}\overline{a} & \text{(signal)} \\
 & | & g & \text{(gate)} \\
 & | & \mathtt{S} \mid \mathtt{S} & \text{(parallel)} \\
 & | & (\mathtt{new}\ x)\ \mathtt{S} & \text{(new)}
\end{array}
$$

A gate is a term that accepts input signals $u{:}\overline{a}$ and emits output signals, reversibly. The form $\mathtt{A}^{\perp}\,.\,\hat{}\,\mathtt{B}.\overline{\mathtt{C}}$ represents input-accepting gates, at least when not considering reverse reactions. $\mathtt{A}^{\perp}$ are the inputs that have been processed, $\mathtt{B}$ are the inputs still to be processed, and $\overline{\mathtt{C}}$ are the outputs to be emitted. The other form $\mathtt{A}^{\perp}\,.\,\overline{\mathtt{B}}\,.\,\hat{}\,\overline{\mathtt{C}}$ represents an output-producing gate (when not considering reverse reactions). The $\mathtt{A}^{\perp}$ is as before, $\overline{\mathtt{B}}$ are the outputs that have been emitted, and $\overline{\mathtt{C}}$ are the outputs still to be emitted. Since all the inputs in a gate have to be processed before the outputs are produced, we do not need to consider other forms. In both forms, the symbol $\hat{}$, called *gate pointer*, indicates the next operations (one forward and one backward)

that the gate can perform. A structure may be either a void structure $\mathbf{0}$, or a signal $u{:}\overline{a}$ denoting an elementary message a with an id u, or a gate g, or a parallel composition " | " that collects gates and signals and allow them to interact. A structure may also be $(\mathtt{new}\ x)\ \mathtt{S}$ that limits the scope of a name or id x to $\mathtt{S}$; x is said to be *bound* in $(\mathtt{new}\ x)\ \mathtt{S}$. This is the only binding operator in reversible structures.

For example, the *transducer* depicted in Figure 4 (leftmost structure) is defined by $u{:}\overline{a}\ \mid\ \hat{}a\,.\,v{:}\overline{b}$. This solution may evolve into $u{:}a\,.\,\hat{}v{:}\overline{b}$ by inputting the signal $u{:}\overline{a}$, see the middle structure in Figure 4. At this stage, a signal $v{:}\overline{b}$ may be emitted, thus becoming $u{:}a\,.\,v{:}\overline{b}\hat{}$ (rightmost structure in Figure 4) or may backtrack to $\hat{}a\,.\,v{:}\overline{b}$ by releasing the signal $u{:}\overline{a}$ (see the following semantics). Another example is a *sink gate*, such as $\hat{}a\,.\,b$, that collects signals (and, in a stochastic model, may hold them for a while). This gate may evolve into $u{:}a\,.\,\hat{}b$, and then may become $u{:}a\,.\,v{:}b\hat{}$.

We often abbreviate the parallel of $\mathtt{S}_i$ for $i \in I$, where I is a finite set, with $\prod_{i\in I}\mathtt{S}_i$. We write $(\mathtt{new}\ x_1, \cdots, x_n)\ \mathtt{S}$ for $(\mathtt{new}\ x_1)\cdots(\mathtt{new}\ x_n)\ \mathtt{S}$, $n \geq 0$, and sometimes we shorten $x_1, \cdots, x_n$ into $\widetilde{x}$. The *free names and ids* in $\mathtt{S}$, denoted $\mathtt{fn}(\mathtt{S})$, are the names and ids in $\mathtt{S}$ with a non-bound occurrence.

Structures we will never want to distinguish for any semantic reason are identified by a congruence. Let $\equiv$, called *structural congruence*, be the least congruence between structures containing alpha equivalence and satisfying the abelian monoid laws for parallel (associativity, commutativity and $\mathbf{0}$ as identity), and the scope laws

$$(\mathtt{new}\ x)\ \mathbf{0} \equiv \mathbf{0} \qquad (\mathtt{new}\ x)\ (\mathtt{new}\ x')\ \mathtt{S} \equiv (\mathtt{new}\ x')\ (\mathtt{new}\ x)\ \mathtt{S},$$

$$\mathtt{S}\ \mid\ (\mathtt{new}\ x)\ \mathtt{S}' \equiv (\mathtt{new}\ x)\ (\mathtt{S}\ \mid\ \mathtt{S}'), \quad \textit{if}\ x \notin \mathtt{fn}(\mathtt{S})$$

It is easy to demonstrate the following property.

Proposition 1 *For every* $\mathtt{S}$, $\mathtt{S} \equiv (\mathtt{new}\ \widetilde{x})\ (\prod_{i\in I} g_i \mid \prod_{j\in J} u_j{:}\overline{a_j})$. *The structure* $(\mathtt{new}\ \widetilde{x})\ (\prod_{i\in I} g_i \mid \prod_{j\in J} u_j{:}\overline{a_j})$, *which is unique up-to alpha equivalence, the order of names and ids in the sequence* $\widetilde{x}$, *and the order of gates and signals, is called the* normal form *of* $\mathtt{S}$.

The semantics of reversible structures is defined operationally by means of a reduction relation.

Definition 1 *The reduction relation of reversible structures is the least relation* $\longrightarrow$ *satisfying the axioms*

(*input capture*) $\quad u{:}\overline{a} \mid \mathtt{A}^{\perp} \,.\, \hat{}a \,.\, \mathtt{B} \,.\, \overline{\mathtt{C}} \longrightarrow \mathtt{A}^{\perp} \,.\, u{:}a \,.\, \hat{}\mathtt{B} \,.\, \overline{\mathtt{C}}$

(*input release*) $\quad \mathtt{A}^{\perp} \,.\, u{:}a \,.\, \hat{}\mathtt{B} \,.\, \overline{\mathtt{C}} \longrightarrow u{:}\overline{a} \mid \mathtt{A}^{\perp} \,.\, \hat{}a \,.\, \mathtt{B} \,.\, \overline{\mathtt{C}}$

(*output release*) $\quad \mathtt{A}^{\perp} \,.\, \overline{\mathtt{B}} \,.\, \hat{}u{:}\overline{a} \,.\, \overline{\mathtt{C}} \longrightarrow u{:}\overline{a} \mid \mathtt{A}^{\perp} \,.\, \overline{\mathtt{B}} \,.\, u{:}\overline{a} \,.\, \hat{}\overline{\mathtt{C}}$

(*output capture*) $\quad u{:}\overline{a} \mid \mathtt{A}^{\perp} \,.\, \overline{\mathtt{B}} \,.\, u{:}\overline{a} \,.\, \hat{}\overline{\mathtt{C}} \longrightarrow \mathtt{A}^{\perp} \,.\, \overline{\mathtt{B}} \,.\, \hat{}u{:}\overline{a} \,.\, \overline{\mathtt{C}}$

and closed under the rules

$$\frac{\mathtt{S} \longrightarrow \mathtt{S}'}{(\mathtt{new}\ a)\ \mathtt{S} \longrightarrow (\mathtt{new}\ a)\ \mathtt{S}'} \qquad \frac{\mathtt{S} \longrightarrow \mathtt{S}'}{\mathtt{S} \mid \mathtt{S}'' \longrightarrow \mathtt{S}' \mid \mathtt{S}''}$$

$$\frac{\mathtt{S}_1 \equiv \mathtt{S}'_1 \quad \mathtt{S}'_1 \longrightarrow \mathtt{S}'_2 \quad \mathtt{S}'_2 \equiv \mathtt{S}_2}{\mathtt{S}_1 \longrightarrow \mathtt{S}_2}$$

Sequences of reductions, called computations*, are noted* $\longrightarrow^*$.

The reductions (*input capture*) and (*output release*) are called *forward reductions*, the reductions (*input release*) and (*output capture*) are called *backward reductions*.

The axioms of reversible structures semantics are explained below by discussing the reductions of the transducer $\hat{}a \,.\, v{:}\overline{b}$ when exposed to signals $u{:}\overline{a}$ and $w{:}\overline{a}$. The transducer may behave either as $u{:}\overline{a} \mid w{:}\overline{a} \mid \hat{}a \,.\, v{:}\overline{b} \longrightarrow w{:}\overline{a} \mid u{:}a \,.\, \hat{}v{:}\overline{b}$ or as $u{:}\overline{a} \mid w{:}\overline{a} \mid \hat{}a \,.\, v{:}\overline{b} \longrightarrow u{:}\overline{a} \mid w{:}a \,.\, \hat{}v{:}\overline{b}$ according to whether the axiom (*input capture*) is instantiated either with the signal $u{:}\overline{a}$ or with $w{:}\overline{a}$ – in these cases $\mathtt{A}^{\perp}$ is empty. In turn, $w{:}\overline{a} \mid u{:}a \,.\, \hat{}v{:}\overline{b}$ may reduce with (*output release*) as $w{:}\overline{a} \mid u{:}a \,.\, \hat{}v{:}\overline{b} \longrightarrow w{:}\overline{a} \mid u{:}a \,.\, v{:}\overline{b}\hat{} \mid v{:}\overline{b}$ or may backtrack with (*input release*) as follows $w{:}\overline{a} \mid u{:}a \,.\, \hat{}v{:}\overline{b} \longrightarrow u{:}\overline{a} \mid w{:}\overline{a} \mid \hat{}a \,.\, v{:}\overline{b}$. This backtracking is always possible in our algebra. In fact, it is a direct consequence of the property that, for every axiom $\mathtt{S} \longrightarrow \mathtt{S}'$ of Definition 1, there is a "converse one" $\mathtt{S}' \longrightarrow \mathtt{S}$.

Proposition 2 *For any reduction* $\mathtt{S} \longrightarrow \mathtt{S}'$ *there exists a converse one* $\mathtt{S}' \longrightarrow \mathtt{S}$.

We notice that, $\hat{}a \,.\, u{:}\overline{b} \mid v{:}\overline{a} \mid \hat{}a \,.\, u{:}\overline{b} \equiv v{:}\overline{a} \mid \hat{}a \,.\, u{:}\overline{b} \mid \hat{}a \,.\, u{:}\overline{b}$ (and similarly for every permutation of gates and signals). In these structures,

the two occurrences of $\hat{}a \,.\, u{:}\overline{b}$ are indistinguishable, that is it is not possible to identify the precise gate $\hat{}a \,.\, u{:}\overline{b}$ that performs the reduction $\hat{}a \,.\, u{:}\overline{b} \mid v{:}\overline{a} \mid \hat{}a \,.\, u{:}\overline{b} \longrightarrow v{:}a \,.\, \hat{}u{:}\overline{b} \mid \hat{}a \,.\, u{:}\overline{b}$. This feature formalizes the well-mixing assumption of chemical solutions, namely that the probability of collision between two molecules is independent of their position. This is also the main difference between our model and reversible process calculi models as [5, 15], where every element has a unique tag. We finally notice that, as a consequence of the above identities, the notions of causality and independence of reductions are different from [5, 16] because different molecules of the same chemical species are indistinguishable in our setting.

By Proposition 1 and the definition of the reduction relation, it is possible to restrict the arguments about the dynamics of reversible structures to structures in normal forms. In turns, the following statement allows one to limit the analysis to the subclass of structures without `new`s when the interest is in computations of "closed" structures, namely structures that do not interact with the external environment. (This simplifies the following notion of weak coherence.)

Proposition 3 $(\texttt{new}\ \widetilde{x})\ (\prod_{i\in I} g_i \mid \prod_{j\in J} u_j{:}\overline{a_j}) \longrightarrow (\texttt{new}\ \widetilde{x})(\prod_{i\in I} g'_i \mid \prod_{j\in J'} u'_j{:}\overline{a'_j})$ *if and only if* $\prod_{i\in I} g_i \mid \prod_{j\in J} u_j{:}\overline{a_j} \longrightarrow \prod_{i\in I} g'_i \mid \prod_{j\in J'} u'_j{:}\overline{a'_j}$.

In the following, if not otherwise specified, the structures will be considered without `new`s.

Definition 2 *A structure* `S` *is* weak coherent *whenever ids are uniquely associated to names and co-names. That is, if* $u{:}\alpha$ *and* $u{:}\alpha'$ *occur in* `S` *then either* $\alpha = \alpha'$ *or* $\alpha = \overline{\alpha'}$.

For example, the structure $u{:}a \,.\, v{:}\overline{b}\hat{} \mid v{:}\overline{c}$ is not weak coherent because v is associated to two different co-names, while $u{:}a \,.\, v{:}\overline{b}\hat{} \mid v{:}\overline{b}$ is weak coherent. Weak coherence is an invariant of the reduction relation.

Proposition 4 *If* `S` *is weak coherent and* $\texttt{S} \longrightarrow \texttt{S}'$ *then* $\texttt{S}'$ *is weak coherent.*

4 The compilation of reversible structures into three-domains DNA strands

In this section we detail the implementation of weak coherent reversible structures into three-domains DNA strands. We have already presented these

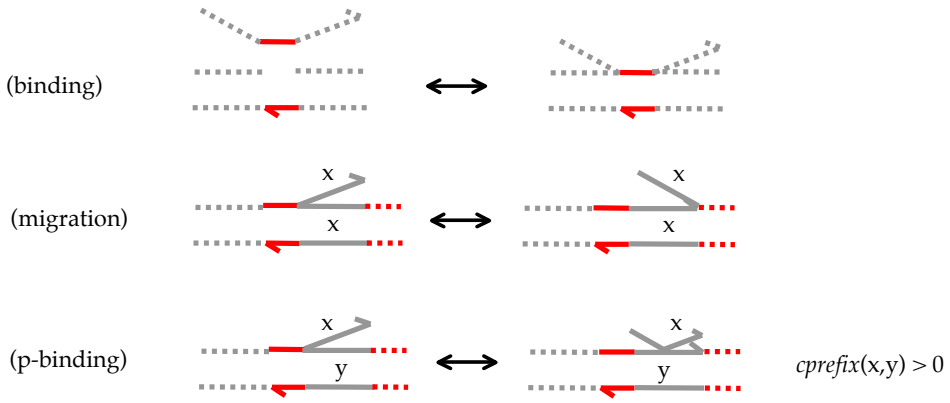


Figure 6: Dynamics of DNA strands

strands in Section 2. Here we complete the presentation by discussing the semantic rules in Figure 6.

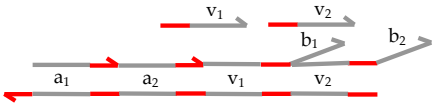
A DNA solution is a *multiset* of strands that may be either single two-domains toehold/long-domain or single three-domains long-domain/toehold/long-domain or double strands. The double strands are (i) composed of two single strands with opposite orientation, where the bottom strand is the complement of the top strand; (ii) are *toehold-mediated*, namely they are sequences of alternating toeholds and long-domains. We assume there is a unique toehold in strands; therefore complementary toeholds always match. The definition of DNA solution is purposely left informal because below we only consider a subclass of solutions implementing reversible structures.

The dynamics of DNA solutions is defined in Figure 6. Rule (*binding*) models toehold hybridizations between a double strand with an hole in the upper strand, in correspondence of a toehold and a signal. The dotted lines represent domains that may miss so, technically, the rule is a schema. Rule (*migration*) defines branch migrations of matching long-domains, while rule (*p-binding*) defines partial branch migrations of mismatching domains. In this latter case the bases may hybridize until the longest matching prefix of the domains and then may unbind. Formally, there is a function $\mathit{cprefix}(x, y)$ from pairs of names and ids to naturals that defines the largest matching prefix between them. This function is not defined on pairs of identical names or of identical ids. The rule (*p-binding*) is applied provided $\mathit{cprefix}(x, y)$ is positive.

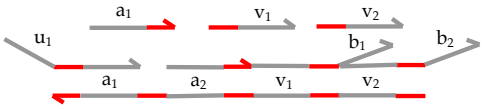
Without loss of generality, by Proposition 3, we restrict the following discussion to structures without `new`. The encoding $(\!|\ \cdot\ |\!)$ of reversible structures to DNA strands is homomorphic with respect to parallel, is such that $(\!|\ \mathbf{0}\ |\!) = \emptyset$, and it is defined on signals and gates as follows (for gates we only illustrate the encodings of configurations of $a_1 \,.\, a_2 \,.\, v_1{:}\overline{b_1} \,.\, v_2{:}\overline{b_2}$):

$(\!|\ u{:}\overline{a}\ |\!)$ =

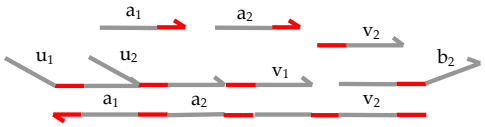
$(\!|\ \hat{}a_1 \,.\, a_2 \,.\, v_1{:}\overline{b_1} \,.\, v_2{:}\overline{b_2}\ |\!)$ =



$(\!|\ u_1{:}a_1 \,.\, \hat{}a_2 \,.\, v_1{:}\overline{b_1} \,.\, v_2{:}\overline{b_2}\ |\!)$ =



$(\!|\ u_1{:}a_1 \,.\, u_2{:}a_2 \,.\, v_1{:}\overline{b_1} \,.\, \hat{}v_2{:}\overline{b_2}\ |\!)$ =



The strict correspondence between reversible structures and DNA three-domains strands is fixed by the following statement. Let $\mathtt{T}$ and $\mathtt{T}'$ be two DNA solutions. We write $\mathtt{T} =_{b,p} \mathtt{T}'$ if and only if $\mathtt{T} \longrightarrow^* \mathtt{T}'$ with rules (binding) and (p-binding). By reversibility, the relation $=_{b,p}$ is symmetric.

Theorem 1 *$\mathtt{S} \longrightarrow \mathtt{S}'$ implies $(\!|\ \mathtt{S}\ |\!) \longrightarrow^* (\!|\ \mathtt{S}'\ |\!)$. Additionally, if $\mathtt{S}$ is weak coherent and $(\!|\ \mathtt{S}\ |\!) \longrightarrow^* \mathtt{T}$ then there is $\mathtt{S}'$ such that $\mathtt{T} =_{b,p} (\!|\ \mathtt{S}'\ |\!)$.*

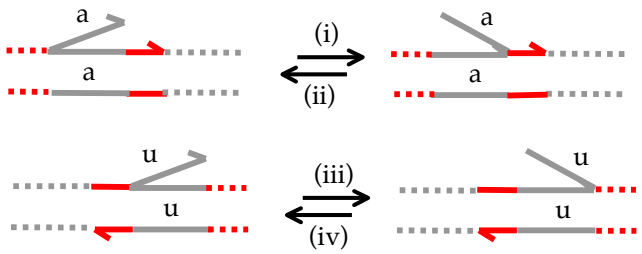
Proof: The proof of $\mathtt{S} \longrightarrow \mathtt{S}'$ implies $(\!|\ \mathtt{S}\ |\!) \longrightarrow^* (\!|\ \mathtt{S}'\ |\!)$ is a simple case analysis on the axiom used in $\mathtt{S} \longrightarrow \mathtt{S}'$. Figure 4 illustrates the correspondence when the axiom is an (*input capture*) (in the simple case of a transducer). The analysis of the other axioms is omitted.

Let $\mathtt{S}$ be weak coherent. The proof that $(\!|\ \mathtt{S}\ |\!) \longrightarrow^* \mathtt{T}$ then there is $\mathtt{S}'$ such that $\mathtt{T} =_{b,p} (\!|\ \mathtt{S}'\ |\!)$ is by induction on the number of rules (*migration*) used in $(\!|\ \mathtt{S}\ |\!) \longrightarrow^* \mathtt{T}$. The case when this number is 0 is obvious. Assume

the statement holds when there are n rules (*migration*), let us prove the case $n+1$. So the computation may be split into

$$(\!|\, \mathtt{S} \,|\!) \longrightarrow^* \mathtt{T}_1 \longrightarrow \mathtt{T}_2 \longrightarrow^* \mathtt{T}$$

where $\mathtt{T}_1 \longrightarrow \mathtt{T}_2$ is the $n+1$ reduction due to an instance of (*migration*) and $\mathtt{T}_2 =_{b,p} \mathtt{T}$. By inductive hypotheses, there is $\mathtt{T}'_1$ such that $\mathtt{T}_1 \longrightarrow^* \mathtt{T}'_1$ with rules that are instances of (*binding*) and (*p-binding*) and such that $\mathtt{T}'_1 = (\!|\, \mathtt{S}'_1 \,|\!)$, for some $\mathtt{S}'_1$. In case of strands obtained by the encoding $(\!|\, \cdot \,|\!)$ there are four possible types of reduction $\mathtt{T}_1 \longrightarrow \mathtt{T}_2$:



In case (i), according to the signals that are present in the solutions, the upper domain a must be the final part of a signal starting with an id, let's say u, and having a toehold in between – the signal is $u{:}a$. Next, take the reverse of $\mathtt{T}_1 \longrightarrow^* \mathtt{T}'_1$, let it be $\mathtt{T}'_1 \longrightarrow^* \mathtt{T}_1$, and consider the computation $\mathtt{T}'_1 \longrightarrow \mathtt{T}''_1 \longrightarrow \mathtt{T}'''_1$ where the first reduction is the (*binding*) of the toehold of $u{:}a$ and the second one is the (*migration*) of the domain a. By definition $\mathtt{T}'''_1 = (\!|\, \mathtt{S}' \,|\!)$ where $\mathtt{S}'_1 \longrightarrow \mathtt{S}'$ is an instance of (*input capture*). It remains to be proved that $(\!|\, \mathtt{S}' \,|\!) \longrightarrow^* \mathtt{T}_2$ and then, by reversibility, we obtain $\mathtt{T} =_{b,p} (\!|\, \mathtt{S}' \,|\!)$.

Consider the computation $(\!|\, \mathtt{S}' \,|\!) \longrightarrow^* \mathtt{T}'_2$ obtained by performing the sequence of reductions of $\mathtt{T}'_1 \longrightarrow^* \mathtt{T}_1$ and skipping those concerning the signal $u{:}a$ and the toehold of the double strand involved in $\mathtt{T}'_1 \longrightarrow \mathtt{T}''_1 \longrightarrow \mathtt{T}'''_1$. We observe that

1. in $\mathtt{T}'_2$ the context of the signal $u{:}a$ and the toehold of the double strand involved in $\mathtt{T}'_1 \longrightarrow \mathtt{T}''_1 \longrightarrow \mathtt{T}'''_1$ are the same as in $\mathtt{T}_2$ because reductions of DNA strands are context-free;

2. the configurations signal $u{:}a$ and the toehold of the double strand involved in $\mathtt{T}'_1 \longrightarrow \mathtt{T}''_1 \longrightarrow \mathtt{T}'''_1$ are the same in $\mathtt{T}'_2$ and $\mathtt{T}_2$ because (i) $\mathtt{T}'_1 \longrightarrow^* \mathtt{T}_1$, by definition of $\mathtt{T}'_1$, must have an odd number of (*binding*) rules concerning the toehold of the signal $u{:}a$ since the toehold initially

is unbound and at the end is bound (otherwise the rule (*migration*) cannot occur); (ii) the effects of reverse reductions on a solution are void.

Therefore $\mathtt{T}'_2 = \mathtt{T}_2$.

The other cases are omitted because similar; we only observe that (ii) corresponds to an instance of (*input release*, (iii) corresponds to an instance of (*output release*), and (iv) corresponds to an instance of (*output capture*). ■

The second part of Theorem 1 is restricted to weak coherent structures. In fact, $(\!|\ \mathtt{S}\ |\!) \longrightarrow^* (\!|\ \mathtt{S}'\ |\!)$ implies $\mathtt{S} \longrightarrow^* \mathtt{S}'$ is false in the unrestricted case. Consider the above encoding of the gate $u_1{:}a_1 \,.\, u_2{:}a_2 \,.\, v_1{:}\overline{b_1} \,.\, \hat{}\, v_2{:}\overline{b_2}$ and observe that, in the DNA strand, the co-name $\overline{b}_1$ never appears. If the (not weak coherent) structure also contained the signal $v_1{:}\overline{c}$ then the DNA strand might reduce to the encoding of $u_1{:}a_1 \,.\, u_2{:}a_2 \,.\, \hat{}\, v_1{:}\overline{c} \,.\, v_2{:}\overline{b_2}$. However this gate cannot be obtained from the structure $u_1{:}a_1 \,.\, u_2{:}a_2 \,.\, v_1{:}\overline{b_1} \,.\, \hat{}\, v_2{:}\overline{b_2}$. It is worth noticing that there is a way for removing the constraint of weak coherence in Theorem 1. The technique associates different toeholds to different names – that is the toeholds record the identity of names. While this method works fine for solutions with few names, it is not practicable in general because toeholds are usually very few with respect to (long domains) names. We also notice that the three-domains structures used by Quian and Winfree [17] have the output parts of gates without ids. It is easy to verify that such a model identifies more computations (*i.e.* mixes causal dependencies) than our model.

We finally remark that Theorem 1 establishes a relation between "meaningful" reductions of the DNA solution and reductions of reversible structures, where "meaningful" means those reductions obtained with rules (*migration*). In facts, rules (*binding*) and (*p-binding*) must be considered as bureaucracies used to prepare the solution for long-domains hybridizations.

5 Modelings

Despite their simplicity, reversible structures are quite expressive. In this section we discuss a number of synchronization patterns that are standard in process calculi and detail their modeling in reversible structures. As we will see, in every case, reversibility plays a basic role.

Join patterns. Join patterns have been introduced in join-calculus [7]. Here we consider the so-called "zero-adic" version where channels carry no message. Join patterns, written $a_1\&\cdots\&a_m \triangleright \overline{b_1}|\cdots|\overline{b_n}$ reduce as follows

$$\overline{a_1} \mid \cdots \mid \overline{a_m} \mid a_1\&\cdots\&a_m \triangleright \overline{b_1}|\cdots|\overline{b_n} \quad \longrightarrow \quad \overline{b_1} \mid \cdots \mid \overline{b_n}$$

that is a join pattern $a_1\&\cdots\&a_m$ triggers provided all the messages $\overline{a_1}$, $\cdots$, $\overline{a_m}$ are available in the solution. The rule specifies that all the messages are grabbed at once – an *all-or-nothing* requirement – and all the messages $\overline{b_1}$, $\cdots$, $\overline{b_n}$ are released at once.

The modeling of the above join pattern in reversible structures is given by the term

$$(\mathtt{new}\ u_1, \cdots, u_n)\ (\hat{}a_1\,.\,\cdots\,.\,a_m\,.\,u_1{:}\overline{b_1}\,.\,\cdots\,.\,u_n{:}\overline{b_n})$$

There is a difference between the semantics of this term and that of join patterns that turns out to be irrelevant. In the above terms, messages $\overline{a_1}$, $\cdots$, $\overline{a_m}$ are taken in order: first a_1, then a_2, and so on. It is possible that one takes a_1 and a_2 and then realizes that there is no a_3 in the solution – a circumstance that never occurs in join-calculus. However this is not an issue because (i) the input capture is reversible, therefore the messages a_1 and a_2 may be released in the solution, and (ii) messages/signals are asynchronous – they have no continuation – and therefore no (continuation) process has been triggered. Additionally, because of asynchrony, releasing messages $\overline{b_1}$, $\cdots$, $\overline{b_n}$ in order or all at once is semantically the same.

Mixed choice. Mixed choice is a standard operation in process calculi (see, for instance, CCS [12] or pi calculus [13, 14]) that allows the progress of exactly one among several processes. The operator, in case of asynchronous calculi, is usually written $\sum_{i\in I} a_i.\overline{b_i} + \sum_{j\in J} \overline{c_j}$ and the semantics is defined by one of the following rules:

$$\overline{a_k} \mid \textstyle\sum_{i\in I} a_i.\overline{b_i} + \sum_{j\in J} \overline{c_j} \quad \longrightarrow \quad \overline{b_k} \qquad (k \in I)$$

$$\overline{c}_k \mid \textstyle\sum_{i\in I} a_i.\overline{b_i} + \sum_{j\in J} \overline{c_j} \quad \longrightarrow \quad \mathbf{0} \qquad (k \in J)$$

In particular, the semantics excludes communications between two branches of the choice. The above mixed choice is modelled in reversible structures by the term

$$(\mathtt{new}\ v, e, {u_i}^{i\in I}, {w_j}^{j\in J})\ (\prod_{i\in I} e\,.\,a_i\,.\,u_i{:}\overline{b_i} \ \mid\ \prod_{j\in J} e\,.\,w_j{:}\overline{c_j} \ \mid\ v{:}\overline{e}\ )$$

that implements choice with parallel composition. Every gate of the above term is prefixed by an input on the name e that is local to the term. The correctness of the modeling follows by the properties (i) there is at most one gate that progresses because of the presence of exactly one signal $v{:}\overline{e}$; (ii) if the chosen gate is one of $e \,.\, a_i \,.\, u_i{:}\overline{b_i}$ and the solution does not contain a signal $\overline{a_i}$, it is possible to revert the decision and select another branch.

Smooth orchestrators. Smooth orchestrators, introduced in [10] for modeling synchronization patterns in web services, combine join patterns and (input-guarded) choices. A smooth orchestrator is a term $\sum_{i\in I} a_1^i \& \cdots \& a_{m_i}^i \triangleright \overline{b_1^i} | \cdots | \overline{b_{n_i}^i}$ with the semantics defined by ($k \in I$)

$$\overline{a}_1^k \mid \cdots \mid \overline{a}_{m_k}^k \mid \sum_{i\in I} a_1^i \& \cdots \& a_{m_i}^i \triangleright \overline{b_1^i} | \cdots | \overline{b_{n_i}^i} \quad \longrightarrow \quad \overline{b_1^k} \mid \cdots \mid \overline{b_{n_k}^k}$$

The modeling of such operator in reversible structures is a byproduct of the above encodings:

$$(\mathtt{new}\ v, e, u_1^i, \cdots, u_{n_i}^{i\ i\in I})\ (\prod_{i\in I} e \,.\, a_1^i \,.\, \cdots \,.\, a_{m_i}^i \,.\, u_1^i{:}\overline{b_1^i} \,.\, \cdots \,.\, u_{n_i}^i{:}\overline{b_{n_i}^i} \mid v{:}\overline{e}\ )$$

and correctness follows with arguments similar to the ones above.

6 The encoding of asynchronous RCCS into reversible structures

In this section we give a precise assessment of the expressive power of reversible structures, by discussing the encoding of a process calculus with a reversible transition relation: the *asynchronous* RCCS [5, 6]. As a consequence, it is possible establish properties of asynchronous RCCS using those of reversible structures. (See for example the Standardization Theorem in [4], which has been proved for RCCS in [5].)

The syntax of asynchronous RCCS uses an infinite set of *names*, ranged over by $a, b, c, \cdots$, and a disjoint set of *co-names*, ranged over by $\overline{a}, \overline{b}, \overline{c}, \cdots$. Names and co-names are ranged over by $\alpha, \beta, \cdots$ and are generically called *actions*. *Processes* P,, Q, $\cdots$ are defined by the following grammar (the set I is always finite):

$$P \ ::= \quad \mathbf{0} \quad | \quad \textstyle\sum_{i\in I} \alpha_i.P_i \quad | \quad \prod_{i\in I} P_i \quad | \quad (\mathtt{new}\ a)\ P$$

The term $\mathbf{0}$ defines the terminated process; $\sum_{i\in I}\alpha_i.P_i$ defines a process that may perform one action α_i and continues as P_i; $\prod_{i\in I}P_i$ defines the parallel composition of processes P_i; finally the term $(\texttt{new}\ a)\ P$ defines a name with scope P. Processes meet the following well-formed conditions:

- in $\overline{a}.P$, the process P is $\mathbf{0}$ (continuations of co-names are empty);
- in $\prod_{i\in I}P_i$ the processes P_i are guarded choices.

The semantics of asynchronous RCCS is defined in terms of a *transition relation* that uses

- *memories* m:

$$m \ ::= \ \langle\ \rangle \ \mid \ \langle i\rangle_n \bullet m \ \mid \ \langle m, \alpha, P\rangle \bullet m$$

- *run-time processes* R:

$$R \ ::= \ m \triangleright P \ \mid \ R \mid R \ \mid \ (\texttt{new}\ a)\ R$$

- *structural congruence* $\equiv$, defined in the standard way (see Section 3), plus the rules

$$m \triangleright (\textstyle\prod_{i\in 1..n} P_i) \ \equiv \ \prod_{i\in 1..n}\langle i\rangle_n \bullet m \triangleright P_i$$

$$m \triangleright (\texttt{new}\ a)\ P \ \equiv \ (\texttt{new}\ a)\ (m \triangleright P) \qquad (a \notin \mathsf{fn}(m))$$

The reduction relation $\longrightarrow$ is the least relation on run-time processes satisfying the axioms:

- $m \triangleright (a.P + Q) \mid m' \triangleright (\overline{a} + R) \longrightarrow \langle m', a, Q\rangle \bullet m \triangleright P \mid \langle m, \overline{a}, R\rangle \bullet m' \triangleright \mathbf{0}$,
- $\langle m', a, Q\rangle \bullet m \triangleright P \mid \langle m, \overline{a}, R\rangle \bullet m' \triangleright \mathbf{0} \longrightarrow m \triangleright (a.P + Q) \mid m' \triangleright (\overline{a} + R)$,

and closed under the contextual rules for parallel, new and structural congruence.

Our encoding (whose main ideas have been already discussed in Section 5) uses *environments* Γ that map memories to signals $u{:}\overline{c}$ such that:

1. (Γ is injective) if $m \neq m'$ then the signals $\Gamma(m)$ and $\Gamma(m')$ are different (they have different ids and co-names);
2. (Γ is suffix-closed) if $s \bullet m \in \texttt{dom}(\Gamma)$ then $m \in \texttt{dom}(\Gamma)$;

3. (Γ is branch-closed) if $\langle i\rangle_n \bullet m \in \mathtt{dom}(\Gamma)$ then $\langle 1\rangle_n \bullet m, \cdots, \langle n\rangle_n \bullet m \in \mathtt{dom}(\Gamma)$;

4. (Γ is depth-closed) if $\langle m', \alpha, P\rangle \bullet m \in \mathtt{dom}(\Gamma)$ then $m' \in \mathtt{dom}(\Gamma)$.

Let $\mathsf{fn}(\Gamma) \stackrel{\text{def}}{=} \{a \mid \exists\, m, u.\ \Gamma(m) = u{:}\overline{a}\}$.

Let Γ be an environment such that, for every $i \in I$, $\Gamma(m_i) = u_i{:}\overline{c_i}$, for some u_i. The encoding $[\![\cdot]\!]^\Gamma$ uses a number of fresh names and fresh ids and is defined by

$$[\![\textstyle\prod_{i\in I} m_i \triangleright P_i]\!]^\Gamma \;=\; \prod_{i\in I}([\![P_i]\!]_{c_i} \;\mid\; \Gamma(m_i)) \;\mid\; \mathcal{U}(\{m_i \mid i \in I\}, \Gamma)$$

where P_i are guarded choices and where the auxiliary functions $[\![P]\!]_c$ and $\mathcal{U}(M, \Gamma)$ are defined in Figure 7.

Lemma 1 *If $R \longrightarrow R'$ then, for suitable Γ and Γ', $[\![R]\!]^\Gamma \longrightarrow^* [\![R']\!]^{\Gamma'}$, with $id(\mu_i) \cap id(\Gamma(m) \mid \Gamma(m')) \neq \emptyset$.*

Proof: Without loss of generality, let

$$R = m \triangleright \sum_{i\in I} a_i.P_i + \sum_{j\in J} \overline{a_j} \;\mid\; m' \triangleright \sum_{i\in I'} b_i.Q_i + \sum_{j\in J'} \overline{b_j} \;\mid\; m'' \triangleright P''$$

and let $\overline{a_h} = b_k = \overline{a}$ and $P' = \sum_{i\in I\setminus\{h\}} a_i.P_i + \sum_{j\in J} \overline{a_j}$ and $Q' = \sum_{i\in I'} b_i.Q_i + \sum_{j\in J'\setminus\{k\}} \overline{b_j}$. Then $R \longrightarrow \langle m', a, P'\rangle \bullet m \triangleright P_h \;\mid\; \langle m, \overline{a}, Q'\rangle \bullet m' \triangleright \mathbf{0} \;\mid\; m'' \triangleright P''$.

Let Γ be such that $\Gamma(m) = w{:}\overline{c}$, $\Gamma(m') = w'{:}\overline{c'}$ and $\Gamma(m'') = w''{:}\overline{c''}$.

$$
\begin{array}{rcl}
[\![\mathbf{0}]\!]_c & = & \hat{}\,c \\[2ex]
[\![\sum_{i\in I} a_i.P_i + \sum_{j\in J} \overline{a_j}]\!]_c & = & \prod_{i\in I}(\hat{}\,c\,.\,a_i\,.\,u_i{:}\overline{c_i} \;\mid\; [\![P_i]\!]_{c_i}) \;\mid\; \prod_{j\in J} \hat{}\,c\,.\,u_j{:}\overline{a_j} \\
 & & \text{where } {c_i}^{i\in I}, {u_\ell}^{\ell\in I\cup J} \text{ are fresh} \\[2ex]
[\![\prod_{i\in 1..n} P_i]\!]_c & = & \hat{}\,c\,.\,u_1{:}\overline{c_1}\,.\,\cdots\,.\,u_n{:}\overline{c_n} \mid \prod_{i\in 1..n}[\![P]\!]_{c_i} \\
 & & \text{where } c_i, {u_i}^{i\in 1..n} \text{ are fresh} \\[2ex]
[\![(\mathtt{new}\ a)\ P]\!]_c & = & (\mathtt{new}\ a)\ ([\![P]\!]_c)
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{U}(\{\langle 1\rangle_n \bullet m, \cdots, \langle n\rangle_n \bullet m\} \uplus M, \Gamma) & = & u{:}c\,.\,v_1{:}\overline{c_1}\,.\,\cdots\,.\,v_n{:}\overline{c_n}\hat{} \\
 & & \mid \mathcal{U}(\{m\} \uplus M, \Gamma) \\
 & & \text{where } \Gamma(m) = u{:}\overline{c} \\
 & & \text{and } \Gamma(\langle i\rangle_n \bullet m) = v_i{:}\overline{c_i} \\[2ex]
\mathcal{U}(\{\langle m', a, P\rangle \bullet m, \langle m, \overline{a}, Q\rangle \bullet m'\} \uplus M, \Gamma) & = & u{:}c\,.\,w{:}a\,.\,v{:}\overline{c_1}\hat{} \;\mid\; [\![P]\!]_c \\
 & & \mid\; u'{:}c'\,.\,w{:}\overline{a}\hat{} \;\mid\; [\![Q]\!]_{c'} \\
 & & \mid \mathcal{U}(\{m, m'\} \uplus M, \Gamma) \\
 & & \text{where } \Gamma(m) = u{:}\overline{c} \\
 & & \text{and } \Gamma(m') = u'{:}\overline{c'} \\
 & & \text{and } \Gamma(\langle m', a, P\rangle \bullet m) = v{:}\overline{c_1} \\
 & & \text{and } w \text{ fresh}
\end{array}
$$

Figure 7: The functions $[\![P]\!]_c$ and $\mathcal{U}(M, \Gamma)$

Then

$$
\begin{array}{rcl}
[\![R]\!]^{\Gamma} & \longrightarrow & w{:}c \,.\, \hat{}\, a \,.\, u_k{:}\overline{c_k} \mid [\![P_k]\!]_{c_k} \mid \prod_{i\in I\cup J\setminus\{h\}} [\![P_i]\!]_c \\
& & \quad \mid w'{:}\overline{c'} \mid \prod_{i\in I'\cup J'} [\![Q_i]\!]_{c'} \\
& & \quad \mid w''{:}\overline{c''} \mid [\![P'']\!]_{c''} \mid \mathcal{U}(\{m,m',m''\},\Gamma) \\
& \longrightarrow & w{:}c \,.\, \hat{}\, a \,.\, u_k{:}\overline{c_k} \mid [\![P_k]\!]_{c_k} \mid \prod_{i\in I\cup J\setminus\{h\}} [\![P_i]\!]_c \\
& & \quad \mid w'{:}c' \,.\, \hat{}\, v'_h{:}\overline{a} \mid \prod_{i\in I'\cup J'\setminus\{h\}} [\![Q_i]\!]_{c'} \\
& & \quad \mid w''{:}\overline{c''} \mid [\![P'']\!]_{c''} \mid \mathcal{U}(\{m,m',m''\},\Gamma) \\
& \longrightarrow & w{:}c \,.\, \hat{}\, a \,.\, u_k{:}\overline{c_k} \mid [\![P_k]\!]_{c_k} \mid \prod_{i\in I\cup J\setminus\{h\}} [\![P_i]\!]_c \\
& & \quad \mid v'_h{:}\overline{a} \mid w'{:}c' \,.\, v'_h{:}\overline{a} \,.\, \hat{} \mid \prod_{i\in I'\cup J'\setminus\{h\}} [\![Q_i]\!]_{c'} \\
& & \quad \mid w''{:}\overline{c''} \mid [\![P'']\!]_{c''} \mid \mathcal{U}(\{m,m',m''\},\Gamma) \\
& \longrightarrow & w{:}c \,.\, v'_h{:}a \,.\, \hat{}\, u_k{:}\overline{c_k} \mid [\![P_k]\!]_{c_k} \mid \prod_{i\in I\cup J\setminus\{h\}} [\![P_i]\!]_c \\
& & \quad \mid w'{:}c' \,.\, v'_h{:}\overline{a} \,.\, \hat{} \mid \prod_{i\in I'\cup J'\setminus\{h\}} [\![Q_i]\!]_{c'} \\
& & \quad \mid w''{:}\overline{c''} \mid [\![P'']\!]_{c''} \mid \mathcal{U}(\{m,m',m''\},\Gamma) \\
& \longrightarrow & u_k{:}\overline{c_k} \mid [\![P_k]\!]_{c_k} \mid w{:}c \,.\, v'_h{:}a \,.\, u_k{:}\overline{c_k}\hat{} \\
& & \quad \mid \prod_{i\in I\cup J\setminus\{h\}} [\![P_i]\!]_c \\
& & \quad \mid w'{:}c' \,.\, v'_h{:}\overline{a} \,.\, \hat{} \mid \prod_{i\in I'\cup J'\setminus\{h\}} [\![Q_i]\!]_{c'} \\
& & \quad \mid w''{:}\overline{c''} \mid [\![P'']\!]_{c''} \mid \mathcal{U}(\{m,m',m''\},\Gamma) \\
& = & [\![R']\!]^{\Gamma'}
\end{array}
$$

where $\Gamma' = \Gamma[\langle m', a, P'\rangle \bullet m \mapsto u_k{:}\overline{c_k} \;;\; \langle m, \overline{a}, Q'\rangle \bullet m' \mapsto v''_h{:}\overline{c_h}]$. Since our structures and asynchronous RCCS are both reversible, the above computation also demonstrates that $R' \longrightarrow R$ implies $[\![R']\!]^{\Gamma'} \longrightarrow^* [\![R]\!]^{\Gamma}$. ■

It is worth observing that, in the above encoding, there is a strict correspondence between *reversibility of our structures and reversibility in asynchronous* RCCS. To formalize this correspondence, let an occurrence of an id u be *positive* in a structure $\mathtt{S}$ if u occurs in a signal or in a gate $\mathtt{A}^{\perp} \,.\, \overline{\mathtt{B}} \,.\, \hat{}\,\overline{\mathtt{C}}$ or $\mathtt{A}^{\perp} \,.\, \hat{}\,\mathtt{B} \,.\, \overline{\mathtt{C}}$ in the $\mathtt{A}^{\perp}$ sequence or in the $\overline{\mathtt{C}}$ sequence. The occurrence of u is *negative* if it is in the $\overline{\mathtt{B}}$ sequence of a gate $\mathtt{A}^{\perp} \,.\, \overline{\mathtt{B}} \,.\, \hat{}\,\overline{\mathtt{C}}$. Let the *type of* g, written *type*(g), be the sequence of ids of co-names in g. For example *type*$(v{:}a \,.\, \hat{}\, a \,.\, u{:}\overline{a} \,.\, w{:}\overline{c}) = uw$ (as usual, dots are omitted in sequences of ids).

Definition 3 *A weak coherent structure* $\mathtt{S}$ *is* coherent *whenever*

– *different gates in* $\mathtt{S}$ *have types with no id in common;*

– *ids occur at most twice: one occurrence is positive and the other is negative.*

Proposition 5 *1. If* $\mathtt{S}$ *is coherent and* $\mathtt{S} \longrightarrow \mathtt{S}'$ *then* $\mathtt{S}'$ *is coherent.*

2. $[\![\prod_{i\in I} m_i \triangleright P_i]\!]^\Gamma$*, where* Γ *is an environment as discussed above and* P_i *are guarded choices, is a coherent structure.*

The coherence of $[\![\prod_{i\in I} m_i \triangleright P_i]\!]^\Gamma$ and the properties of Γ have an immediate consequence: the terms that are desynchronized are exactly those implementing processes that actually interacted in the past. Said more technically, in the proof of Lemma 1, there is a unique way to desynchronize the process $[\![P_k]\!]_{c_k}$ in the solution $[\![R']\!]^{\Gamma'}$, namely undoing exactly the steps until the solution $[\![R]\!]^\Gamma$. No other signal/gate may interfere with these steps.

We finally observe that the reverse encoding of $[\![\cdot]\!]^\Gamma$ seems impossible (or at least we do not have any solution at present) because reversible structures are not asynchronous. In facts, sequences of outputs in our gates are not encodable in asynchronous RCCS.

7 Conclusion

In this paper we have introduced reversible structures, an algebra for massive concurrent systems, where terms retain bits of causal dependencies that allow one to reverse computation histories. We have discussed the model that has inspired reversible structures, the DNA three-domains strands – and studied the implementation of (weak coherent) reversible structures in DNA strands. We have finally analyzed significant synchronization patterns of process algebra and the modeling schemas into reversible structures.

In the companion paper [4] we develop the theory of causal dependencies in reversible structures. Following Lévy [11], we define an equivalence on computations based on labels of terms that abstracts away from the order of causally independent reductions – the *permutation equivalence*. We then demonstrate a standardization theorem that permits shortening of computations by removing converse reductions. This theorem seems strange because, in reversible structures, removals may address reverse reductions performed by different terms (of same species). In facts, in our setting, labels are not powerful enough to discriminate among molecules of the same

species. We finally study *coherent* reversible structures that have been defined in Section 6. We demonstrate that the reachability problem in coherent structures has a computational complexity that is quadratic with respect to the size of the structures, a problem that is EXPSPACE-complete in weak coherent structures.

Our study prompts a thorough analysis of reversible calculi where processes have multiplicities and the causal dependencies between copies may be exchanged. Open questions are (i) What synchronization schemas can be programmed in massive concurrent systems? (ii) Are there other constraints, different than coherence, such that relevant bio-chemical properties retain better algorithms than in standard structures? (iii) What is the theory of massive (reversible) systems *with irreversible operators* and what is the relationship with standard programming languages?

References

[1] G. Boudol and I. Castellani. Permutation of transitions: An event structure semantics for CCS and SCCS. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 411–427. Springer, 1989.

[2] L. Cardelli. Strand algebras for DNA computing. In *DNA 2009*, volume 5877 of *Lecture Notes in Computer Science*, pages 12–24, 2009.

[3] L. Cardelli. Two-domain DNA strand displacement. In *Developments in Computational Models (DCM 2010)*, volume 25 of *EPTCS*, pages 33–47, 2010.

[4] L. Cardelli and C. Laneve. Reversible structures. In *Proceedings of 9th Int. Conf. on Computational Methods in Systems Biology (CMSB 2011)*, ACM Digital Library, 2011.

[5] V. Danos and J. Krivine. Reversible communicating systems. In *CONCUR 2004*, volume 3170 of *Lecture Notes in Computer Science*, pages 292–307, 2004.

[6] V. Danos and J. Krivine. Transactions in RCCS. In *CONCUR 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2005.

[7] C. Fournet and G. Gonthier. The reflexive cham and the join-calculus. In *Proc. of the 23rd Symposium on Principles of Programming Languages (POPL '96)*, pages 372–385. ACM, 1996.

[8] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem*, 81:2340–2361, 1977.

[9] I. Lanese, C. A. Mezzina, and J.-B. Stefani. Reversing higher-order pi. In *Proceedings of CONCUR 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 478–493. Springer, 2010.

[10] C. Laneve and L. Padovani. Smooth orchestrators. In *Foundations of Software Science and Computation Structures, 9th Int. Conf (FOSSACS 2006)*, volume 3921 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2006.

[11] J.-J. Lévy. An algebraic interpretation of the *lambda-beta-k*-calculus; and an application of a labelled *lambda* -calculus. *Theor. Comput. Sci.*, 2(1):97–114, 1976.

[12] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[13] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, i and ii. *Inf. Comput.*, 100(1):1–77, 1992.

[14] U. Nestmann and B. C. Pierce. Decoding choice encodings. *Inf. Comput.*, 163(1):1–59, 2000.

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

[16] I. Phillips and I. Ulidowski. Reversing algebraic process calculi. *J. Log. Algebr. Program.*, 73(1-2):70–96, 2007.

[17] L. Qian and E. Winfree. A simple dna gate motif for synthesizing large-scale circuits. In *14th International Meeting on DNA Computing*, volume 5347 of *Lecture Notes in Computer Science*, pages 70–89. Springer, 2008.

[18] D. Y. Zhang and E. Winfree. Control of dna strand displacement kinetics using toehold exchange. *Journal of the American Chemical Society*, 131(47):17303–17314, 2009.