Theoretical Computer Science

A process algebraic approach to reaction systems.

--Manuscript Draft--

Manuscript Number:	TCS-D-20-00267R2
Article Type:	SI: RS
Section/Category:	C - Theory of natural computing
Keywords:	Process algebras; reaction systems; Assertion language; Hennessy-Milner logic.
Corresponding Author:	Moreno Falaschi University of Siena Siena, ITALY
First Author:	Linda Brodo, Researcher
Order of Authors:	Linda Brodo, Researcher
	Roberto Bruni, Professor
	Moreno Falaschi, Professor
Manuscript Region of Origin:	ITALY
Abstract:	In the area of Natural Computing, Reaction Systems (RSs) are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. RSs interact with a context, and pose challenges for modularity, compositionality, extendibility and behavioural equivalence. In this paper we define a modular encoding of RSs as processes in the chained Core Network Algebra (cCNA), which is a new variant of the link-calculus. The encoding represents the behaviour of each entity separately and preserves faithfully their features, and we prove its correctness and completeness. Our encoding provides a Labelled Transition System (LTS) semantics for RSs. Based on the LTS semantics, we adapt the classical notion of bisimulation to define a novel equivalence, called bio-similarity, for studying properties of RSs. In particular, we define a new assertion language based on regular expressions, which allows us to specify the properties of interest, and use it to extend Hennessy-Milner logic to our setting. We prove that our bio-similarity relation and the logical equivalence, that are defined parametrically on some assertion of interest, coincide. Finally, we claim that our encoding contributes to increase the expressiveness of RSs, by exploiting the interaction among different RSs.

Dear Editor and Referees,

we thank you very much for your detailed comments and remarks, which greatly helped us to correct and improve our presentation.
We updated the manuscript following the advices of the referees. Please find below a detailed description of the way in which
the items pointed out by the referees were addressed in the revised version.

Best regards,
Linda Brodo, Roberto Bruni, Moreno Falaschi

Manuscript Number: TCS-D-20-00267R1
Manuscript Title: A process algebraic approach to reaction systems.
Author(s): Linda Brodo, Researcher; Roberto Bruni, Professor; Moreno Falaschi, Professor
Submitted to: Theoretical Computer Science

Reviewer #1: I think my original comments have been addressed satisfactorily by the authors, but I still have some remarks:

p.6, l.12 - replace "set of entities" by "sets of entities"
> OK. Done

p.7, l.42 - replace "where v_i is a link chain" by "where v is a link chain"
as in the syntax from the previous line you use v.P and not v_i.P
> OK. Done

p.8, l.13, (Res) rule - make sure all a's have same font as it seems the one in (\nu a)P' is different
> OK. Done

p.8, l.30 - replace "Fig.1" by "Fig. 1"
> OK. Done

p.12, l.35 (item 2 of Lemma 12) - replace "for each reaction" by "each reaction"
> OK. We have corrected the formulation.

p.12, l.51 - in Def. 13 shouldn't be P \xrightarrow{(\nu names)v} P' instead of P \xrightarrow{v}P' ? Only starting on p.15, l.52 you say "In this section, for simplicity, we shall often omit topmost restrictions (\nu names)". Thus, either move this assumption earlier or till Section 5 keep using (\nu names) on \xrightarrow
> No, Def.13 is correct: you can imagine that P contains no restrictions and therefore P \xrightarrow{v}P' is such that all the names in v are visible. Then (\nu names)P \xrightarrow{(\nu names)v} (\nu names)P' but in the label (\nu names)v all the names are transformed to silent actions \tau.

p.13, l.56 - (\nu names) missing from \xrightarrow label
> No, see above.

p.17, l.49 - replace "(starting at cxt and ending at p1" by "(starting at cxt and ending at p1)", namely add missing bracket
> OK. Done

p.33, l.41 - shouldn't be "j \geq 1" (as assumed on p.33, l.38) instead of "j \geq 0"?
> OK. Done

p.33, l.47 and l.49 - in the proof of the base case "j=1", instead of "j-1" shouldn't be "j"? This is due to the fact that Prop. 16 you mention you have the case j=1 and also on p.33 on l.52 you got \gamma^j and not \gamma^{j-1}.
> We imagine you meant "in the proof of the inductive case". You are right, we have slightly expanded the proof to make it clear the application of the inductive hypothesis.

Revised submission (PDF)

A process algebraic approach to reaction systems ☆

Linda Brodo[b], Roberto Bruni[a], Moreno Falaschi[c]

[a] Dipartimento di Informatica, Università di Pisa, Italy
[b] Dipartimento di Scienze Economiche e Aziendali, Università di Sassari, Italy
[c] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Univ. di Siena, Italy

Abstract

In the area of Natural Computing, Reaction Systems (RSs) are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. RSs interact with a context, and pose challenges for modularity, compositionality, extendibility and behavioural equivalence. In this paper we define a modular encoding of RSs as processes in the chained Core Network Algebra (cCNA), which is a new variant of the `link`-calculus. The encoding represents the behaviour of each entity separately and preserves faithfully their features, and we prove its correctness and completeness. Our encoding provides a Labelled Transition System (LTS) semantics for RSs. Based on the LTS semantics, we adapt the classical notion of bisimulation to define a novel equivalence, called bio-similarity, for studying properties of RSs. In particular, we define a new assertion language based on regular expressions, which allows us to specify the properties of interest, and use it to extend Hennessy-Milner logic to our setting. We prove that our bio-similarity relation and the logical equivalence, that are defined parametrically on some assertion of interest, coincide. Finally, we claim that our encoding contributes to increase the expressiveness of RSs, by exploiting the interaction among different RSs.

Keywords: process algebras, Reaction Systems, assertion language, HM-logic

1. Introduction

Natural Computing is an active area of research which builds on two main aspects: human designed computing inspired by nature, and computation performed in nature. Reaction Systems (RSs) [1] are a rewriting formalism inspired by the way biochemical reactions take place in living cells. This theory has already shown to be relevant in several different fields, including biology [2, 3, 4, 5]

☆Research partially supported by Università degli Studi di Sassari *Fondi di Ateneo per la ricerca 2019*.

Email addresses: `brodo@uniss.it` (Linda Brodo), `bruni@di.unipi.it` (Roberto Bruni), `moreno.falaschi@unisi.it` (Moreno Falaschi)

and molecular chemistry [6]. RSs formalise the mechanisms of biochemical systems, such as *facilitation* and *inhibition*. As a qualitative approximation of the real biochemical reactions, they consider if a necessary reagent is present or not, and likewise they consider if an inhibiting molecule is present or not. The possible reactants and inhibitors are called ‘entities’. RSs model in a direct way the interaction of a living cell with the environment (called ‘context’). However, two RSs are seen as independent models and do not interact.

In this paper, which is an extended version of [7], we present an encoding from RSs to the chained Core Network Algebra (cCNA), a variant of the open multiparty process algebra CNA [8]; the CNA equipped with mobility is referred to as the `link`-calculus [9, 10]. Here mobility is not needed. This formalism allows several processes to synchronise and communicate altogether, at the same time, with a new interaction mechanism based on links and link chains. The initial motivation for introducing an open multiparty mechanism in [9] was to encode Mobile Ambients [11], getting a much stronger operational correspondence than any available in the literature, such as the ones in [12, 13]. Later it was shown that the `link`-calculus allowed one to easily encode calculi for biology equipped with membranes, as in [14].

We illustrate our embedding by means of some examples coming both from the computer science and the biological field. We also show that our embedding preserves the main features of RSs, and prove its correctness and completeness from the operational semantics viewpoint.

Then, we present a methodology for verifying formally properties of Reaction Systems. The classical notion of bisimulation for process algebras allows to consider two processes as equivalent when one process can simulate all the actions executed by the other one and vice versa. In this paper we define a new notion of bisimilarity which takes into account the characteristics of biological systems. We define a new *bio-simulation* relation having in mind the possibility that two interacting systems may be compared w.r.t. a subset of the possible biological actions. This is useful for concentrating on the sub-model that one may need to consider for a specific study or application, without getting lost in the complexity of the full biological system, or network. The notion of bio-simulation relies on a new and simple assertion language, which allows to focus on some properties of the Reaction Systems to be verified. In fact, bio-similarity is parametric on a given assertion of interest. Then we prove that the well-known logical characterisation of bisimilarity in terms of Hennessy-Milner Logic [15] (HML) can be extended to the case of bio-similarity by tailoring HML formulas to the same assertion of interest used in bio-similarity. As HML is a powerful formalism for specifying properties of labeled transition systems, its variant introduced here is a suitable option to specify formulas over complex labels by abstracting from unnecessary details.

Our main contributions are as follows:

- RSs are encoded in a modular way, in the sense that all constituents of a RS, namely entities, reactions and the context, are seen as interacting cCNA processes; in principle, this allows to study the behaviour of any constituent in isolation, contrary to what happens in the basic RS framework, where the evolution is possible only when all constituents are given;

- any context is represented as an ordinary cCNA process, which allows us to specify recursive and non-deterministic contexts in a natural way; when deterministic contexts are considered, as in ordinary RSs, the cCNA computation is also deterministic and matches the evolution of the underlying RS;

- we define an assertion language to specify local properties of RSs and exploit it to define a novel notion of behavioural equivalence, called bio-similarity;

- we show that our assertions can be used to extend the Hennessy-Milner's logic to our encoding, in such a way that logical equivalence of processes coincide with bio-similarity.

Moreover, we sketch how the encoding can be used to enhance the expressivity of RSs along different dimensions:

- we sketch how one can express the behaviour of entity mutation, in such a way that the mutated version of the entity s can take part to only a subset of reactions requiring entity s;

- we sketch how with a little coding effort, our embedding allows two RSs to communicate; i.e. we can model scenarios where a subset of those entities that the context can provide, are instead provided by a second RS.

The main drawback of our proposal, is that the resulting cCNA code is verbose. Nevertheless it is clear that our encoding can be automatised by means of a proper front-end in an implementation of the `link`-calculus. The examples that we propose also show that some optimisations are possible to reduce the coding efforts when suitable assumptions are made about the provision of entities.

Related work. Process calculi have been used successfully to model biological processes, see [16] for a recent survey. We are not aware of any Structural Operational Semantics for RS; it seems not the case that a deterministic behaviour, as the one of the RS, once the initial state is fixed, could be defined by inference rules that classically are applied to define non-deterministic transition systems. The reversible computation paradigm for RS extends the RS framework by allowing backward computations as well as forward computations; for this goal a set of inference rules for rewriting logic has been defined in [17] to exactly keep trace of those elements that dissolve in the next computation step.

The proposed network of RSs [18] allows any RS in the network to receive entities produced from its neighbours (that will represent its context), with the hypothesis that RSs without neighbours will receive no entities from the context. In the idea we sketch, a RS can receive some entities from the context and some others from a second RS; moreover we can represent contexts having a recursive, non-deterministic behaviour. By exploiting recursion, the kind of interactions which can be defined can be complex and expressive. Example 34 and more in general the discussion in Section 7.2 show that the interaction between RSs can help to model new scenarios.

As already mentioned, a preliminary version of this paper appeared in [7]. There are several major differences w.r.t. the conference version [7], which is here extended as follows:

- we present several new examples (those in Sections 5 and 6.3) to illustrate our encoding, and give more detailed explanations about its use;
- we introduce an assertion language to specify the properties of RSs;
- we define the notion of bio-simulation to relate RSs and compare their behaviours;
- we show that our assertion language allows to specify properties extending to our encoding the Hennessy-Milner logic;
- we include here all proofs of main results.

Structure of the paper. Section 2 describes RSs and their semantics (interactive processes). Section 3 briefly describes the cCNA process algebra and its operational semantics. Section 4 defines the embedding of RSs in cCNA processes and shows some simple examples to illustrate it. Section 5 presents a couple of examples with automata theory and biological applications. Section 6 presents a methodology for the formal verification of properties of Reaction Systems that are expressed in a novel assertion language. Section 7 presents some features and advantages of our embedding for the compositionality of RSs. Finally, Section 8 discusses future work, and concludes.

2. An Overview of Reaction Systems

Natural Computing is concerned with human-designed computing inspired by nature as well as with computation taking place in nature. The theory of Reaction Systems [1] was born in the field of Natural Computing to model the behaviour of biochemical reactions taking place in living cells. Despite its initial aim, this formalism has shown to be quite useful not only for modeling biological phenomena, but also for the contributions which is giving to computer science [19], theory of computing, mathematics, biology [2, 3, 4, 5], and molecular chemistry [6]. Here we briefly review the basic notions of RSs, see [1] for more details.

The mechanisms that are at the basis of biochemical reactions and thus regulate the functioning of a living cell, are *facilitation* and *inhibition*. These mechanisms are reflected in the basic definitions of RSs.

Definition 1 (Reaction). Let S be a set of entities. A reaction over S is a triple $a = (R, I, P)$, where R, I, P are finite, non empty subsets of S and $R \cap I = \emptyset$.

The sets R, I, P are also written R_a, I_a, P_a and called the *reactant set* of a, the *inhibitor set* of a, and the *product set* of a, respectively. All reactants are needed for the reaction to take place. Any inhibitor blocks the reaction if it is present. Products are the outcome of the reaction. Also, $R_a \cup I_a$ is the set of the resources of a and $rac(S)$ denotes the set of all reactions in S. Because R and I are non empty, all products are produced from at least one reactant and every reaction can be inhibited in some way. Sometimes artificial inhibitors are used that are never produced by any reaction. For the sake of simplicity, in some examples, we will allow I to be empty.

Definition 2 (Reaction System). A Reaction System (RS) is an ordered pair $\mathcal{A} = (S, A)$ such that S is a finite set of entities, and $A \subseteq rac(S)$ is a set of reactions over S.

The set S is called the *background set* of $\mathcal{A}$; its elements represent molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system. The set A is the set of *reactions* of $\mathcal{A}$. Since S is finite, so is A: we denote by $|A|$ the number of reactions in A.

Definition 3 (Reaction Result). Given a set of entities S, let $W \subseteq S$ be a finite subset of entities.

1. Let $a \in rac(S)$ be a reaction over S. Then a is enabled by the entities in the set W, denoted by $en_a(W)$, if $R_a \subseteq W$ and $I_a \cap W = \emptyset$, i.e. all the reactants of a are in W, while none of the inhibitors of a are in W. The result of a on W, denoted by $res_a(W)$, is defined by: $res_a(W) \triangleq P_a$, if $en_a(W)$, and $res_a(W) \triangleq \emptyset$ otherwise.
2. Let $A \subseteq rac(S)$ be a (finite) set of reactions over S. The result of A on W, denoted by $res_A(W)$, is defined by: $res_A(W) \triangleq \bigcup_{a \in A} res_a(W)$.

The theory of Reaction Systems is based on the following assumptions.

- **No permanency.** An entity of a set W vanishes unless it is sustained by a reaction. This reflects the fact that a living cell would die for lack of energy, without chemical reactions.

- **No counting.** The basic model of RSs is very abstract and qualitative, i.e. the quantity of entities that are present in a cell is not taken into account.

- **Threshold nature of resources.** From the previous item, we assume that either an entity is available and there is enough of it (i.e. there are no conflicts), or it is not available at all.

The dynamic behaviour of a RS is formalized in terms of *interactive processes*.

Definition 4 (Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS and let n be a nonnegative integer; An n-step *interactive process* in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$ of finite sequences s.t. $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$ where $C_i, D_i \subseteq S$ are set of entities for any $i \in [0, n]$, $D_0 = \emptyset$, and $D_i = res_A(D_{i-1} \cup C_{i-1})$ for any $i \in [1, n]$.

Living cells are seen as open systems that continuously react with the external environment, in discrete steps. The sequence γ is the *context sequence* of π; it can be arbitrarily defined and represents the influence of the environment on the RS. The sequence δ is the *result sequence* of π and it is entirely determined by γ and A. The sequence $\tau = W_0, \ldots, W_n$ with $W_i = C_i \cup D_i$, for any $i \in [0, n]$ is called a *state sequence*. Each state W_i in a state sequence is the union of two sets: the context C_i at step i and the result $D_i = res_A(W_{i-1})$ from the previous step.

Since we will be able to deal with recursively contexts, we extend the notion of an interactive process to deal with infinite sequences.

Definition 5 (Extended Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an n-step interactive process, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$ Then, we let $\pi^\infty = (\gamma^\infty, \delta^\infty)$ be the extended interactive process of π, defined as $\gamma^\infty = \{C'_i\}_{i\in\mathbb{N}}$, $\delta^\infty = \{D'_i\}_{i\in\mathbb{N}}$, where:

$$C'_j = \begin{cases} C_j & \text{if } j \in [0, n] \\ \emptyset & \text{if } j > n \end{cases} \qquad D'_j = \begin{cases} D_0 & \text{if } j = 0 \\ res_A(D'_{j-1} \cup C'_{j-1}) & \text{if } (j \geq 1) \end{cases}$$

Given an extended interactive process $\pi = (\gamma, \delta)$, we denote by π^k the shift of π starting at the k-th state sequence; formally we let $\pi^k = (\gamma^k, \delta^k)$ with $\gamma^k = \{C'_i\}_{i\in\mathbb{N}}$, $\delta^k = \{D'_i\}_{i\in\mathbb{N}}$ with $C'_0 = C_k \cup D_k$, $D'_0 = \emptyset$, and $C'_i = C_{i+k}$, $D'_i = D_{i+k}$ for any $i \geq 1$.

3. Chained CNA (cCNA)

In this section we introduce the syntax and operational semantics of the process algebra cCNA (chained CNA) [7] to be used for encoding RSs. As already explained in the Introduction, cCNA is a variant of CNA [8], the non-mobile fragment of `link`-calculus [9, 10]. In cCNA the action prefixes are link chains and not just links.

Link Chains. Let $\mathcal{C}$ be the set of channels, ranged over by $\mathsf{a}, \mathsf{b}, \ldots$, and let $Act \triangleq \mathcal{C} \cup \{\tau\} \cup \{\square\}$ be the set of actions, ranged over by $\alpha, \beta, \ldots$, where the symbol τ denotes a *silent* action, while the symbol $\square$ denotes a *virtual* (non-specified) action. A *link* is a pair $\ell = {}^{\alpha}\backslash_{\beta}$; it is *solid* if $\alpha, \beta \neq \square$; intuitively, α and β are two interaction points, one for incoming requests and the other for outgoing requests. The link ${}^{\square}\backslash_{\square}$ is called *virtual*. A link is *valid* if it is solid or virtual. We let $\mathcal{L}$ be the set of valid links. A *link chain* is a finite sequence $v = \ell_1 \ldots \ell_n$ of (valid) links $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ such that:

1. for any $i \in [1, n-1]$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. $\exists i \in [1, n].\ \ell_i \neq {}^{\square}\backslash_{\square}$.

Virtual links represent missing elements of a chain. A chain is called *solid* if it does not contain any virtual link. The empty chain is denoted by ϵ. The equivalence $\bowtie$ models expansion/contraction of virtual links to adjust the length of a link chain.

Definition 6 (Equivalence $\bowtie$). We let $\bowtie$ be the least equivalence relation over link chains closed under the axioms (whenever both sides are well defined):

$$\begin{array}{rclcrcl} v{}^{\square}\backslash_{\square} & \bowtie & v & \qquad & v_1{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}v_2 & \bowtie & v_1{}^{\square}\backslash_{\square}v_2 \\ {}^{\square}\backslash_{\square}v & \bowtie & v & \qquad & v_1{}^{\alpha}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash_{\beta}v_2 & \bowtie & v_1{}^{\alpha}\backslash^{\square}_{\mathsf{a}}\backslash^{\mathsf{a}}_{\square}\backslash_{\beta}v_2 \end{array}$$

Two link chains of equal length can be merged whenever each position occupied by a solid link in one chain is occupied by a virtual link in the other chain and solid links in adjacent positions match. Positions occupied by virtual links in both chains remain virtual. Merging is denoted by $v_1 \bullet v_2$. For example, given $v_1 = {}^{\tau}\backslash^{\square}_{\mathsf{a}}\backslash^{\square}_{\square}\backslash_{\square}$, $v_2 = {}^{\square}\backslash^{\mathsf{a}}_{\square}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$ and $v = {}^{\tau}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$ we have $v_1 \bullet v_2 = v$, whereas $v_1 \bullet v$ is not defined. Notably the merge operation is commutative and associative.

Some names in a link chain can be restricted as non observable and transformed into silent actions τ. This is possible only if they are matched by some adjacent link. Restriction is denoted by $(\nu\,\mathsf{a})v$. For example, given $v = {}^{\tau}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$ as above, we have $(\nu\,\mathsf{a})v = {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$, whereas $(\nu\,\mathsf{b})v$ is not defined.

Syntax. The set of cCNA processes, denoted as $\mathcal{P}$ and ranged over by P, Q, is defined by the following grammar:

$$P, Q \quad ::= \quad \mathbf{0} \ \mid\ v.P \ \mid\ P + Q \ \mid\ P|Q \ \mid\ (\nu\,\mathsf{a})P \ \mid\ A$$

where v_i is a link chain, and A is a process identifier. The syntax of cCNA extends that of CNA [8] by allowing to use link chains as prefixes instead of links, i.e. we allow to write $v.P$ instead of $\ell.P$. For the rest it features nondeterministic choice $P + Q$ (also called sum), parallel composition $P|Q$, restriction $(\nu\,\mathsf{a})P$, and possibly recursively defined process identifiers A. Here we do not consider name mobility, which is present instead in the `link`-calculus and we omit the relabelling operator of CNA (not needed in the encoding).

As common in process algebras we restrict to consider prefix-guarded sums $v_1.P_1 + v_2.P_2$ and, exploiting associativity, we use the shorthand $\sum_{i \in I} v_i.P_i$ for a finite set of indexes $I = \{i_1, ..., i_k\}$ instead of $v_{i_1}.P_{i_1} + \cdots + v_{i_k}.P_{i_k}$. The inactive process $\mathbf{0}$ is thus just the empty summation.

In a prefix $v.P$ we can always assume that the links at the extremities of v are solid: if v needs to be used in larger chains, the operational semantics will add as many virtual links as needed by exploiting the equivalence $\bowtie$ (see rule

$$\frac{\upsilon \bowtie \upsilon_j \quad j \in I}{\sum_{i\in I} \upsilon_i.P_i \xrightarrow{\upsilon} P_j}\ (Sum) \qquad \frac{P \xrightarrow{\upsilon} P' \qquad (A \triangleq P) \in \Delta}{A \xrightarrow{\upsilon} P'}\ (Ide)$$

$$\frac{P \xrightarrow{\upsilon} P'}{(\nu\, \mathsf{a})P \xrightarrow{(\nu\, \mathsf{a})\upsilon} (\nu\, a)P'}\ (Res) \qquad \frac{P \xrightarrow{\upsilon} P'}{P|Q \xrightarrow{\upsilon} P'|Q}\ (Lpar) \qquad \frac{Q \xrightarrow{\upsilon} Q'}{P|Q \xrightarrow{\upsilon} P|Q'}\ (Rpar)$$

$$\frac{P \xrightarrow{\upsilon'} P' \qquad Q \xrightarrow{\upsilon} Q'}{P|Q \xrightarrow{\upsilon \bullet \upsilon'} P'|Q'}\ (Com)$$

Figure 1: SOS semantics of $\mathrm{c}\mathsf{CNA}$ processes.

Sum in Fig. 1). For example, the process ${}^{\mathsf{a}}\backslash_{\mathsf{b}}.P$ and ${}^{\mathsf{a}}\backslash_{\mathsf{b}}^{\square}\backslash_{\square}.P$ are completely equivalent.

Regarding process constants, we rely on a given set $\Delta = \{A_i \triangleq P_i\}_{i\in I}$ of (possibly recursive) process definitions.

Semantics. The operational semantics of cCNA is defined in the SOS style by the inference rules in Fig.1. The rules are reminiscent of those for Milner's CCS and they essentially coincide with those of CNA in [8]. The only difference is due to the presence of prefixes that are link chains. Briefly: rule (*Sum*) selects one alternative and puts as label a possible contraction/expansion of the link chain in the selected prefix; rule (*Ide*) selects one transition of the defining process for a constant; rule (*Res*) restricts some names in the label (it cannot be applied when $(\nu\, \mathsf{a})\upsilon$ is not defined); rules (*Lpar*) and (*Rpar*) account for interleaving in parallel composition; rule (*Com*) synchronises interactions (it cannot be applied when $\upsilon \bullet \upsilon'$ is not defined).

Example 7. As some simple examples, consider the recursive process definitions $H \triangleq {}^{\mathsf{a}}\backslash_{\mathsf{b}}.{}^{\mathsf{a}}\backslash_{\mathsf{c}}.H$ and $K \triangleq {}^{\mathsf{a}}\backslash_{\mathsf{b}}.K + {}^{\mathsf{a}}\backslash_{\mathsf{c}}.K$: the former recursively provides a link from a to b and then, at the next step, from a to c; the latter provides at each step a link from a to b or from a to c, nondeterministically.

Analogously to CNA, the operational semantics of cCNA satisfies the so called Accordion Lemma: whenever $P \xrightarrow{\upsilon} P'$ and $\upsilon' \bowtie \upsilon$ then $P \xrightarrow{\upsilon'} P'$.

3.1. Notation for link chains

Hereafter we make use of some new notations for link chains that will facilitate the presentation of our encoding.

Definition 8 (Replication). Let υ be a valid link chain such that $\upsilon\upsilon$ is also a valid link chain. The n times replication of υ, written υ^n, is defined recursively by letting $\upsilon^0 = \epsilon$ (i.e. the empty chain) and $\upsilon^n = \upsilon\upsilon^{n-1}$.

For example, the expression $({}^{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square})^3$ denotes the chain ${}^{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash^{\mathsf{a}}_{\square}\backslash^{\square}_{\mathsf{b}}\backslash^{\mathsf{a}}_{\square}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$. Instead the expression $({}^{\mathsf{a}}\backslash_{\mathsf{b}})^2$ is ill-defined because a does not match with b.

Then, we introduce the notation for *half links* that will be used in conjunction with the *open block of chain* to form regular link chains.

Definition 9 (Half links). Let a be a channel name, we denote by ${}^{\mathsf{a}}\backslash$ the *half left link*, and by $\backslash_{\mathsf{a}}$ the *half right link*.

In the encoding of RS we will make extensive use of subscripted names: each name a will come in two variants a_i and a_o. This is just a technical issue to prevent accidental matching between links. To see why this is important, compare the chain ${}^{\tau}\backslash^{\square}_{\mathsf{a}}\backslash^{\mathsf{a}}_{\square}\backslash_{\tau}$ with ${}^{\tau}\backslash^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\backslash_{\tau}$: the former is $\bowtie$ equivalent to the solid chain ${}^{\tau}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash_{\tau}$; the latter cannot become solid unless merged with a chain that links a_i to a_o, like ${}^{\square}\backslash^{\mathsf{a}_i}_{\square}\backslash^{\square}_{\mathsf{a}_o}\backslash_{\square}$. This form of subscripting is exploited in the definition of open blocks.

Definition 10 (Open block). Let σ be a finite sequence of names. We define an *open block* as $\Big(\mathop{\backslash\!\backslash}_{\mathsf{a}\,\in\,\sigma} {}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\Big)$, where a_i and a_o are annotated version of the name a (as explained above), by letting

$$\Big(\mathop{\backslash\!\backslash}_{\mathsf{a}\,\in\,\sigma} {}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\Big) \triangleq \begin{cases} \epsilon & \text{if } \sigma = \epsilon \text{ is the empty sequence} \\ {}^{\square}_{\mathsf{b}_i}\backslash^{\mathsf{b}_o}_{\square}\Big(\mathop{\backslash\!\backslash}_{\mathsf{a}\,\in\,\rho} {}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\Big) & \text{if } \sigma = \mathsf{b}\rho \end{cases}$$

Abusing the notation, we will use the open block notation for sets of names rather than sequences, assuming the names in the set are taken according to some default order (e.g. the lexicographic one).

We then combine half links and open blocks to form valid link chains.

For example, for $X = \{\mathsf{a}, \mathsf{b}\}$ the expression $\Big(\mathop{\backslash\!\backslash}_{\mathsf{c}\,\in\,X} {}^{\square}_{\mathsf{c}_i}\backslash^{\mathsf{c}_o}_{\square}\Big)$ denotes the block ${}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\backslash^{\square}_{\mathsf{b}_i}\backslash^{\mathsf{b}_o}_{\square}$; and the expression ${}^{\mathsf{r}_1}\backslash\Big(\mathop{\backslash\!\backslash}_{\mathsf{c}\,\in\,X} {}^{\square}_{\mathsf{c}_i}\backslash^{\mathsf{c}_o}_{\square}\Big)\backslash_{\mathsf{r}_2}$ denotes the chain ${}^{\mathsf{r}_1}\backslash^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\backslash^{\square}_{\mathsf{b}_i}\backslash^{\mathsf{b}_o}_{\square}\backslash_{\mathsf{r}_2}$.

4. From Reaction Systems to cCNA

Here we define an encoding of Reaction Systems into cCNA. The idea is to define separated processes for representing the behaviour of each entity, each reaction, and for the provisioning of each entity by the context.

In the following we refer to a given set of entities S and a set of reactions $A \subseteq rac(S)$, i.e. that the Reaction System $\mathcal{A} = (S, A)$ is known.

Processes for entities. Given an entity $s \in S$, we exploit five different pairs of channel names for the interactions over s:

- names s_i, s_o are used to test the presence of s in the system;
- names $\widehat{s}_i$, $\widehat{s}_o$ are used to test the provisioning of s from the context;

- names $\widetilde{s}_i$, $\widetilde{s}_o$ are used to test the production of s by some reaction;
- names $\overline{s}_i$, $\overline{s}_o$ are used to test the absence of s in the system;
- names $\underline{s}_i$, $\underline{s}_o$ are used to test the absence of s from the context.

We let P_s be the process implementing the presence of s in the system, and $\overline{P_s}$ be the one for its absence. They can be seen as instances of the same template, which is given below.

$$
\begin{array}{rcl}
P_s & \triangleq & E(s, \widetilde{s}, \widehat{s}, \underline{s}) \qquad \overline{P_s} \;\triangleq\; E(\overline{s}, \widetilde{s}, \widehat{s}, \underline{s}) \\
E(s, \widetilde{s}, \widehat{s}, \underline{s}) & \triangleq & \sum_{h,k \geq 0} ({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h \; {}^{\widehat{s}_i}\backslash^{\square}_{\widehat{s}_o}\backslash_{\square} ({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k . P_s \\
 & & + \\
 & & \sum_{h \geq 0, k \geq 1} ({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h \; {}^{\underline{s}_i}\backslash^{\square}_{\underline{s}_o}\backslash_{\square} ({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k . P_s \\
 & & + \\
 & & \sum_{h \geq 0} ({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h \; {}^{\underline{s}_i}\backslash_{\underline{s}_o} . \overline{P_s}
\end{array}
$$

The first line of $E(s, \widetilde{s}, \widehat{s}, \underline{s})$ accounts for the case where s is tested for presence by h reactions and produced by k reactions, while being provided by the context (${}^{\widehat{s}_i}\backslash_{\widehat{s}_o}$). Thus, s will be present at the next step (the continuation is P_s). Here h and k are not known a priori and therefore any combination is possible. In practice, by knowing the number of reactions that test s, we can bound the maximum values of h and k. The second line accounts for the analogous case where s is not provided by the context (${}^{\underline{s}_i}\backslash_{\underline{s}_o}$). The condition $k \geq 1$ guarantees that s will remain present (the continuation is P_s). The third line accounts for the case where s is tested for presence, but it is neither produced nor provided by the context. Therefore, in the next step s will be absent in the system (the continuation is $\overline{P_s}$). Note that in the case of $\overline{P_s}$ the test for presence of s in the system is just replaced by the test for its absence.

Processes for reactions. Here we focus on the encoding of the set of reactions $A \subseteq rac(S)$. We assume that all the reactions a are numbered and use j as an index for reactions. We introduce two channel names for each reaction aj:

- r_j to mark the occurrence of the reaction;
- p_j to mark the product set of the reaction.

We shall exploit names r_j, p_j to join the chains provided by the application of all the reactions. The process for the jth reaction $aj = (R_j, I_j, P_j)$ must assert either the possibility to apply the reaction or its impossibility. The first case happens when all its reactants are present (the link ${}^{s_i}\backslash_{s_o}$ is requested for any $s \in R_j$) and all its inhibitors are absent (the link ${}^{\overline{e}_i}\backslash_{\overline{e}_o}$ is requested for any $e \in I_j$), then the product set is released (the link ${}^{\widetilde{c}_i}\backslash_{\widetilde{c}_o}$ is requested for any $c \in P_j$). The second case can happen for two reasons: one of the reactants is absent (the link ${}^{\overline{s}_i}\backslash_{\overline{s}_o}$ is requested for some $s \in R_j$) or one of the inhibitors is present (the link ${}^{e_i}\backslash_{e_o}$ is requested for some $e \in I_j$). The process is recursive so that reactions can be applied at any step.

$$
\begin{array}{ll}
P_{aj} \triangleq & \\
{}^{r_j}\backslash \Big(\mathop{\backslash\!\!\backslash}\limits_{s\,\in\,R_j} {}^{\square}_{s_i}\backslash^{s_o}_{\square}\Big) \;\backslash \Big(\mathop{\backslash\!\!\backslash}\limits_{e\,\in\,I_j} {}^{\square}_{\overline{e}_i}\backslash^{\overline{e}_o}_{\square}\Big) \;\backslash^{\square}_{r_{j+1}}\backslash^{p_j}_{\square}\backslash \Big(\mathop{\backslash\!\!\backslash}\limits_{c\,\in\,P_j} {}^{\square}_{\widetilde{c}_i}\backslash^{\widetilde{c}_o}_{\square}\Big) \;\backslash_{p_{j+1}}.P_{aj} & \{aj \text{ is applicable}\} \\
+ & \\
\sum_{s \in R_j} {}^{r_j}\backslash^{\square}_{\overline{s}_i}\backslash^{\overline{s}_o}_{\square}\backslash^{\square}_{r_{j+1}}\backslash^{p_j}_{\square}\backslash_{p_{j+1}}.P_{aj} & \{aj \text{ is not applicable}\} \\
+ & \\
\sum_{e \in I_j} {}^{r_j}\backslash^{\square}_{e_i}\backslash^{e_o}_{\square}\backslash^{\square}_{r_{i+1}}\backslash^{p_j}_{\square}\backslash_{p_{j+1}}.P_{aj} & \{aj \text{ is not applicable}\}
\end{array}
$$

Channels r_j and r_{j+1} enclose the enabling/disabling condition of reaction aj. Channels p_j and p_{j+1} enclose the links related to the entities produced by aj. Each reaction defines a pattern to be satisfied, i.e. each reaction inserts as many virtual links as the number of reactants, inhibitors, and products, as required by the corresponding reaction.

We will see that all the link chain labels of transitions follow the same schema: first we find all the reactions limited to the reactants and inhibitors (chained using r_j channels), then all the supplies by the contexts (chained using the channel cxt, to be introduced next), and finally the products for all the reactions (chained using p_j channels). For notational convenience, we fix that $r_{|A|+1} = cxt$ and $p_{|A|+1} = \tau$. This schema will be later illustrated in detail in Example 15.

Processes for contexts. For marking the part of the chain provided by the context, we exploit the name cxt. In RSs, the context sequence γ provides a set of entities C_n at each instant of time n: for each entity $s \in S$, the context must say if the entity is provided or not. Correspondingly, we introduce another process Cxt_n defined as follows:

$$
Cxt_n \quad \triangleq \quad {}^{cxt}\backslash \Big(\mathop{\backslash\!\!\backslash}\limits_{s\,\in\,C_n} {}^{\square}_{\widehat{s}_i}\backslash^{\widehat{s}_o}_{\square}\Big) \;\backslash \Big(\mathop{\backslash\!\!\backslash}\limits_{e\,\notin\,C_n} {}^{\square}_{\underline{e}_i}\backslash^{\underline{e}_o}_{\square}\Big) \;\backslash_{p_1}.Cxt_{n+1}
$$

We only consider Cxt_n with $n > 0$, as the entities that are present at step zero are considered to be present in the initial system (if $s \in C_0$ the process P_s will be present initially, otherwise $\overline{P_s}$ will be present).

Encoding. In the following we use the following conventions for denoting different categories of names:

- *decs* $\triangleq \{s, \overline{s}, \widetilde{s}, \widehat{s}, \underline{s} \mid s \in S\}$ is the set of channel names for decorated entities (without subscripts $_i$ and $_o$);
- *ents* $\triangleq \{d_i, d_o \mid d \in decs\}$ is the set of channel names for entities;
- *reacts* $\triangleq \{r_1, ..., r_{|A|+1}\}$ is the set of channel names r_j associated with each reaction aj (we remind that $r_{|A|+1} = cxt$);

- *prods* $\triangleq \{p_1, ..., p_{|A|}\}$ is the set of channel names p_j for product sets associated with each reaction aj (we remind that $p_{|A|+1} = \tau$).

Definition 11 (Encoding). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in\mathbb{N}}$. We define its cCNA encoding $[\![\mathcal{A}, \gamma]\!]$ as follows:

$$[\![\mathcal{A}, \gamma]\!] \triangleq (\nu\, names) \left(I \mid \prod_{a\in A} P_a \mid Cxt_1 \mid \prod_{s\in C_0} P_s \mid \prod_{s\notin C_0} \overline{P_s} \right)$$

where $names = reacts \cup ents \cup prods \cup \{cxt\}$. For technical reasons, we introduce the (trivially recursive) init process $I \triangleq {}^{\tau}\backslash_{r_1}.I$: it is needed to allow the name r_1 to be matched at the start of any chain at any instant of time.

It is important to observe that, for each transition, our cCNA encoding requires all the processes running in parallel to interact in that transition. This is due to the fact that all the channel names r_j, p_j, cxt, including those for decorated names $s_i, s_o, \overline{s}_i, \overline{s}_o, ...$ are restricted.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$. Let $P = [\![\mathcal{A}, \gamma]\!]$ its cCNA encoding. If exists P' such that $P \xrightarrow{(\nu\, names)\upsilon} P'$ is a transition of P, then*

1. *for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in υ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in υ; the channel cxt appears in υ;*
2. *for each reaction $aj \in A$ and each virtual link offered by processes P_a and Cxt_1 is overlapped by exactly one solid link offered by processes representing entities.*

The topmost restriction $(\nu\, names)$ appearing in the process $[\![\mathcal{A}, \gamma]\!]$ serves to guarantee that all names appearing in a link of the chain labelling a transition are matched. Since all names appearing in any prefix of $[\![\mathcal{A}, \gamma]\!]$ are restricted, in the transition $[\![\mathcal{A}, \gamma]\!] \xrightarrow{(\nu\, names)\upsilon} P'$ it means that the observation $(\nu\, names)\upsilon$ has the form ${}^{\tau}\backslash_{\tau} \ldots {}^{\tau}\backslash_{\tau}$, i.e., it is silent, and that υ is solid. Later on we will be interested in reasoning about the actual chain υ used in the transition. It has the peculiarity to start and end with silent actions and to include all names in $reacts \cup prods \cup \{cxt\}$. As a matter of notation we call such chain υ *complete*.

Definition 13 (Complete Chain). A chain υ is called *complete* if it is solid (i.e., it contains no virtual link) and has silent actions τ at its extremes. We write $P \overset{\upsilon}{\Rightarrow} P'$ to mean that $P \xrightarrow{\upsilon} P'$ with υ complete.

We will then use $(\!|\mathcal{A}, \gamma|\!)$ to refer to the encoding without topmost name restrictions, i.e.,

$$(\!|\mathcal{A}, \gamma|\!) \triangleq I \mid \prod_{a\in A} P_a \mid Cxt_1 \mid \prod_{s\in C_0} P_s \mid \prod_{s\notin C_0} \overline{P_s}.$$

and we will focus on the complete transitions of $(\!|\mathcal{A},\gamma|\!)$.

The following Corollary immediately follows from Lemma 12.

Corollary 14. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$. Let $Q = (\!|\mathcal{A},\gamma|\!)$ and $P = [\![\mathcal{A},\gamma]\!] = (\nu\, names)Q$. Then $P \xrightarrow{(\nu\, names)\upsilon} P'$ iff $Q \overset{\upsilon}{\Rightarrow} Q'$ and $P' = (\nu\, names)Q'$.*

Example 15. Let $\mathcal{A}$ be a RS whose specification contains two entities, $s1$ and $s2$, and the reactions $r_1 = (s1, \emptyset, s2)$ and $r_2 = (s2, \emptyset, s1)$ that produce $s2$ if $s1$ is present and $s1$ if $s2$ is present. For simplicity, we consider empty sets of inhibitors, which are not allowed by Definition 1, but the reader can assume a void inhibitor is present in both reactions. Then, we assume an extended interactive process $\pi = (\gamma, \delta)$ where the context γ provides $s1$ and $s2$ at every step, but we assume that only $s1$ is initially present. Since the context sequence is constant, we omit the subscript from Cxt. The corresponding cCNA process is $[\![\mathcal{A},\gamma]\!] \triangleq (\nu\, names)(\!|\mathcal{A},\gamma|\!)$, with

$$(\!|\mathcal{A},\gamma|\!) \triangleq I \mid P_{s1} \mid \overline{P}_{s2} \mid P_{r1} \mid P_{r2} \mid Cxt$$

where:

$$\begin{array}{rcl}
P_{r_1} & \triangleq & {}^{r_1}\backslash^{\square}_{s1_i}\backslash^{s1_o}_{\square}\backslash^{\square}_{r_2}\backslash^{p_1}_{\square}\backslash^{\square}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\square}\backslash_{p_2}.P_{r_1} + \ldots;\\
P_{r_2} & \triangleq & \cdots + {}^{r_2}\backslash^{\square}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_2}_{\square}\backslash_{\tau}.P_{r_2} + \ldots;\\
P_{s1} & \triangleq & {}^{s1_i}\backslash^{\square}_{s1_o}\backslash^{\widehat{s1}_i}_{\square}\backslash_{\widehat{s1}_o}.P_{s1} + \ldots;\\
\overline{P}_{s2} & \triangleq & {}^{\overline{s2}_i}\backslash^{\square}_{\overline{s2}_o}\backslash^{\widehat{s2}_i}_{\square}\backslash^{\square}_{\widehat{s2}_o}\backslash^{\widetilde{s2}_i}_{\square}\backslash_{\widetilde{s2}_o}.P_{s2} + \ldots;\\
Cxt & \triangleq & {}^{cxt}\backslash^{\square}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\square}\backslash^{\square}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\square}\backslash_{p_1}.Cxt
\end{array}$$

For clarity of exposition, we show the code of the processes just in part, to focus on the prefixes that will be involved in the first transition of the system. In Figure 2 we show the structure of a link chain label related to the execution of such a transition. The yellow blocks are referred to init process I, to the processes encoding the reactions, P_{r_1} and P_{r_2}, and to the context Cxt. As the figure puts in evidence, these two kinds of processes determine the structure of the link chain, from end to end, i.e. from the left τ to the right one. We could say that these processes form the *backbone* of the interaction. In contrast, the processes encoding the entities, P_{s1}, $\overline{P}_{s2}$, provide the solid links to be merged with the virtual links of the backbone (i.e. to be plugged in the backbone). In Figure 2, at the bottom of the chain, we have underlined with brackets the origin of the solid links that appear in the chain: the notation $P_1(P_2, P_3)$ means that the segment of the link chain is delimited by the process P_1 and it leaves "holes" where processes between the brackets, in this case P_2 and P_3, insert their links. Formally, we have

$$(\!|\mathcal{A},\gamma|\!) \xrightarrow{{}^{\tau}\backslash^{r_1}_{r_1}\backslash^{s1_i}_{s1_i}\backslash^{s1_o}_{s1_o}\backslash^{r_2}_{r_2}\backslash^{\overline{s2}_i}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\overline{s2}_o}\backslash^{cxt}_{cxt}\backslash^{\widehat{s1}_i}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\widehat{s1}_o}\backslash^{\widehat{s2}_i}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\widehat{s2}_o}\backslash^{p_1}_{p_1}\backslash^{\widetilde{s2}_i}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\widetilde{s2}_o}\backslash^{p_2}_{p_2}\backslash_{\tau}} P'$$

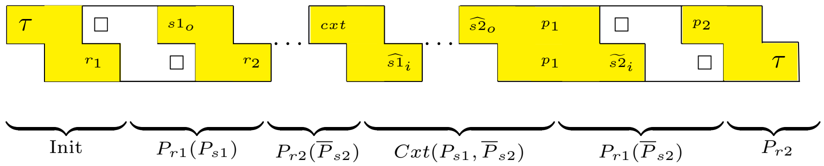

Figure 2: The link chain structure arising from reactions and context processes.

Since the chain in the transition label is complete we can also write

$$(\!|\mathcal{A}, \gamma|\!) \xRightarrow{{}^{\tau}\backslash^{r_1}_{r_1}\backslash^{s1_i}_{s1_i}\backslash^{s1_o}_{s1_o}\backslash^{r_2}_{r_2}\backslash^{\overline{s2}_i}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\overline{s2}_o}\backslash^{cxt}_{cxt}\backslash^{\widehat{s1}_i}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\widehat{s1}_o}\backslash^{\widehat{s2}_i}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\widehat{s2}_o}\backslash^{p_1}_{p_1}\backslash^{\widetilde{s2}_i}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\widetilde{s2}_o}\backslash^{p_2}_{p_2}\backslash_{\tau}} P'$$

Example 15 outlines two different roles of the processes defining the translation of an interactive process: those processes encoding the reactions and the context provide the backbone of each transition, whereas the processes encoding the entities provide the resources needed for the communication to take place.

The flat function. Our transition labels are quite verbose; then, to simplify their processing, we introduce a function that takes a solid link chain and returns a simple string by eliminating all the channel matching pairs and leaving just one placeholder for them. This transformation is harmless, in the sense that it retains all the information in the chain, because it is applied to complete chains only. The function $\mathit{flat}(\cdot)$ is defined inductively as follows:

$$\mathit{flat}(\epsilon) \triangleq \epsilon \qquad \mathit{flat}({}^{\alpha}\backslash_{\beta}) \triangleq \begin{cases} \beta & \text{if } \beta \in \mathit{reacts} \cup \{\mathit{cxt}\} \cup \mathit{prods} \\ d & \text{if } \beta = d_i \text{ with } d \in \mathit{decs} \\ \epsilon & \text{otherwise} \end{cases}$$

$$\mathit{flat}({}^{\alpha}\backslash_{\beta}\upsilon) \quad \triangleq \quad \mathit{flat}({}^{\alpha}\backslash_{\beta})\mathit{flat}(\upsilon)$$

where the usual string concatenation if represented by juxtaposition.

For example, if we consider again the complete label

$$\upsilon = {}^{\tau}\backslash^{r_1}_{r_1}\backslash^{s1_i}_{s1_i}\backslash^{s1_o}_{s1_o}\backslash^{r_2}_{r_2}\backslash^{\overline{s2}_i}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\overline{s2}_o}\backslash^{cxt}_{cxt}\backslash^{\widehat{s1}_i}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\widehat{s1}_o}\backslash^{\widehat{s2}_i}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\widehat{s2}_o}\backslash^{p_1}_{p_1}\backslash^{\widetilde{s2}_i}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\widetilde{s2}_o}\backslash^{p_2}_{p_2}\backslash_{\tau}$$

from Example 15, we have

$$\mathit{flat}(\upsilon) = r_1 \; s1 \; r_2 \; \overline{s2} \; \mathit{cxt} \; \widehat{s1} \; \widehat{s2} \; p_1 \; \widetilde{s2} \; p_2.$$

It is then immediate to define the function *unflat* to rebuild the complete label from the compact string (here we exploit again the half link and block notation):

$$\mathit{unflat}(x) \triangleq \begin{cases} {}^{x}_{x} & \text{if } x \in \mathit{reacts} \cup \{\mathit{cxt}\} \cup \mathit{prods} \\ {}^{x_i}_{x_i}\backslash^{x_o}_{x_o} & \text{if } x \in \mathit{decs} \end{cases}$$

$$unflat(x_1 \dots x_n) \triangleq {}^{\tau}\backslash unflat(x_1)\backslash \dots \backslash unflat(x_n)\backslash_{\tau}$$

It is immediate to check that for any complete label υ of our processes we have $\upsilon = unflat(flat(\upsilon))$.

With the next proposition, we analyse the structure of a cCNA process encoding of a reactive process after one transition step. In the following four statements, for brevity, we let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in A, with $\gamma = \{C_i\}_{i\in\mathbb{N}}$ and $\delta = \{D_i\}_{i\in\mathbb{N}}$.

Proposition 16 (Correctness 1). *Let $P = [\![\mathcal{A}, \gamma]\!]$ with*

$$P = (\nu\, names) \left(I \mid \prod_{a\in A} P_a \mid Cxt_1 \mid \prod_{s\in C_0} P_s \mid \prod_{s\notin C_0} \overline{P}_s \right).$$

If there exists P' such that $P \xrightarrow{\upsilon} P'$, it holds that:

1. $\upsilon = {}^{\tau}\backslash_{\tau} \dots {}^{\tau}\backslash_{\tau}$*, and*
2. $P' = (\nu\, names) \left(I \mid \prod_{a\in A} P_a \mid Cxt_2 \mid \prod_{s\in C_1\cup D_1} P_s \middle| \prod_{s\notin C_1\cup D_1} \overline{P}_s \right)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, we have $P' = [\![\mathcal{A}, \gamma^1]\!]$.

Now, we extend the previous result to a series of transitions.

Corollary 17 (Correctness 2). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{{}^{\tau}\backslash_{\tau}\dots{}^{\tau}\backslash_{\tau}}{}^{j} P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = [\![\mathcal{A}, \gamma^j]\!]$.*

With the following propositions, we prove that, given a RS $\mathcal{A} = (S, A)$ and an extended interactive process $\pi = (\gamma, \delta)$, then the cCNA process $[\![\mathcal{A}, \gamma]\!]$ can simulate all the evolutions of π.

Proposition 18 (Completeness 1). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $\pi^1 = (\gamma^1, \delta^1)$. Then, $P \xrightarrow{{}^{\tau}\backslash_{\tau}\dots{}^{\tau}\backslash_{\tau}} P' = [\![\mathcal{A}, \gamma^1]\!]$.*

Now, we extend the previous result to a series of transitions.

Corollary 19 (Completeness 2). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $\pi^j = (\gamma^j, \delta^j)$. Then, $P \xrightarrow{{}^{\tau}\backslash_{\tau}\dots{}^{\tau}\backslash_{\tau}}{}^{j} P'' = [\![\mathcal{A}, \gamma^j]\!]$.*

5. Examples

Semantically, the topmost restriction ($\nu\, names$) filters out any interaction with virtual links, and releases a private interaction among all participants where all the channel names in the transition labels are hidden (their occurrences are all replaced by τ). This amounts to require that only complete chains are computed by the interaction. In this section, for simplicity, we shall often omit topmost restrictions ($\nu\, names$) from our encoding, but we shall take into account only transitions whose labels are complete chains, i.e. they do not contain virtual links and start/end with the τ action symbol. This way it is possible to observe all channel names that occur in the interaction.

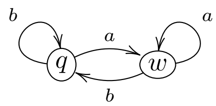

Figure 3: Minimal deterministic labelled transition system.

5.1. Labelled transition system

This example is inspired by the example in [1], where a deterministic transition system is encoded in the Reaction System framework. Here we consider the minimal deterministic transition system in Figure 3.

At the level of RSs, the set of entities to consider is the union of sets of states and of labels of the transition system. Moreover, there is one reaction for each transition: its reactant set consists of the source state and transition label, its inhibitor set includes every other state and label, and its product set is the singleton with the target state. For the transition system in Fig. 3, we take $S = \{q, w, a, b\}$ and the reactions are as follows:

$$
\begin{array}{llcll}
1 & (\{q,a\},\{w,b\},\{w\}) & \qquad & 2 & (\{q,b\},\{w,a\},\{q\}) \\
3 & (\{w,a\},\{q,b\},\{w\}) & \qquad & 4 & (\{w,b\},\{q,a\},\{q\})
\end{array}
$$

Next we show how the above RS is encoded in cCNA.

Encoding of the reactions. The encoding of the reactions is given in a parametric way, with $n \in \{1, 2, 3, 4\}$:

$$
\begin{array}{lcl}
P_n(q,b,w,a,q) & \triangleq & \upsilon_n(q,b,w,a,q).P_n(q,b,w,a,q) \\
 & & + \\
 & & \sum_{x\in\{\overline{q},\overline{b},w,a\}} \ \upsilon'_n(x).P_n(q,b,w,a,q)
\end{array}
$$

where

$$
\begin{array}{lcl}
\upsilon_n(q,b,w,a,q) & \triangleq & {}^{r_n}\backslash^{\square}_{q_i}\backslash^{q_o}_{\square}\backslash^{\square}_{b_i}\backslash^{b_o}_{\square}\backslash^{\square}_{\overline{w}_i}\backslash^{\overline{w}_o}_{\square}\backslash^{\square}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash^{\square}_{\widetilde{q}_i}\backslash^{\widetilde{q}_o}_{\square}\backslash_{p_{n+1}} \\
\upsilon'_n(x) & \triangleq & {}^{r_n}\backslash^{\square}_{x_i}\backslash^{x_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash_{p_{n+1}}
\end{array}
$$

Then, we have

$$
\begin{array}{lcllcl}
P_1 & \triangleq & P_1(q,a,w,b,w) & \qquad P_3 & \triangleq & P_3(w,a,q,b,w) \\
P_2 & \triangleq & P_2(q,b,w,a,q) & \qquad P_4 & \triangleq & P_4(w,b,q,a,q)
\end{array}
$$

and we put, as usual, $r_5 = cxt$ and $p_5 = \tau$.

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate the encoding for the entities that are not provided by the context and that can be produced by the reactions, and the ones that can be provided by the context and that are not produced by the reactions.

Here, for the entities q and w that are not provided by the context, we let:

$$\begin{array}{lcllcl} P_q & \triangleq & E(q,\widetilde{q}) & \overline{P}_q & \triangleq & E(\overline{q},\widetilde{q}) \\ P_w & \triangleq & E(w,\widetilde{w}) & \overline{P}_w & \triangleq & E(\overline{w},\widetilde{w}) \end{array}$$

where:

$$\begin{array}{lcl} E(q,\widetilde{q}) & \triangleq & \sum_{h=1}^{3}({}^{q_i}\backslash^{\Box}_{q_o}\backslash_{\Box})^h\ {}^{\widetilde{q}_i}\backslash_{\widetilde{q}_o}.P_q \\ & & + \\ & & \sum_{h=1}^{3}({}^{q_i}\backslash^{\Box}_{q_o}\backslash_{\Box})^h.\overline{P}_q \end{array}$$

In fact the presence/absence of q and w will be exploited by at least one reaction and at most three reactions.

Here, for the entities a and b that can be provided by the context but not produced by reactions, we let:

$$\begin{array}{lcllcl} P_a & \triangleq & E(a,\widehat{a},\underline{a}) & \overline{P}_a & \triangleq & E(\overline{a},\widehat{a},\underline{a}) \\ P_b & \triangleq & E(b,\widehat{b},\underline{b}) & \overline{P}_b & \triangleq & E(\overline{b},\widehat{b},\underline{b}) \end{array}$$

where:

$$\begin{array}{lcl} E(a,\widehat{a},\underline{a}) & \triangleq & \sum_{h=1}^{3}({}^{a_i}\backslash^{\Box}_{a_o}\backslash_{\Box})^h\ {}^{\widehat{a}_i}\backslash_{\widehat{a}_o}.P_a \\ & & + \\ & & \sum_{h=1}^{3}({}^{a_i}\backslash^{\Box}_{a_o}\backslash_{\Box})^h\ {}^{\underline{a}_i}\backslash_{\underline{a}_o}.\overline{P}_a \end{array}$$

Finally, for the context, the encoding follows:

$$Cxt \triangleq {}^{cxt}\backslash^{\Box}_{\widehat{a}_i}\backslash^{\widehat{a}_o}_{\Box}\backslash^{\Box}_{\underline{b}_i}\backslash^{\underline{b}_o}_{\Box}\backslash_{p_1}.Cxt \quad + \quad {}^{cxt}\backslash^{\Box}_{\widehat{b}_i}\backslash^{\widehat{b}_o}_{\Box}\backslash^{\Box}_{\underline{a}_i}\backslash^{\underline{a}_o}_{\Box}\backslash_{p_1}.Cxt$$

Notice that we exploit here the capabilities of the process algebraic framework to define a nondeterministic, recursive context. We model the context to always offer either a or b, but never both the entities together. The reason is that in the other cases (providing both a and b or neither of them) would lead the system to be stuck because of the simplifications we have adopted in the other processes.

Now, we assume that we have an initial configuration containing the entities q and b:

$$Sys \triangleq I \mid P_q \mid \overline{P}_w \mid \overline{P}_a \mid P_b \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt.$$

Then, only the second reaction can be applied, and the transition carries the complete label υ below

$$\tau\backslash^{r_1}_{r_1}\backslash^{\overline{\mathbf{a}}_{\mathbf{i}}}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\overline{\mathbf{a}}_{\mathbf{o}}}\backslash^{r_2}_{r_2}\backslash^{\mathbf{q}_{\mathbf{i}}}_{q_i}\backslash^{q_o}_{\mathbf{q}_{\mathbf{o}}}\backslash^{\mathbf{b}_{\mathbf{i}}}_{b_i}\backslash^{b_o}_{\mathbf{b}_{\mathbf{o}}}\backslash^{\overline{\mathbf{w}}_{\mathbf{i}}}_{\overline{w}_i}\backslash^{\overline{w}_o}_{\overline{\mathbf{w}}_{\mathbf{o}}}\backslash^{\overline{\mathbf{a}}_{\mathbf{i}}}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\overline{\mathbf{a}}_{\mathbf{o}}}\backslash^{r_3}_{r_3}\backslash^{\overline{\mathbf{a}}_{\mathbf{i}}}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\overline{\mathbf{a}}_{\mathbf{o}}}\backslash^{r_4}_{r_4}\backslash^{\overline{\mathbf{w}}_{\mathbf{i}}}_{\overline{w}_i}\backslash^{\overline{w}_o}_{\overline{\mathbf{w}}_{\mathbf{o}}}\backslash^{cxt}_{cxt}\backslash^{\widehat{\mathbf{a}}_{\mathbf{i}}}_{\widehat{a}_i}\backslash^{\widehat{a}_o}_{\widehat{\mathbf{a}}_{\mathbf{o}}}\backslash^{\underline{\mathbf{b}}_{\mathbf{i}}}_{\underline{b}_i}\backslash^{\underline{b}_o}_{\underline{\mathbf{b}}_{\mathbf{o}}}\backslash^{p_1}_{p_1}\backslash^{p_2}_{p_2}\backslash^{\widetilde{\mathbf{q}}_{\mathbf{i}}}_{\widetilde{q}_i}\backslash^{\widetilde{q}_o}_{\widetilde{\mathbf{q}}_{\mathbf{o}}}\backslash^{p_3}_{p_3}\backslash^{p_4}_{p_4}\backslash\tau$$

The parts in bold are provided by the entity processes, the other parts are provided by the processes encoding the reactions and by the process encoding the context (starting at cxt and ending at p_1. In the label we can read that reactions 1 and 4 have been not executed because the entity a is absent, the reaction 3 has been not applied because the entity w is absent, then only reaction 2 has been applied, and it has produced the entity q. Also, the context provides entity a, that will be available in the next state, and not the entity b. Now, to let the label more readable, we show the result of the application of the function $flat(\cdot)$ to it:

$$r_1\ \overline{a}\ r_2\ q\ b\ \overline{w}\ \overline{a}\ r_3\ \overline{a}\ r_4\ \overline{w}\ cxt\ \widehat{a}\ \underline{b}\ p_1\ p_2\ \widetilde{q}\ p_3\ p_4.$$

5.2. A biological toy example of gene expression

We consider a biological toy example in the style of gene's alternative splicing [20]. Alternative splicing is a regulated process during gene expression that results in a single gene coding for multiple proteins. In practice, particular exons of a gene may be included within or excluded from the final processed messenger RNA (mRNA) produced from that gene. In our example, a gene a codes for a protein T when molecules G is present and C is absent, and in the opposite situation a codes for protein T'. This behavior is encoded in reactions 1 and 2. Then, reaction 3 codes for the production of C when proteins T and F are present, and T' absent; reaction 4 codes for the production of G when proteins T' is present and F is absent.

Encoding of the reactions. The encoding of reactions is given in a parametric way:

$$\begin{array}{rcl} P_n(a,G,C,T) & \triangleq & \pi_n(a,G,C,T).P_n(a,G,C,T) \\ & & + \\ & & \sum_{x\in\{a,G,C\}} \pi'_n(x).P_n(a,G,C,T) \end{array}$$

where

$$\begin{array}{rcl} \pi_n(a,G,C,T) & \triangleq & {}^{r_n}\backslash^{\square}_{a_i}\backslash^{a_o}_{\square}\backslash^{\square}_{G_i}\backslash^{G_o}_{\square}\backslash^{\square}_{\overline{C}_i}\backslash^{\overline{C}_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash^{\square}_{\widetilde{T}_i}\backslash^{\widetilde{T}_o}_{\square}\backslash_{p_{n+1}} \\ \pi'_n(x) & \triangleq & {}^{r_n}\backslash^{\square}_{x_i}\backslash^{x_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash_{p_{n+1}} \end{array}$$

Then we have

$$P_1 \triangleq P_1(a,G,C,T) \qquad P_2 \triangleq P_2(a,C,G,T') \qquad P_3 \triangleq P_3(F,T,T',C)$$

$$\begin{array}{rcl} P_4 & \triangleq & {}^{r_4}\backslash^{\square}_{T'_i}\backslash^{T'_o}_{\square}\backslash^{\square}_{\overline{F}_i}\backslash^{\overline{F}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_4}_{\square}\backslash^{\square}_{\widetilde{G}_i}\backslash^{\widetilde{G}_o}_{\square}\backslash_{\tau}.P_4 \\ & & + \\ & & {}^{r_4}\backslash^{\square}_{\overline{T'}_i}\backslash^{\overline{T'}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_4}_{\square}\backslash_{\tau}.P_4 \;+\; {}^{r_4}\backslash^{\square}_{F_i}\backslash^{F_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_4}_{\square}\backslash_{\tau}.P_4 \end{array}$$

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate three types of encodings: (1) for the entities that are not provided by the context and can be produced by the reactions; (2) for the entities that can be provided by the context and can be produced by the reactions; (3) for the entities that are only provided by the context.

Here, the entities T and T' that can be produced by the reactions and that are not provided by the context:

$$P(T,\widetilde{T}) \;\triangleq\; \sum_{h=0}^{1}({}^{T_i}\backslash^{\square}_{T_o}\backslash_{\square})^h\; {}^{\widetilde{T}_i}\backslash_{\widetilde{T}_o}.P(T,\widetilde{T}) \;+\; {}^{T_i}\backslash_{T_o}.P(\overline{T},\widetilde{T})$$

Then, we have

$$P_T \triangleq P(T,\widetilde{T}) \qquad \overline{P}_T \triangleq P(\overline{T},\widetilde{T}) \qquad P_{T'} \triangleq P(T',\widetilde{T'}) \qquad \overline{P}_{T'} \triangleq P(\overline{T'},\widetilde{T'})$$

The entities that can be produced by the reactions and that can be provided by the context are as follows:

$$\begin{array}{rcl} P(C,\widehat{C},\underline{C},\widetilde{C}) & \triangleq & \sum_{h=0}^{1}({}^{C_i}\backslash^{\square}_{C_o}\backslash_{\square})^h\ {}^{\widehat{C}_i}\backslash^{\square}_{\widehat{C}_o}\backslash_{\square}({}^{\widetilde{C}_i}\backslash^{\square}_{\widetilde{C}_o}\backslash_{\square})^h.P(C,\widehat{C},\underline{C},\widetilde{C}) \\ & & + \\ & & \sum_{h=0}^{1}({}^{C_i}\backslash^{\square}_{C_o}\backslash_{\square})^h\ {}^{\underline{C}_i}\backslash_{\underline{C}_o}.P(\overline{C},\widehat{C},\underline{C},\widetilde{C}) \\ & & + \\ & & \sum_{h=0}^{1}({}^{C_i}\backslash^{\square}_{C_o}\backslash_{\square})^h\ {}^{\underline{C}_i}\backslash_{\underline{C}_o}{}^{\square}\backslash^{\widehat{C}_i}_{\square}\backslash_{\widehat{C}_o}.P(C,\widehat{C},\underline{C},\widetilde{C}) \end{array}$$

Then, we have

$$\begin{array}{rclcrcl} P_C & \triangleq & P(C,\widehat{C},\underline{C},\widetilde{C}) & \qquad & P_G & \triangleq & P(G,\widehat{G},\underline{G},\widetilde{G}) \\ \overline{P}_C & \triangleq & P(\overline{C},\widehat{C},\underline{C},\widetilde{C}) & \qquad & \overline{P}_G & \triangleq & P(\overline{G},\widehat{G},\underline{G},\widetilde{G}) \end{array}$$

The encoding of the entity F that can only be provided by the context follows:

$$\begin{array}{rclcl} P_F & \triangleq & \sum_{h=0}^{1}({}^{F_i}\backslash^{\square}_{F_o}\backslash_{\square})^h\ {}^{\widehat{F}_i}\backslash_{\widehat{F}_o}.P_F & + & \sum_{h=0}^{1}({}^{F_i}\backslash^{\square}_{F_o}\backslash_{\square})^h\ {}^{\underline{F}_i}\backslash_{\underline{F}_o}.\overline{P}_F \\ \overline{P}_F & \triangleq & \sum_{h=0}^{1}({}^{\overline{F}_i}\backslash^{\square}_{\overline{F}_o}\backslash_{\square})^h\ {}^{\widehat{F}_i}\backslash_{\widehat{F}_o}.P_F & + & \sum_{h=0}^{1}({}^{\overline{F}_i}\backslash^{\square}_{\overline{F}_o}\backslash_{\square})^h\ {}^{\underline{F}_i}\backslash_{\underline{F}_o}.\overline{P}_F \end{array}$$

Also in this example we account for a nondeterministic context that can (nondeterministically) provide any combination of the entities C, G F:

$$Cxt \triangleq \sum_{\substack{C^\star \in \{C,\overline{C}\} \\ G^\star \in \{G,\overline{G}\} \\ F^\star \in \{F,\overline{F}\}}} {}^{cxt}\backslash^{\square}_{C_i^\star}\backslash^{C_o^\star}_{\square}\backslash^{\square}_{G_i^\star}\backslash^{G_o^\star}_{\square}\backslash^{\square}_{F_i^\star}\backslash^{F_o^\star}_{\square}\backslash_{p_1}.Cxt$$

Now, to show a possible composition of a transition label, we assume a system where only the entities a, T', and G are present:

$$Sys \triangleq I \mid P_a \mid \overline{P}_C \mid P_G \mid \overline{P}_F \mid \overline{P}_T \mid P_{T'} \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt$$

In the above configuration, reactions 1 and 4 can be applied, and also we assume that the context will provide the entity F, that will be available in the target configuration. Instead of showing the complete transition label, we give its flattened version obtained by applying the function *flat*$(\cdot)$:

$$r_1\ a\ G\ \overline{C}\ r_2\ G\ r_3\ T'\ r_4\ T'\ \overline{F}\ cxt\ \underline{C}\ \underline{G}\ \widehat{F}\ p_1\ \widetilde{T}\ p_2\ p_3\ p_4\ \widetilde{G}.$$

The original label can then be reconstructed just applying the function *unflat*$(\cdot)$ to the string above.

In [7] we have shown a more complex example, by modeling a RS of a regulatory network for *lac* operon, presented in [3].

6. Bio-simulation

The classical notion of bisimulation for process algebras equates two processes when one process can simulate all the instructions executed by the other one and viceversa. In its weak formulation, internal instructions, i.e. non visible by external observers, are abstracted away. There are many variants of the bisimulation for process algebras, for example the barbed bisimulation [21] only considers the execution of invisible actions, and then equates two processes when they expose the same prefixes; for the mobile ambients [11], a process algebra equipped with a reduction semantics, a notion of behavioural equivalence equates two processes when they expose the same ambients [22].

There are some previous works based on bisimulation applied to models for biological systems. Barbuti et al [23] define a classical setting for bisimulation for two formalisms: the Calculus of Looping Sequences, which is a rewriting system, and the Brane Calculi, which is based on process calculi. Bisimulation is used to verify properties of the regulation of lactose degradation in Escherichia coli and the EGF signalling pathway. These calculi allow the authors to model membranes' behaviour. Cardelli et al [24] present two quantitative behavioral equivalences over species of a chemical reaction network with semantics based on ordinary differential equations. Bisimulation identifies a partition where each equivalence class represents the exact sum of the concentrations of the species belonging to that class. Bisimulation also relates species that have identical solutions at all time points when starting from the same initial conditions. Both the fore mentioned formalisms [23, 24] adopt a classical approach to bisimulation. Albeit the bisimulation is a powerful tool for verifying if the behaviour of two different software programs is indistinguishable, in the case of biological systems the classical bisimulation seems to be inappropriate, as the labels of the transitions systems are too concrete. In fact, in a biological soup, a high number of interactions occur at every computational stage, and generally, biologists are only interested to analyse a small subset of them and to focus just on some entities.

For this reason, we propose an alternative notion of bisimulation, that hereafter we call *bio-simulation*, that allows us to compare two biological systems by restricting the observation to only a limited set of events of interest, which can be chosen according to the property one wants to investigate in an experiment. This allows one to tailor the equivalence to different applications and purposes.

The transition labels of our systems record detailed information about all the reactions that have been applied in one transition, about the elements that acted as reagents, as inhibitors or that have been produced, or that have been provided by the context. All these information are stored in the label because they are necessary to compose a transition in a modular way. Depending on the application, only a suitable abstraction over the label can be of interest.

In a way, we want to query our transition labels to extract only the information we care about. To this goal, we introduce a simple language that allows us to formulate detailed and partial queries about what happened in a single transition.

Example 20. For instance we would like to express properties about each step of the bio-simulation of a system like the ones below:

1. Has the entity s_i been used by reaction r_j as reagent?
2. Has the entity s_i been blocked the application of reaction r_j?
3. Has the entity s_i been produced by reaction r_j?
4. Has the entity s_i been produced by some reaction?
5. Has the entity s_i been provided by the context?
6. Has the reaction r_j not been applied?

As detailed before, in the following we assume that: (i) the context can be non-deterministic, otherwise it makes little sense to rely on bisimulation to observe the branching structure of system dynamics; (ii) we are interested in observing the names of the entities involved in the transitions and also the reactions that have been applied, thus we assume top level restrictions are absent and rely on solid transitions only (with leftmost and rightmost silent actions).

6.1. Assertion language

Next, we introduce an assertion language that operates on strings and that combines regular expression operators with conjunction and disjunction. We let *names* be the set of symbolic names used in our assertion language.

Definition 21 (Assertion Language). Atomic assertions ζ and general assertions F are built from the syntax below:

$$\begin{array}{rcl} \zeta & ::= & \eta \ \mid \ ? \ \mid \ [N] \\ \mathsf{F} & ::= & \epsilon \ \mid \ \zeta \ \mid \ \mathsf{F} :: \mathsf{F} \ \mid \ \mathsf{F}^{+} \ \mid \ \mathsf{F}^{*} \ \mid \ \mathsf{F} \vee \mathsf{F} \ \mid \ \mathsf{F} \wedge \mathsf{F} \end{array}$$

where $\eta \in \mathit{names}$ and $N \subseteq \mathit{names}$.

Roughly, an atomic assertion ζ denotes either a string composed by a single name η (one of the symbols in the set *names* for denoting a particular entity, reaction, production or context), or the wildcard ? that stands for any symbol, or the pattern $[N]$ that stands for any string composed by a single symbol in the set N. Clearly ? is just a shorthand for $[\mathit{names}]$. We write $\mathbf{0}$ for $[\emptyset]$ and $[s_1, ..., s_n]$ instead of $[\{s_1, ..., s_n\}]$.

An assertion F is either the empty string ϵ, an atomic assertion ζ, the concatenation of two assertions $\mathsf{F}_1 :: \mathsf{F}_2$, the replication of F for 1 or more times F^{+}, the replication of F for 0 or more times F^{*}, the disjunction of two assertions $\mathsf{F}_1 \vee \mathsf{F}_2$ or their conjunction $\mathsf{F}_1 \wedge \mathsf{F}_2$. We denote by $\star$ the assertion $?^{*}$.

An assertion denotes a set of strings over the alphabet *names* as expected. Below we let $\wp(X)$ denote the powerset of a set X.

Definition 22 (Semantics of Assertions). We define $[\![\mathsf{F}]\!] \subseteq \wp(\mathit{names}^{*})$ by induction on the structure of F:

$$\begin{array}{rclcrcl} [\![\epsilon]\!] & \triangleq & \{\epsilon\} & \quad & [\![\mathsf{F}_1 :: \mathsf{F}_2]\!] & \triangleq & \{\omega_1 :: \omega_2 \ \mid \ \omega_1 \in [\![\mathsf{F}_1]\!] \wedge \omega_2 \in [\![\mathsf{F}_2]\!]\} \\ [\![\alpha]\!] & \triangleq & \{\alpha\} & & [\![\mathsf{F}^{+}]\!] & \triangleq & [\![\mathsf{F}]\!]^{+} \\ [\![?]\!] & \triangleq & \mathit{names} & & [\![\mathsf{F}^{*}]\!] & \triangleq & [\![\mathsf{F}]\!]^{*} \\ [\![[N]]\!] & \triangleq & N & & [\![\mathsf{F}_1 \vee \mathsf{F}_2]\!] & \triangleq & [\![\mathsf{F}_1]\!] \cup [\![\mathsf{F}_2]\!] \\ & & & & [\![\mathsf{F}_1 \wedge \mathsf{F}_2]\!] & \triangleq & [\![\mathsf{F}_1]\!] \cap [\![\mathsf{F}_2]\!] \end{array}$$

Definition 23 (Satisfaction as Membership). Let υ be a transition label, and F be an assertion. We write $\upsilon \models \mathsf{F}$ (read as the transition label υ satisfies the assertion F) if $\mathit{flat}(\upsilon) \in [\![\mathsf{F}]\!]$, otherwise we write $\upsilon \not\models \mathsf{F}$ (or also $\upsilon \models \neg\mathsf{F}$) and say that F does not hold at υ.

Given two transition labels υ, w we write $\upsilon \equiv_{\mathsf{F}} w$ if $\upsilon \models \mathsf{F} \ \Leftrightarrow\ w \models \mathsf{F}$, i.e. if both υ, w satisfy F or they do not.

Example 24. The assertions corresponding to the sample queries listed in Example 20 are as follows:

1. $\star :: r_j :: [s_1, ..., s_n]^* :: s_i :: [s_1, ..., s_n]^* :: [\overline{s}_1, ..., \overline{s}_n]^+ :: \star$
2. $\star :: r_j :: [s_i, \overline{s}_i] :: r_{j+1} :: \star$
3. $\star :: p_j :: [\mathit{ents}]^* :: \widetilde{s}_i :: \star$
4. $\star :: \widetilde{s}_i :: \star$
5. $\star :: \widehat{s}_i :: \star$
6. $\star :: r_j ::? :: r_{j+1} :: \star$

where in 1, 2, 6 we exploit the fact that in a reaction (R, I, P) the sets R of reactants and I of inhibitors are non empty, so that if there is only one symbol between the occurrence of r_j and r_{j+1} it means the reaction r_j has not been applied. Viceversa, if the reaction r_j has been applied the occurrence of r_j must be followed by at least one of the symbols in $\{s_1, ..., s_n\}$ and then by at least one of the symbols in $\{\overline{s}_1, ..., \overline{s}_n\}$.

6.2. Bio-similarity and bio-logical equivalence

The notion of bio-simulation builds on the above language of assertions to parameterize the induced equivalence on the property of interest. Please recall that we have defined the behaviour of the context in a non deterministic way, thus at each step, different possible sets of entities can be provided to the system and different sets of reactions can be enabled/disabled. Bio-simulation can thus be used to compare the behaviour of different systems that share some of the reactions or entities or also to compare the behaviour of the same set of reactions when different contexts are provided.

Definition 25 (Bio-similarity $\sim_{\mathsf{F}}$). Given an assertion F, a *bio-simulation* $\mathbf{R}_{\mathsf{F}}$ that respects F is a binary relation over cCNA processes such that, whenever $P\ \mathbf{R}_{\mathsf{F}}\ Q$ then:

- for any υ, P' such that $P \stackrel{\upsilon}{\Rightarrow} P'$ then there exist w, Q' such that $Q \stackrel{w}{\Rightarrow} Q'$ with $\upsilon \equiv_{\mathsf{F}} w$ and $P'\ \mathbf{R}_{\mathsf{F}}\ Q'$.
- for any w, Q' such that $Q \stackrel{w}{\Rightarrow} Q'$ then there exist υ, P' such that $P \stackrel{\upsilon}{\Rightarrow} P'$ with $\upsilon \equiv_{\mathsf{F}} w$ and $P'\ \mathbf{R}_{\mathsf{F}}\ Q'$.

We let $\sim_{\mathsf{F}}$ denote the largest bio-simulation and we say that P is *bio-similar* to Q, with respect to F, if $P \sim_{\mathsf{F}} Q$.

Remark 26. Please remember that the notation $P \overset{v}{\Rightarrow} P'$ refers to ordinary transitions $P \xrightarrow{v} P'$ where v is a complete chain (solid and with τ actions at the extremes). The double arrow notation should not be confused with the notation for weak transitions commonly found in the literature on process algebras.

Remark 27. An alternative way to look at a bio-simulation that respects F is to define it as an ordinary bisimulation over the transition system labelled over $\{\mathsf{F}, \neg\mathsf{F}\}$ obtained by transforming each transition $P \overset{v}{\Rightarrow} P'$ such that $v \models \mathsf{F}$ into $P \xrightarrow{\mathsf{F}} P'$ and each transition $P \overset{v}{\Rightarrow} P'$ such that $v \not\models \mathsf{F}$ into $P \xrightarrow{\neg\mathsf{F}} P'$.

It can be easily shown that the identity relation is a bio-simulation and that bio-simulations are closed under (relational) inverse, composition and union and that, as a consequence, bio-similarity is an equivalence relation.

Now, we introduce a slightly modified version of the Hennessy Milner Logic (HML) [15], called bioHML; due to the reasons we explained above, we do not want to look at the complete transition labels, thus we rely on our simple assertion language to make it parametric w.r.t the assertion F of interest:

Definition 28 (BioHML). Let F be an assertion, then the set of bioHML formulas G that respects F are built by the following syntax:

$$\begin{array}{rcl} \chi & ::= & \mathsf{F} \mid \neg\mathsf{F} \\ \mathsf{G}, \mathsf{H} & ::= & \mathtt{t} \mid \mathtt{f} \mid \mathsf{G} \wedge \mathsf{G} \mid \mathsf{G} \vee \mathsf{G} \mid \langle\chi\rangle\mathsf{G} \mid [\chi]\mathsf{G} \end{array}$$

Remark 29. An alternative way to look at bioHML formulas is as ordinary HML formulas over the set of labels $\{\mathsf{F}, \neg\mathsf{F}\}$.

As usual, the semantics of a bioHML formula is the set of cCNA processes that satisfy it.

Definition 30 (Semantics of BioHML). We define $[\![\mathsf{G}]\!] \subseteq \mathcal{P}$ by induction on the structure of G:

$$\begin{array}{rclcrcl} [\![\mathtt{t}]\!] & \triangleq & \mathcal{P} & \quad & [\![\mathsf{G} \wedge \mathsf{H}]\!] & \triangleq & [\![\mathsf{G}]\!] \cap [\![\mathsf{H}]\!] \\ [\![\mathtt{f}]\!] & \triangleq & \emptyset & & [\![\mathsf{G} \vee \mathsf{H}]\!] & \triangleq & [\![\mathsf{G}]\!] \cup [\![\mathsf{H}]\!] \end{array}$$

$$\begin{array}{rcl} [\![\langle\chi\rangle\mathsf{G}]\!] & \triangleq & \{P \in \mathcal{P} : \exists v, P'.\ P \overset{v}{\Rightarrow} P' \text{ with } v \models \chi \text{ and } P' \in [\![\mathsf{G}]\!]\} \\ [\![[\chi]\mathsf{G}]\!] & \triangleq & \{P \in \mathcal{P} : \forall v, P'.\ P \overset{v}{\Rightarrow} P' \text{ implies } v \models \chi \text{ and } P' \in [\![\mathsf{G}]\!]\} \end{array}$$

We write $P \models \mathsf{G}$ (P satisfies G) if and only if $P \in [\![\mathsf{G}]\!]$.

Negation is not included in the syntax, but the converse $\overline{\mathsf{G}}$ of a bioHML formula G can be easily defined inductively in the same way as for HML logic.

Definition 31 (Converse). Given a bioHML formula G we define its converse $\overline{\mathsf{G}}$ as follows:

$$\begin{array}{rclcrclcrcl} \overline{\mathtt{t}} & \triangleq & \mathtt{f} & \quad & \overline{\mathsf{G} \wedge \mathsf{H}} & \triangleq & \overline{\mathsf{G}} \vee \overline{\mathsf{H}} & \quad & \overline{\langle\chi\rangle\mathsf{G}} & \triangleq & [\chi]\overline{\mathsf{G}} \\ \overline{\mathtt{f}} & \triangleq & \mathtt{t} & & \overline{\mathsf{G} \vee \mathsf{H}} & \triangleq & \overline{\mathsf{G}} \wedge \overline{\mathsf{H}} & & \overline{[\chi]\mathsf{G}} & \triangleq & \langle\chi\rangle\overline{\mathsf{G}} \end{array}$$

We observe that, as expected, for any bioHML formula G and process P we have $\overline{\overline{\mathsf{G}}} = \mathsf{G}$ and $P \models \overline{\mathsf{G}}$ iff $P \not\models \mathsf{G}$.

Definition 32 (Bio-logical equivalence). We let $\mathcal{L}_{\mathsf{F}}$ be the set of all bioHML formulas that respects F and we say that two processes P, Q are *bio-logically equivalent w.r.t.* F, written $P \equiv_{\mathcal{L}_{\mathsf{F}}} Q$, when P and Q satisfy exactly the same bioHML formulas in $\mathcal{L}_{\mathsf{F}}$, i.e. when for any $\mathsf{G} \in \mathcal{L}_{\mathsf{F}}$ we have $P \models \mathsf{G} \;\Leftrightarrow Q \models \mathsf{G}$.

Finally, we extend the classical result establishing the correspondence between the logical equivalence induced by HML with bisimilarity for proving that bio-similarity coincides with bio-logical equivalence.

Theorem 33 (Correspondence). $\sim_{\mathsf{F}} \, = \, \equiv_{\mathcal{L}_{\mathsf{F}}}$

6.3. Bio-simulation at work

We will show how bio-simulation works. For the sake of space, we consider a very simple example with only two reactions. Reaction P_1 requires G to produce C; reaction P_2 requires C to produce G; both reactions have H as inhibitor. Now, we set two systems defined by the same two reactions, with the two different initial configuration, and with two different context definitions. The two reactions work as follows (where we omit to specify the cases where H is present, as they will never happen):

$$\begin{array}{rcl}
P_1 & \triangleq & {}^{\tau}\backslash^{\square}_{G_i}\backslash^{G_o}_{\square}\backslash^{\square}_{\overline{H}_i}\backslash^{\overline{H}_o}_{\square}\backslash^{\square}_{r_2}\backslash^{p_1}_{\square}\backslash^{\square}_{\widetilde{C}_i}\backslash^{\widetilde{C}_o}_{\square}\backslash_{p_2}.P_1 \quad + \quad {}^{\tau}\backslash^{\square}_{\overline{G}_i}\backslash^{\overline{G}_o}_{\square}\backslash^{\square}_{r_2}\backslash^{p_1}_{\square}\backslash_{p_2}.P_1 \\
P_2 & \triangleq & {}^{r_2}\backslash^{\square}_{C_i}\backslash^{C_o}_{\square}\backslash^{\square}_{\overline{H}_i}\backslash^{\overline{H}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_2}_{\square}\backslash^{\square}_{\widetilde{G}_i}\backslash^{\widetilde{G}_o}_{\square}\backslash_{\tau}.P_2 \quad + \quad {}^{r_2}\backslash^{\square}_{\overline{C}_i}\backslash^{\overline{C}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_2}_{\square}\backslash_{\tau}.P_2
\end{array}$$

The two contexts follow :

$$Cxt \triangleq {}^{cxt}\backslash^{\square}_{\widehat{C}_i}\backslash^{\widehat{C}_o}_{\square}\backslash^{\square}_{\underline{G}_i}\backslash^{\underline{G}_o}_{\square}\backslash^{\square}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\square}\backslash_{p_1}.Cxt + {}^{cxt}\backslash^{\square}_{\underline{C}_i}\backslash^{\underline{C}_o}_{\square}\backslash^{\square}_{\underline{G}_i}\backslash^{\underline{G}_o}_{\square}\backslash^{\square}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\square}\backslash_{p_1}.Cxt$$

$$Cxt' \triangleq {}^{cxt}\backslash^{\square}_{\widehat{G}_i}\backslash^{\widehat{G}_o}_{\square}\backslash^{\square}_{\underline{C}_i}\backslash^{\underline{C}_o}_{\square}\backslash^{\square}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\square}\backslash_{p_1}.Cxt' + {}^{cxt}\backslash^{\square}_{\underline{G}_i}\backslash^{\underline{G}_o}_{\square}\backslash^{\square}_{\underline{C}_i}\backslash^{\underline{C}_o}_{\square}\backslash^{\square}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\square}\backslash_{p_1}.Cxt'$$

The definition of the processes encoding G and C is similar, and it is given in a parametric way:

$$P(G) \;\triangleq\; {}^{G_i}\backslash_{G_o}.\overline{P}(G) \qquad\qquad \overline{P}(G) \;\triangleq\; {}^{\overline{G}_i}\backslash^{\square}_{\overline{G}_o}\backslash^{\widetilde{G}_i}_{\square}\backslash_{\widetilde{G}_o}.P(G)$$

and we have $P_G \triangleq P(G)$, $P_C \triangleq P(C)$, then $\overline{P}_H \triangleq {}^{\overline{H}_i}\backslash_{\overline{H}_o}.\overline{P}_H + {}^{\overline{H}_i}\backslash^{\square}_{\overline{H}_o}\backslash^{\overline{H}_i}_{\square}\backslash_{\overline{H}_i}.\overline{P}_H$, as H is neither produced nor provided by the context. Then, the initial configuration of system Sys_1 includes G and not C and the context can only provide C, the initial configuration of system Sys_2 includes C and not G and the context can only provide G:

$$\begin{array}{rcl}
Sys_1 & \triangleq & I \mid P_1 \mid P_2 \mid P_G \mid \overline{P}_C \mid \overline{P}_H \mid Cxt \\
Sys_2 & \triangleq & I \mid P_1 \mid P_2 \mid \overline{P}_G \mid P_C \mid \overline{P}_H \mid Cxt'
\end{array}$$

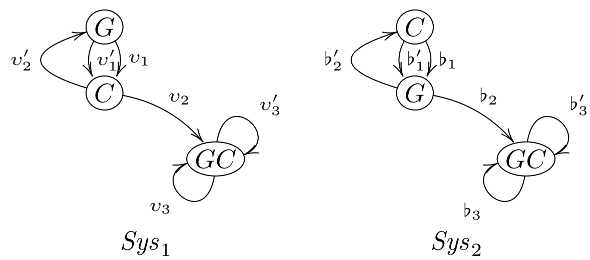

Figure 4: The operational semantics of Sys_1 and Sys_2.

In Figure 4 we show the operational semantics of Sys_1 (on the left) and Sys_2 (on the right), limited to the transitions with complete and solid labels. To improve readability, we named the states of the transition system with the entities that are present in the state. For example, in the leftmost figure, since in Sys_1 only the entity G is available, we name the topmost state G instead of Sys_1. Similarly, for the rightmost figure, where we write, e.g. C instead of Sys_2. As before we show the output of the $\mathit{flat}(\cdot)$ function applied to the transition labels:

$$\begin{array}{lcl}
\mathit{flat}(\upsilon_1) & \triangleq & r_1\ G\ \overline{H}\ r_2\ \overline{C}\ \mathit{cxt}\ \widehat{C}\ \underline{G}\ \underline{H}\ p_1\ \widetilde{C}\ p_2 \\
\mathit{flat}(\upsilon_1') & \triangleq & r_1\ G\ \overline{H}\ r_2\ \overline{C}\ \mathit{cxt}\ \underline{C}\ \underline{G}\ \underline{H}\ p_1\ \widetilde{C}\ p_2 \\[6pt]
\mathit{flat}(\upsilon_2) & \triangleq & r_1\ \overline{G}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \widehat{C}\ \underline{G}\ \underline{H}\ p_1\ p_2\ \widetilde{G} \\
\mathit{flat}(\upsilon_2') & \triangleq & r_1\ \overline{G}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \underline{C}\ \underline{G}\ \underline{H}\ p_1\ p_2\ \widetilde{G} \\[6pt]
\mathit{flat}(\upsilon_3) & \triangleq & r_1\ G\ \overline{H}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \widehat{C}\ \underline{G}\ \overline{H}\ p_1\ \widetilde{C}\ p_2\ \widetilde{G} \\
\mathit{flat}(\upsilon_3') & \triangleq & r_1\ G\ \overline{H}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \underline{C}\ \underline{G}\ \overline{H}\ p_1\ \widetilde{C}\ p_2\ \widetilde{G}
\end{array}$$

The labels $\flat_i$ $\flat_i'$, with $i \in \{1,2,3\}$ can be obtained by labels υ_i, υ_i' by substituting G with C and viceversa.

Now, it easy to check that Sys_1 and Sys_2 are bio-similar w.r.t the property F saying that G and C are simultaneously produced, formally: $Sys_1 \sim_{\mathsf{F}} Sys_2$ with $\mathsf{F} = \star :: \widetilde{G} :: \star \wedge \star :: \widetilde{C} :: \star$.

On the contrary, $Sys_1 \not\sim_{\mathsf{F}'} Sys_2$ with $\mathsf{F}' = \star :: \widetilde{C} :: \star$, because it happens that both transition labels ℓ_1 and ℓ_1', in Sys_1, record the production of C, whereas transition labels $\flat_1$ and $\flat_1'$ do not. In fact, the bioHML formula $\mathsf{G} \triangleq \langle \mathsf{F}' \rangle \mathtt{t}$ can be used to distinguish Sys_1 from Sys_2, as $Sys_1 \models \mathsf{G}$ and $Sys_2 \not\models \mathsf{G}$.

7. Towards Enhanced Reaction Systems

Our encoding increases the expressivity of RS concerning: the possibility of alternative behaviour of mutated entities, and the communication between two different RSs. It is important to note that, when the context is deterministic, our encoding guarantees that from each state, in the cCNA transition system, only one state is reachable, as the dynamics is totally deterministic.

7.1. Mutating entities

In RS, when an entity is present, it can potentially be involved in each reactions where it is required. With a few more lines of code, in cCNA it is possible to describe the behaviour of a mutation of an entity, in a way that the mutated version of the entity can take part to only a subset of the reactions requiring the *normal version* of the entity. For example, let us assume that entity $s1$ is consumed by reactions $a1$ and $a2$. Reaction $a1$ produces also $s1$ if $s2$ is present, otherwise $a1$ produces a mutated version of $s1$, say $s1'$. When $s1'$ is produced, reaction $a2$ behaves in the same way as if $s1$ would be absent, whereas $a2$ recognises the presence of $s1'$ and behaves in the same way as if $s1$ would be present. Technically, in both cases it is enough to add one more nondeterministic choice in the code of P_{a1} and P_{a2}.

7.2. Communicating Reaction Systems

We sketch how it is possible to program two RSs encodings, in a way that the entities that usually come from the context of one RS will be provided instead from the other RS.

Example 34. Let $rs1$ and $rs2$ be two RSs, defined, respectively, by the reactions $a_1 = (s, \emptyset, x)$ and $a_2 = (y, \emptyset, s)$. Now, we set our example such that the two contexts, for $rs1$ and $rs2$, do not provide any entities. We also assume that entity s in $rs1$ is provided by $rs2$, as $rs2$ produces a quantity of s that is enough for $rs1$ and $rs2$. For technical reasons, we can not use the same name for s in both the two RSs, then we use the name ss in $rs2$. We need to modify our encoding technique to suit this new setting. As we do not model contexts, we introduce *dummy* channel names dx and dss to model the absence of entities. Also, thanks to the simplicity of the example, we can leave out the use of the p_i channels. This streamlining does not affect the programming technique we propose to make two RSs communicate. First, we translate the reaction in $rs1$, by setting $[\![a_1]\!] \triangleq P_{a_1}$ where:

$$P_{a_1} \quad \triangleq \quad {}^{\tau}\backslash^{\square}_{s_i}\backslash^{s_o}_{\square}\backslash^{\square}_{\widetilde{x}_i}\backslash^{\widetilde{x}_o}_{\square}\backslash_{a_2}.P_{a_1} + {}^{\tau}\backslash^{\square}_{\overline{s}_i}\backslash^{\overline{s}_o}_{\square}\backslash^{\square}_{dx_i}\backslash^{dx_o}_{\square}\backslash_{a_2}.P_{a_1}$$

Please note, that prefixes of process P_{a_1} end with the channel name a_2, as the link chain is now connected with the reaction of $rs2$. The encoding for the entities is given by setting $[\![s]\!] \triangleq P_s$ and $[\![x]\!] \triangleq P_x$, where:

$$\begin{array}{rcl}
P_s & \triangleq & {}^{s_i}\backslash^{\square}_{s_o}\backslash^{\widehat{s}_i}_{\square}\backslash_{\widehat{s}_o}.P_s + {}^{s_i}\backslash_{s_o}.\overline{P_s} \\
\overline{P_s} & \triangleq & {}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash^{\widehat{s}_i}_{\square}\backslash_{\widehat{s}_o}.P_s + {}^{\overline{s}_i}\backslash_{\overline{s}_i}.\overline{P_s} \\
P_x & \triangleq & {}^{\widetilde{x}_i}\backslash_{\widetilde{x}_o}.P_x + {}^{dx_i}\backslash_{dx_o}.\overline{P_x} \\
\overline{P_x} & \triangleq & {}^{\widetilde{x}_i}\backslash_{\widetilde{x}_o}.P_x + {}^{dx_i}\backslash_{dx_o}.\overline{P_x}
\end{array}$$

The encoding for $rs2$ is given by $[\![a_2]\!] \triangleq P_{a_2}$, where:

$$P_{a_2} \quad \triangleq \quad {}^{a_2}\backslash^{\square}_{y_i}\backslash^{y_o}_{\square}\backslash^{\square}_{\widetilde{ss}_i}\backslash^{\widetilde{ss}_o}_{\square}\backslash_{\tau}.P_{a_2} + {}^{a_2}\backslash^{\square}_{\overline{y}_i}\backslash^{\overline{y}_o}_{\square}\backslash^{\square}_{dss_i}\backslash^{dss_o}_{\square}\backslash_{\tau}.P_{a_2}$$

In the encoding of the entities in $rs2$, we introduce the mechanism that allows the entity s (ss in $rs2$) to be provided in $rs1$. Every time ss is produced in $rs2$, a virtual link is created to synchronise with $rs1$ on link ${}^{\widehat{s}_i}\backslash_{\widehat{s}_o}$. To this purpose we define $[\![ss]\!] \triangleq P_{ss}$ and $[\![y]\!] \triangleq P_y$, where:

$$\begin{array}{lcl}
P_{ss} & \triangleq & {}^{\widetilde{ss}_i}\backslash^{\square}_{\widehat{s}_i}\backslash^{\widehat{s}_o}_{\square}\backslash_{\widetilde{ss}_o}.P_{ss} + {}^{dss_i}\backslash_{dss_o}.\overline{P_{ss}} \\
\overline{P_{ss}} & \triangleq & {}^{\color{red}\widetilde{ss}_i}\backslash^{\color{blue}\square}_{\color{red}\widehat{s}_i}\backslash^{\color{red}\widehat{s}_o}_{\color{blue}\square}\backslash_{\color{red}\widetilde{ss}_o}.P_{ss} + {}^{dss_i}\backslash_{dss_o}.\overline{P_{ss}} \\
P_y & \triangleq & {}^{\color{gray}y_i}\backslash_{\color{gray}y_o}.\overline{P_y} \\
\overline{P_y} & \triangleq & {}^{\overline{y}_i}\backslash_{\overline{y}_o}.\overline{P_y}
\end{array}$$

We now assume that the initial system is $S \triangleq (\nu\, names)(P_{a_1}|P_{a_2}|P_s|P_y|\overline{P_x}|\overline{P_{ss}})$, i.e. only entities s and y are present. Now, the only possible transition has the following label (that we report without restriction):

$${}^{\tau}\backslash^{\color{blue}s_i}_{s_i}\backslash^{s_o}_{\color{blue}s_o}\backslash^{\color{gray}\widetilde{x}_i}_{\widetilde{x}_i}\backslash^{\widetilde{x}_o}_{\color{gray}\widetilde{x}_o}\backslash^{a_2}_{a_2}\backslash^{\color{gray}y_i}_{y_i}\backslash^{y_o}_{\color{gray}y_o}\backslash^{\color{red}\widetilde{ss}_i}_{\widetilde{ss}_i}\backslash^{\color{blue}\widehat{s}_i}_{\color{red}\widehat{s}_i}\backslash^{\color{red}\widehat{s}_o}_{\color{blue}\widehat{s}_o}\backslash^{\widetilde{ss}_o}_{\color{red}\widetilde{ss}_o}\backslash_{\tau},$$

where the black links belong to the prefixes of P_{a_1}, and P_{a_2}, the blue links belong to P_s, the gray links belong to P_y, and $\overline{P_x}$ and the red links belong to $\overline{P_{ss}}$. After the execution, the entity s is still present in $rs1$ as it has been provided by $rs2$.

As we have briefly sketched, our model of two *communicating Reaction Systems* can enable the study of the behaviour of one RS in relation to another one. Thus, the products of the reactions of one RS can become the input for another one. This could allow for a modular approach to modeling complex systems, by composing different Reaction Systems.

8. Conclusion

In this paper we have introduced cCNA, that generalises CNA by allowing the use of prefixes that are link chains and not just single links. This extension was initially described in the future work section of [8]. Thanks to this enhancement, cCNA allowed us to define a faithful encoding of Reaction Systems in a process algebraic framework. This encoding shows several benefits. First, contexts of RSs can be easily defined recursively and exhibit non deterministic behaviour. Second, the operational semantics is defined in a compositional way by a set of SOS inference rules. Third, we have defined a new assertion language, which allows us to specify the properties to be verified over the labels of the operational semantics. Assertions can be used to tailor the classical notion of bisimilarity and Hennessy-Milner logic to focus on some particular aspects or experiments. We have called *bio-similarity* the induced notion of equivalence.

We are currently investigating how to integrate our methodology with other formal techniques to prove properties of the modeled systems, along the lines in [25, 26, 27]. Moreover, we are considering possible enhancements of RSs based on entity mutation and on the possibility for two RSs to exchange entities.

As future work, we plan to implement a prototype of our embedding, with an automatic translation from RSs to `link`-calculus, so to exploit the implementation of the symbolic semantics of the `link`-calculus [28] that can be found in [29].

References

[1] R. Brijder, A. Ehrenfeucht, M. Main, G. Rozenberg, A tour of reaction systems, International Journal of Foundations of Computer Science 22 (07) (2011) 1499–1517.

[2] S. Azimi, B. Iancu, I. Petre, Reaction system models for the heat shock response, Fundamenta Informaticae 131 (3-4) (2014) 299–312.

[3] L. Corolli, C. Maj, F. Marinia, D. Besozzi, G. Mauri, An excursion in reaction systems: From computer science to biology, Theoretical Computer Science 454 (2012) 95–108.

[4] S. Azimi, Steady states of constrained reaction systems, Theor. Comput. Sci. 701 (C) (2017) 20–26.

[5] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Investigating dynamic causalities in reaction systems, Theor. Comput. Sci. 623 (2016) 114–145.

[6] F. Okubo, T. Yokomori, The computational capability of chemical reaction automata, Natural Computing 15 (2) (2016) 215–224.

[7] L. Brodo, R. Bruni, M. Falaschi, Enhancing reaction systems: A process algebraic approach, in: M. Alvim, K. Chatzikokolakis, C. Olarte, F. Valencia (Eds.), The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy, Vol. 11760 of Lecture Notes in Computer Science, Springer Berlin, 2019, pp. 68–85.

[8] C. Bodei, L. Brodo, R. Bruni, A formal approach to open multiparty interactions, Theoretical Computer Science 763 (2019) 38–65.

[9] C. Bodei, L. Brodo, R. Bruni, Open multiparty interaction, in: Recent Trends in Algebraic Development Techniques, 21st International Workshop, WADT 2012, Vol. 7841 of Lecture Notes in Computer Science, Springer, 2012, pp. 1–23.

[10] C. Bodei, L. Brodo, R. Bruni, The link-calculus for open multiparty interactions, Information and Computation (2020) 104587.

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

[12] G. Ciobanu, V. A. Zakharov, Encoding mobile ambients into the π-calculus, in: Perspectives of Systems Informatics, Vol. 4378, Springer Berlin Heidelberg, 2007, pp. 148–165.

[13] L. Brodo, On the expressiveness of pi-calculus for encoding mobile ambients, Mathematical Structures in Computer Science 28 (2) (2018) 202–240.

[14] C. Bodei, L. Brodo, R. Bruni, D. Chiarugi, A flat process calculus for nested membrane interactions, Sci. Ann. Comp. Sci. 24 (1) (2014) 91–136.

[15] M. Hennessy, R. Milner, On observing nondeterminism and concurrency, in: J. de Bakker, J. van Leeuwen (Eds.), Automata, Languages and Programming, Vol. 85 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 1980, pp. 299–309.

[16] A. Bernini, L. Brodo, P. Degano, M. Falaschi, D. Hermith, Process calculi for biological processes, Natural Computing 17 (2) (2018) 345–373.

[17] B. Aman, G. Ciobanu, Controlled reversibility in reaction systems, in: M. Gheorghe, G. Rozenberg, A. Salomaa, Z. Claudio (Eds.), Membrane Computing, Springer International Publishing, 2018, pp. 40–53.

[18] P. Bottoni, A. Labella, G. Rozenberg, Networks of reaction systems, International Journal of Foundations of Computer Science 31 (01) (2020) 53–71.

[19] A. Męski, W. Penczek, G. Rozenberg, Model checking temporal properties of reaction systems, Information Sciences 313 (2015) 22–42.

[20] J. D. Watson, T. A. Baker, S. P. Bell, Molecular Biology of the Gene, Pearson Education, USA, 2013.

[21] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 685–695.

[22] A. Gordon, L. Cardelli, Equational properties of mobile ambients, Mathematical Structures in Computer Science 13 (3) (2003) 371–408.

[23] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Bisimulations in calculi modelling membranes, Formal Aspects of Computing 20 (4) (2008) 351–377.

[24] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Forward and backward bisimulations for chemical reaction networks, in: 26th International Conference on Concurrency Theory, CONCUR 2015, Vol. 42, Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2015, pp. 226–239.

[25] D. Chiarugi, M. Falaschi, D. Hermith, C. Olarte, L. Torella, Modelling non-markovian dynamics in biochemical reactions, BMC Systems Biology 9 (S-3) (2015) S8.

[26] C. Olarte, D. Chiarugi, M. Falaschi, D. Hermith, A proof theoretic view of spatial and temporal dependencies in biochemical systems, Theor. Comput. Sci. 641 (2016) 25–42.

[27] C. Bodei, L. Brodo, R. Gori, F. Levi, A. Bernini, D. Hermith, A static analysis for Brane Calculi providing global occurrence counting information, Theoretical Computer Science 696 (2017) 11–51.

[28] L. Brodo, C. Olarte, Symbolic semantics for multiparty interactions in the link-calculus, in: Proc. of SOFSEM'17, Vol. 10139 of Lecture Notes in Computer Science, Springer, 2017, pp. 62–75.

[29] C. Olarte, SiLVer: Symbolic links verifier, http://subsell.logic.at/links/links-web/index.html (Dec. 2018).

Appendix A. Omitted Proofs

In this section we report the proofs for the results in Section 4 and in Section 6.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$. Let $P = [\![\mathcal{A}, \gamma]\!]$ its cCNA encoding. If exists P' such that $P \xrightarrow{(\nu\, names)\upsilon} P'$ is a transition of P, then*

1. *for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in υ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in υ; the channel cxt appears in υ;*
2. *for each reaction $aj \in A$ and each virtual link offered by processes P_a and Cxt_1 is overlapped by exactly one solid link offered by processes representing entities.*

Proof. We prove the two items separately:

1. by Definition 11, all the names that appear in the prefixes of any subprocess are in the set *names* and thus restricted. They include all the reaction names r_j, all the production names p_j, all the entity names s_i, s_o, in all their decorated versions, and the name cxt. Therefore the chain υ must start and end with a τ action and cannot contain virtual links. The only prefix that starts with τ is the one of the recursive init process I (prefix ${}^{\tau}\backslash_{r_1}$) and the only prefixes that end with τ are those associated to reaction a_u (as we have assumed that $p_{u+1} = \tau$, where u is the number of reactions). Then each prefix that starts with r_j involves p_j and r_{j+1}. Thus all the prefixes associated with reactions must be concatenated and also the prefix associated with the context (remember that $r_{u+1} = cxt$), forming the backbone of the label. Since the context processes is involved, then all entities processes are also involved. Then, all the processes I, P_a, P_s (or $\overline{P_s}$), and Cxt_1 must participate to each transition.
2. for each reaction $aj \in A$, the cCNA code of P_{aj} leaves one virtual link between two solid links of the types ${}^{r_j}\backslash_{\ldots}\backslash^{\square}_{s_i}\backslash^{s_o}_{\square}\backslash_{\ldots}\backslash_{r_{j+1}} \ldots {}^{p_j}\backslash_{\ldots}\backslash_{p_{j+1}}$. Then, it derives that the process P_s, encoding the behaviour of entity s, can participate by filling the virtual link in the above transition by only offering one solid link of the form ${}^{s_i}\backslash_{s_o}$. In fact, there is no other way to generate a solid chain from s_i to s_o. The same reasoning holds for the processes Cxt_1 and for all the decorated versions of s_i, s_o.

Proposition 16 (Correctness 1). *Let $P = [\![\mathcal{A}, \gamma]\!]$ with*

$$P = (\nu\, names)\left(I \mid \prod_{a \in A} P_a \mid Cxt_1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P}_s \right).$$

If there exists P' such that $P \xrightarrow{\upsilon} P'$, it holds that:

1. $v = {}^{\tau}\backslash_{\tau} \ldots {}^{\tau}\backslash_{\tau}$, *and*
2. $P' = (\nu\ names)\big(I \mid \prod_{a \in A} P_a \mid Cxt_2 \mid \prod_{s \in C_1 \cup D_1} P_s \vert \prod_{s \notin C_1 \cup D_1} \overline{P}_s\big)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$*, we have* $P' = [\![\mathcal{A}, \gamma^1]\!]$.

Proof. First, we note that all the channels in the system are restricted, see Definition 11, then it holds that the transition labels are of the form $v = {}^{\tau}\backslash_{\tau} \ldots {}^{\tau}\backslash_{\tau}$. Now, by Definition 11 and by Lemma 12.1, all the channels r_j, p_j, with $j \in [1, \ldots, u]$, and cxt, and all the annotated versions of s_i, s_o are restricted. Also, processes Cxt_1 always requires the interaction with P_s on either on channels $\widehat{s}_i$, $\widehat{s}_o$ or on channels $\underline{s}_i$, $\underline{s}_o$. It derives that all the processes: P_a (coding the behaviour of reaction $a \in A$), P_s (coding the behaviour of entity $s \in S$), and Cxt_1 (coding the behaviour of the context regarding all the entities) have been involved in the transition.

For any process P_{aj} encoding a reaction aj we have the following cases:

(a) if aj is applicable and it produces the entity s, the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash^{\square}_{\widetilde{s}_i} \backslash^{\widetilde{s}_o}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}} . P_{aj};$$

(b) if aj is applicable and it consumes the entity s, the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{s_i} \backslash^{s_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}} . P_{aj};$$

(c) if aj is applicable and it requires the absence of the entity s, the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{\overline{s}_i} \backslash^{\overline{s}_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}} . P_{aj};$$

(d) if aj is not applicable, the process P_{aj} executes a code capturing either the absence of one of its reactants (case 1), or the presence of one of its inhibitors (case 2):

1. $P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{\overline{s}_i} \backslash^{\overline{s}_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}} . P_{aj}$;
2. $P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{s_i} \backslash^{s_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}} . P_{aj}$.

Now, we consider the structure of the process Cxt_1. By Definition 11, Cxt_1 is the unique process encoding the behaviour of the context regulating the supply of any entity.

(e) The code of the process Cxt_1 that provides s and not e has the following structure:

$$Cxt_1 \triangleq {}^{cxt}\backslash_{\ldots} \backslash^{\square}_{\widehat{s}_i} \backslash^{\widehat{s}_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{\underline{e}_i} \backslash_{\square}{}^{\underline{e}_o} \backslash_{\ldots} \backslash_{p_1} . Cxt_2.$$

The code executed by P_s has the following structure:

(f) $P_s \triangleq \sum_{h,k\geq 0}({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h\, {}^{\widehat{s}_i}\backslash^{\square}_{\widehat{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k\, .P_s$, if $s \in C_{i+1}$;

(g) $P_s \triangleq \sum_{h\geq 0,k\geq 1}({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h\; {}^{\underline{s}_i}\backslash^{\square}_{\underline{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k.P_s$, if $s \notin C_{i+1}$

(h) $P_s \triangleq \sum_{h\geq 0}({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h\, {}^{\underline{s}_i}\backslash_{\underline{s}_o}.\overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the presence of s plus possibly some reactions not requiring s; and k is the number of reactions producing s.

Similarly, the code executed by $\overline{P}_s$ has the following structure:

(f') $\overline{P}_s \triangleq \sum_{h,k\geq 0}({}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash_{\square})^h\, {}^{\widehat{s}_i}\backslash^{\square}_{\widehat{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k\, .P_s$, if $s \in C_{i+1}$;

(g') $\overline{P}_s \triangleq \sum_{h\geq 0,k\geq 1}({}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash_{\square})^h\; {}^{\underline{s}_i}\backslash^{\square}_{\underline{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k.P_s$, if $s \notin C_{i+1}$

(h') $\overline{P}_s \triangleq \sum_{h\geq 0}({}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash_{\square})^h\; {}^{\underline{s}_i}\backslash_{\underline{s}_o}.\overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the absence of s plus possibly some reactions requiring s; and k is the number of reactions producing s.

It is worth nothing that, depending on the presence (P_s) or the absence ($\overline{P}_s$) of each entity s, for each process P_a (encoding a reaction a) the choice between the execution of the reaction code (points **(a)**, **(b)**, **(c)**) or the code expressing that reaction a is not applicable (point **(d)**) is deterministic. Also, the building of the code of process *Cxt* (points **(e)**, **(f)**), is univocally determined by the evolution of γ. It derives that the trend followed by the processes P_s (or $\overline{P}_s$) is also deterministic (points **(f)**, **(g)**, **(h)** or **(f')**, **(g')**, **(h')**), leading to $P' = [\![\mathcal{A}, \gamma^1]\!]$.

Corollary 17 (Correctness 2). *Let* $P = [\![\mathcal{A}, \gamma]\!]$ *and* $j \geq 1$. *If there exists* P'' *such that* $P \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}}{}^{j}\; P''$, *then letting* $\pi^j = (\gamma^j, \delta^j)$ *we have* $P'' = [\![\mathcal{A}, \gamma^j]\!]$.

Proof. We proceed by induction on the transition number $j \geq 0$.

base case $j = 1$**:** This case falls into the case of Proposition 16.

inductive case: We assume, by inductive hypothesis, that $\exists P'$ such that

$$P \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}}{}^{j-1}\; P'$$

and $P' = [\![\mathcal{A}, \gamma^{j-1}]\!]$. As P' is the encoding of an extended interactive process, by Proposition 16, it exists P'' such that $P' \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}} P''$, and $P'' = [\![\mathcal{A}, \gamma^j]\!]$.

Proposition 18 (Completeness 1). *Let* $P = [\![\mathcal{A}, \gamma]\!]$ *and* $\pi^1 = (\gamma^1, \delta^1)$. *Then,* $P \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}} P' = [\![\mathcal{A}, \gamma^1]\!]$.

PROOF. By Proposition 16, if there exists P' such that $P \xrightarrow{\tau\backslash_\tau \cdots \tau\backslash_\tau} P'$, then the structure of P' is deterministically computed.
Now, to prove that always exists P', we observe that even in the case no reaction a is applicable in the interactive process π in A, then process P can always execute a step transition, as its subprocesses P_a can always execute one of the *alternative code for when reaction a is not applicable* (see Definition 11, code for P_a processes).

Corollary 19 (Completeness 2). *Let* $P = [\![\mathcal{A}, \gamma]\!]$ *and* $\pi^j = (\gamma^j, \delta^j)$. *Then,* $P \xrightarrow{\tau\backslash_\tau \cdots \tau\backslash_\tau}{}^{j} P'' = [\![\mathcal{A}, \gamma^j]\!]$.

PROOF. The proof proceeds by induction on the number j, and it is similar to the one of Corollary 17.

Theorem 33 (Correspondence). $\sim_{\mathsf{F}} \, = \, \equiv_{\mathcal{L}_\mathsf{F}}$

PROOF. The proof is just an adaptation of the classical result. The two implications are proved separately.

$\sim_{\mathsf{F}} \subseteq \equiv_{\mathcal{L}_\mathsf{F}}$**:** Given any two processes $P \sim_{\mathsf{F}} Q$ we need to prove that for any bioHML formula $\mathsf{G} \in \mathcal{L}_\mathsf{F}$ we have $P \models \mathsf{G}$ iff $Q \models \mathsf{G}$. Without loss of generality, we prove that $P \models \mathsf{G}$ implies $Q \models \mathsf{G}$. The proof is by structural induction on G.

- if $\mathsf{G} = \mathtt{t}$, then $Q \models \mathsf{G}$.
- if $\mathsf{G} = \mathtt{f}$, then the assumption $P \models \mathsf{G}$ is false and the implication holds.
- if $\mathsf{G} = \mathsf{G}_1 \wedge \mathsf{G}_2$ we take as inductive hypotheses that

$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{G}_1 \Rightarrow S \models \mathsf{G}_1$$
$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{G}_2 \Rightarrow S \models \mathsf{G}_2$$

  We need to prove that $Q \models \mathsf{G}$. Since $P \models \mathsf{G} = \mathsf{G}_1 \wedge \mathsf{G}_2$ we have $P \models \mathsf{G}_1$ and $P \models \mathsf{G}_2$. Since $P \sim_{\mathsf{F}} Q$, by inductive hypotheses we get $Q \models \mathsf{G}_1$ and $Q \models \mathsf{G}_2$. Hence $Q \models \mathsf{G}_1 \wedge \mathsf{G}_2 = \mathsf{G}$.
- if $\mathsf{G} = \mathsf{G}_1 \vee \mathsf{G}_2$ we take as inductive hypotheses that

$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{G}_1 \Rightarrow S \models \mathsf{G}_1$$
$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{G}_2 \Rightarrow S \models \mathsf{G}_2$$

  We need to prove that $Q \models \mathsf{G}$. Since $P \models \mathsf{G} = \mathsf{G}_1 \vee \mathsf{G}_2$ we have $P \models \mathsf{G}_1$ or $P \models \mathsf{G}_2$. If $P \models \mathsf{G}_1$, since $P \sim_{\mathsf{F}} Q$, by inductive hypotheses we get $Q \models \mathsf{G}_1$ and thus $Q \models \mathsf{G}_1 \vee \mathsf{G}_2 = \mathsf{G}$. If $P \models \mathsf{G}_2$, since $P \sim_{\mathsf{F}} Q$, by inductive hypotheses we get $Q \models \mathsf{G}_2$ and thus $Q \models \mathsf{G}_1 \vee \mathsf{G}_2 = \mathsf{G}$.

- if $\mathsf{G} = \langle \chi \rangle \mathsf{H}$ we take as inductive hypothesis that

$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{H} \Rightarrow S \models \mathsf{H}$$

  We need to prove that $Q \models \mathsf{G}$. Since $P \models \langle \chi \rangle \mathsf{H}$ it means that there exists υ, P' such that $P \overset{\upsilon}{\Rightarrow} P'$ with $\upsilon \models \chi$ and $P' \models \mathsf{H}$. Since $P \sim_{\mathsf{F}} Q$, there exists w, Q' such that $Q \overset{w}{\Rightarrow} Q'$ with $w \models \chi$ and $P' \sim_{\mathsf{F}} Q'$. Then, by inductive hypothesis, $Q' \models \mathsf{H}$ and thus $Q \models \langle \chi \rangle \mathsf{H} = \mathsf{G}$.

- if $\mathsf{G} = [\chi]\mathsf{H}$ we take as inductive hypothesis that

$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{H} \Rightarrow S \models \mathsf{H}$$

  We need to prove that $Q \models \mathsf{G}$. If there is no $\upsilon \models \chi$ such that $Q \overset{\upsilon}{\Rightarrow} Q'$ for some Q', then $Q \models [\chi]\mathsf{H} = \mathsf{G}$ trivially. For any υ, Q' such that $Q \overset{\upsilon}{\Rightarrow} Q'$ with $\upsilon \models \chi$, then as $P \sim_{\mathsf{F}} Q$ there must exist w, P' such that $P \overset{w}{\Rightarrow} P'$ with $w \models \chi$ and $P' \sim_{\mathsf{F}} Q'$. Since $P \models \mathsf{G} = [\chi]\mathsf{H}$ then it must be $P' \models \mathsf{H}$. Since $P' \sim_{\mathsf{F}} Q'$, by inductive hypothesis $Q' \models \mathsf{H}$. Hence $Q \models [\chi]\mathsf{H} = \mathsf{G}$.

$\equiv_{\mathcal{L}_\mathsf{F}} \subseteq \sim_{\mathsf{F}}$: We prove that $\equiv_{\mathcal{L}_\mathsf{F}}$ is a bio-simulation and thus included in $\sim_{\mathsf{F}}$. Take two generic processes $P \equiv_{\mathcal{L}_\mathsf{F}} Q$ and suppose $P \overset{\upsilon}{\Rightarrow} P'$ for some υ, P'.

- If $\upsilon \models \mathsf{F}$ we want to prove that there exists some w, Q' such that $Q \overset{w}{\Rightarrow} Q'$, with $w \models \mathsf{F}$ and $P' \equiv_{\mathcal{L}_\mathsf{F}} Q'$.

  Towards a contradiction, assume that we cannot find such w, Q'.

  If there is no transition $Q \overset{w}{\Rightarrow} Q'$ such that $w \models \mathsf{F}$, then the bioHML formula $\mathsf{G} \triangleq \langle \mathsf{F} \rangle \mathtt{t}$ is such that $P \models \mathsf{G}$ and $Q \not\models \mathsf{G}$, contradicting the assumption $P \equiv_{\mathcal{L}_\mathsf{F}} Q$.

  Otherwise, let $\mathcal{Q} \triangleq \{Q' \mid \exists w.\ Q \overset{w}{\Rightarrow} Q' \ \wedge\ w \models \mathsf{F}\}$ be the (non-empty) set of processes reachable from Q via a transition with a (complete) label that satisfies F. Since our processes are with guarded recursion, the set $\mathcal{Q}$ is finite. Let $\mathcal{Q} = \{Q'_1, ..., Q'_n\}$. By hypothesis all processes in $\mathcal{Q}$ must not be bio-logically equivalent to P', hence for any $i \in [1, n]$ there exists a bioHML formula $\mathsf{G}_i \in \mathcal{L}_\mathsf{F}$ such that $P' \models \mathsf{G}_i$ and $Q'_i \not\models \mathsf{G}_i$ (if it was the opposite, $P' \not\models \mathsf{H}_i$ and $Q'_i \models \mathsf{H}_i$ for some H_i, we can use the converse formula $\mathsf{G}_i \triangleq \overline{\mathsf{H}}_i$). But then the formula $\mathsf{G} \triangleq \langle \mathsf{F} \rangle (\mathsf{G}_1 \wedge \cdots \wedge \mathsf{G}_n)$ is such that $P \models \mathsf{G}$ and $Q \not\models \mathsf{G}$, contradicting the assumption $P \equiv_{\mathcal{L}_\mathsf{F}} Q$.

- If $\upsilon \not\models \mathsf{F}$ then the proof is analogous to the previous case (by exploiting $\neg\mathsf{F}$) and thus omitted.

A process algebraic approach to reaction systems ☆

Linda Brodo[b], Roberto Bruni[a], Moreno Falaschi[c]

[a] Dipartimento di Informatica, Università di Pisa, Italy
[b] Dipartimento di Scienze Economiche e Aziendali, Università di Sassari, Italy
[c] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Univ. di Siena, Italy

Abstract

In the area of Natural Computing, Reaction Systems (RSs) are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. RSs interact with a context, and pose challenges for modularity, compositionality, extendibility and behavioural equivalence. In this paper we define a modular encoding of RSs as processes in the chained Core Network Algebra (cCNA), which is a new variant of the `link`-calculus. The encoding represents the behaviour of each entity separately and preserves faithfully their features, and we prove its correctness and completeness. Our encoding provides a Labelled Transition System (LTS) semantics for RSs. Based on the LTS semantics, we adapt the classical notion of bisimulation to define a novel equivalence, called bio-similarity, for studying properties of RSs. In particular, we define a new assertion language based on regular expressions, which allows us to specify the properties of interest, and use it to extend Hennessy-Milner logic to our setting. We prove that our bio-similarity relation and the logical equivalence, that are defined parametrically on some assertion of interest, coincide. Finally, we claim that our encoding contributes to increase the expressiveness of RSs, by exploiting the interaction among different RSs.

Keywords: process algebras, Reaction Systems, assertion language, HM-logic

1. Introduction

Natural Computing is an active area of research which builds on two main aspects: human designed computing inspired by nature, and computation performed in nature. Reaction Systems (RSs) [1] are a rewriting formalism inspired by the way biochemical reactions take place in living cells. This theory has already shown to be relevant in several different fields, including biology [2, 3, 4, 5]

☆Research partially supported by Università degli Studi di Sassari *Fondi di Ateneo per la ricerca 2019*, by MIUR PRIN Project 201784YSZ5 *ASPRA: Analysis of program analyses*, and by Università di Pisa Project PRA_2018_66 *DECLWARE: Metodologie dichiarative per la progettazione e il deployment di applicazioni.*

Email addresses: `brodo@uniss.it` (Linda Brodo), `bruni@di.unipi.it` (Roberto Bruni), `moreno.falaschi@unisi.it` (Moreno Falaschi)

and molecular chemistry [6]. RSs formalise the mechanisms of biochemical systems, such as *facilitation* and *inhibition*. As a qualitative approximation of the real biochemical reactions, they consider if a necessary reagent is present or not, and likewise they consider if an inhibiting molecule is present or not. The possible reactants and inhibitors are called ‘entities’. RSs model in a direct way the interaction of a living cell with the environment (called ‘context’). However, two RSs are seen as independent models and do not interact.

In this paper, which is an extended version of [7], we present an encoding from RSs to the chained Core Network Algebra (cCNA), a variant of the open multiparty process algebra CNA [8]; the CNA equipped with mobility is referred to as the `link`-calculus [9, 10]. Here mobility is not needed. This formalism allows several processes to synchronise and communicate altogether, at the same time, with a new interaction mechanism based on links and link chains. The initial motivation for introducing an open multiparty mechanism in [9] was to encode Mobile Ambients [11], getting a much stronger operational correspondence than any available in the literature, such as the ones in [12, 13]. Later it was shown that the `link`-calculus allowed one to easily encode calculi for biology equipped with membranes, as in [14].

We illustrate our embedding by means of some examples coming both from the computer science and the biological field. We also show that our embedding preserves the main features of RSs, and prove its correctness and completeness from the operational semantics viewpoint.

Then, we present a methodology for verifying formally properties of Reaction Systems. The classical notion of bisimulation for process algebras allows to consider two processes as equivalent when one process can simulate all the actions executed by the other one and vice versa. In this paper we define a new notion of bisimilarity which takes into account the characteristics of biological systems. We define a new *bio-simulation* relation having in mind the possibility that two interacting systems may be compared w.r.t. a subset of the possible biological actions. This is useful for concentrating on the sub-model that one may need to consider for a specific study or application, without getting lost in the complexity of the full biological system, or network. The notion of bio-simulation relies on a new and simple assertion language, which allows to focus on some properties of the Reaction Systems to be verified. In fact, bio-similarity is parametric on a given assertion of interest. Then we prove that the well-known logical characterisation of bisimilarity in terms of Hennessy-Milner Logic [15] (HML) can be extended to the case of bio-similarity by tailoring HML formulas to the same assertion of interest used in bio-similarity. As HML is a powerful formalism for specifying properties of labeled transition systems, its variant introduced here is a suitable option to specify formulas over complex labels by abstracting from unnecessary details.

Our main contributions are as follows:

- RSs are encoded in a modular way, in the sense that all constituents of a RS, namely entities, reactions and the context, are seen as interacting cCNA processes; in principle, this allows to study the behaviour of any constituent in isolation, contrary to what happens in the basic RS framework, where the evolution is possible only when all constituents are given;

- any context is represented as an ordinary cCNA process, which allows us to specify recursive and non-deterministic contexts in a natural way; when deterministic contexts are considered, as in ordinary RSs, the cCNA computation is also deterministic and matches the evolution of the underlying RS;
- we define an assertion language to specify local properties of RSs and exploit it to define a novel notion of behavioural equivalence, called bio-similarity;
- we show that our assertions can be used to extend the Hennessy-Milner's logic to our encoding, in such a way that logical equivalence of processes coincide with bio-similarity.

Moreover, we sketch how the encoding can be used to enhance the expressivity of RSs along different dimensions:

- we sketch how one can express the behaviour of entity mutation, in such a way that the mutated version of the entity s can take part to only a subset of reactions requiring entity s;
- we sketch how with a little coding effort, our embedding allows two RSs to communicate; i.e. we can model scenarios where a subset of those entities that the context can provide, are instead provided by a second RS.

The main drawback of our proposal, is that the resulting cCNA code is verbose. Nevertheless it is clear that our encoding can be automatised by means of a proper front-end in an implementation of the `link`-calculus. The examples that we propose also show that some optimisations are possible to reduce the coding efforts when suitable assumptions are made about the provision of entities.

Related work. Process calculi have been used successfully to model biological processes, see [16] for a recent survey. We are not aware of any Structural Operational Semantics for RS; it seems not the case that a deterministic behaviour, as the one of the RS, once the initial state is fixed, could be defined by inference rules that classically are applied to define non-deterministic transition systems. The reversible computation paradigm for RS extends the RS framework by allowing backward computations as well as forward computations; for this goal a set of inference rules for rewriting logic has been defined in [17] to exactly keep trace of those elements that dissolve in the next computation step.

The proposed network of RSs [18] allows any RS in the network to receive entities produced from its neighbours (that will represent its context), with the hypothesis that RSs without neighbours will receive no entities from the context. In the idea we sketch, a RS can receive some entities from the context and some others from a second RS; moreover we can represent contexts having a recursive, non-deterministic behaviour. By exploiting recursion, the kind of interactions which can be defined can be complex and expressive. Example 34 and more in general the discussion in Section 7.2 show that the interaction between RSs can help to model new scenarios.

As already mentioned, a preliminary version of this paper appeared in [7]. There are several major differences w.r.t. the conference version [7], which is here extended as follows:

- we present several new examples (those in Sections 5 and 6.3) to illustrate our encoding, and give more detailed explanations about its use;
- we introduce an assertion language to specify the properties of RSs;
- we define the notion of bio-simulation to relate RSs and compare their behaviours;
- we show that our assertion language allows to specify properties extending to our encoding the Hennessy-Milner logic;
- we include here all proofs of main results.

Structure of the paper. Section 2 describes RSs and their semantics (interactive processes). Section 3 briefly describes the cCNA process algebra and its operational semantics. Section 4 defines the embedding of RSs in cCNA processes and shows some simple examples to illustrate it. Section 5 presents a couple of examples with automata theory and biological applications. Section 6 presents a methodology for the formal verification of properties of Reaction Systems that are expressed in a novel assertion language. Section 7 presents some features and advantages of our embedding for the compositionality of RSs. Finally, Section 8 discusses future work, and concludes.

2. An Overview of Reaction Systems

Natural Computing is concerned with human-designed computing inspired by nature as well as with computation taking place in nature. The theory of Reaction Systems [1] was born in the field of Natural Computing to model the behaviour of biochemical reactions taking place in living cells. Despite its initial aim, this formalism has shown to be quite useful not only for modeling biological phenomena, but also for the contributions which is giving to computer science [19], theory of computing, mathematics, biology [2, 3, 4, 5], and molecular chemistry [6]. Here we briefly review the basic notions of RSs, see [1] for more details.

The mechanisms that are at the basis of biochemical reactions and thus regulate the functioning of a living cell, are *facilitation* and *inhibition*. These mechanisms are reflected in the basic definitions of RSs.

Definition 1 (Reaction). Let S be a set of entities. A reaction over S is a triple $a = (R, I, P)$, where R, I, P are finite, non empty subsets of S and $R \cap I = \emptyset$.

The sets R, I, P are also written R_a, I_a, P_a and called the *reactant set* of a, the *inhibitor set* of a, and the *product set* of a, respectively. All reactants are needed for the reaction to take place. Any inhibitor blocks the reaction if it is present. Products are the outcome of the reaction. Also, $R_a \cup I_a$ is the set of the resources of a and $rac(S)$ denotes the set of all reactions in S. Because R and I are non empty, all products are produced from at least one reactant and every reaction can be inhibited in some way. Sometimes artificial inhibitors are used that are never produced by any reaction. For the sake of simplicity, in some examples, we will allow I to be empty.

Definition 2 (Reaction System). A Reaction System (RS) is an ordered pair $\mathcal{A} = (S, A)$ such that S is a finite set of entities, and $A \subseteq rac(S)$ is a set of reactions over S.

The set S is called the *background set* of $\mathcal{A}$; its elements represent molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system. The set A is the set of *reactions* of $\mathcal{A}$. Since S is finite, so is A: we denote by $|A|$ the number of reactions in A.

Definition 3 (Reaction Result). Given a set of entities S, let $W \subseteq S$ be a finite subset of entities.

1. Let $a \in rac(S)$ be a reaction over S. Then a is enabled by the entities in the set W, denoted by $en_a(W)$, if $R_a \subseteq W$ and $I_a \cap W = \emptyset$, i.e. all the reactants of a are in W, while none of the inhibitors of a are in W. The result of a on W, denoted by $res_a(W)$, is defined by: $res_a(W) \triangleq P_a$, if $en_a(W)$, and $res_a(W) \triangleq \emptyset$ otherwise.
2. Let $A \subseteq rac(S)$ be a (finite) set of reactions over S. The result of A on W, denoted by $res_A(W)$, is defined by: $res_A(W) \triangleq \bigcup_{a \in A} res_a(W)$.

The theory of Reaction Systems is based on the following assumptions.

- **No permanency.** An entity of a set W vanishes unless it is sustained by a reaction. This reflects the fact that a living cell would die for lack of energy, without chemical reactions.

- **No counting.** The basic model of RSs is very abstract and qualitative, i.e. the quantity of entities that are present in a cell is not taken into account.

- **Threshold nature of resources.** From the previous item, we assume that either an entity is available and there is enough of it (i.e. there are no conflicts), or it is not available at all.

The dynamic behaviour of a RS is formalized in terms of *interactive processes*.

Definition 4 (Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS and let n be a nonnegative integer; An n-step *interactive process* in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$ of finite sequences s.t. $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$ where $C_i, D_i \subseteq S$ are sets of entities for any $i \in [0, n]$, $D_0 = \emptyset$, and $D_i = res_A(D_{i-1} \cup C_{i-1})$ for any $i \in [1, n]$.

Living cells are seen as open systems that continuously react with the external environment, in discrete steps. The sequence γ is the *context sequence* of π; it can be arbitrarily defined and represents the influence of the environment on the RS. The sequence δ is the *result sequence* of π and it is entirely determined by γ and A. The sequence $\tau = W_0, \ldots, W_n$ with $W_i = C_i \cup D_i$, for any $i \in [0, n]$ is called a *state sequence*. Each state W_i in a state sequence is the union of two sets: the context C_i at step i and the result $D_i = res_A(W_{i-1})$ from the previous step.

Since we will be able to deal with recursively contexts, we extend the notion of an interactive process to deal with infinite sequences.

Definition 5 (Extended Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an n-step interactive process, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$ Then, we let $\pi^\infty = (\gamma^\infty, \delta^\infty)$ be the extended interactive process of π, defined as $\gamma^\infty = \{C'_i\}_{i\in\mathbb{N}}$, $\delta^\infty = \{D'_i\}_{i\in\mathbb{N}}$, where:

$$C'_j = \begin{cases} C_j & \text{if } j \in [0, n] \\ \emptyset & \text{if } j > n \end{cases} \qquad D'_j = \begin{cases} D_0 & \text{if } j = 0 \\ res_A(D'_{j-1} \cup C'_{j-1}) & \text{if } (j \geq 1) \end{cases}$$

Given an extended interactive process $\pi = (\gamma, \delta)$, we denote by π^k the shift of π starting at the k-th state sequence; formally we let $\pi^k = (\gamma^k, \delta^k)$ with $\gamma^k = \{C'_i\}_{i\in\mathbb{N}}$, $\delta^k = \{D'_i\}_{i\in\mathbb{N}}$ with $C'_0 = C_k \cup D_k$, $D'_0 = \emptyset$, and $C'_i = C_{i+k}$, $D'_i = D_{i+k}$ for any $i \geq 1$.

3. Chained CNA (cCNA)

In this section we introduce the syntax and operational semantics of the process algebra cCNA (chained CNA) [7] to be used for encoding RSs. As already explained in the Introduction, cCNA is a variant of CNA [8], the non-mobile fragment of `link`-calculus [9, 10]. In cCNA the action prefixes are link chains and not just links.

Link Chains. Let $\mathcal{C}$ be the set of channels, ranged over by $\mathsf{a}, \mathsf{b}, \ldots$, and let $Act \triangleq \mathcal{C} \cup \{\tau\} \cup \{\square\}$ be the set of actions, ranged over by $\alpha, \beta, \ldots$, where the symbol τ denotes a *silent* action, while the symbol $\square$ denotes a *virtual* (non-specified) action. A *link* is a pair $\ell = {}^{\alpha}\backslash_{\beta}$; it is *solid* if $\alpha, \beta \neq \square$; intuitively, α and β are two interaction points, one for incoming requests and the other for outgoing requests. The link ${}^{\square}\backslash_{\square}$ is called *virtual*. A link is *valid* if it is solid or virtual. We let $\mathcal{L}$ be the set of valid links. A *link chain* is a finite sequence $v = \ell_1 \ldots \ell_n$ of (valid) links $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ such that:

1. for any $i \in [1, n-1]$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. $\exists i \in [1, n].\ \ell_i \neq {}^{\square}\backslash_{\square}$.

Virtual links represent missing elements of a chain. A chain is called *solid* if it does not contain any virtual link. The empty chain is denoted by ϵ. The equivalence $\bowtie$ models expansion/contraction of virtual links to adjust the length of a link chain.

Definition 6 (Equivalence $\bowtie$). We let $\bowtie$ be the least equivalence relation over link chains closed under the axioms (whenever both sides are well defined):

$$\begin{array}{lcl@{\qquad\qquad}rcl} v\,{}^{\square}\backslash_{\square} & \bowtie & v & v_1\,{}^{\square}\backslash^{\square}_{\square}\backslash_{\square} v_2 & \bowtie & v_1\,{}^{\square}\backslash_{\square} v_2 \\ {}^{\square}\backslash_{\square} v & \bowtie & v & v_1\,{}^{\alpha}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash_{\beta} v_2 & \bowtie & v_1\,{}^{\alpha}\backslash^{\square}_{\mathsf{a}}\backslash^{\mathsf{a}}_{\square}\backslash_{\beta} v_2 \end{array}$$

Two link chains of equal length can be merged whenever each position occupied by a solid link in one chain is occupied by a virtual link in the other chain and solid links in adjacent positions match. Positions occupied by virtual links in both chains remain virtual. Merging is denoted by $v_1 \bullet v_2$. For example, given $v_1 = {}^{\tau}\backslash^{\square}_{\mathsf{a}}\backslash^{\square}_{\square}\backslash_{\square}$, $v_2 = {}^{\square}\backslash^{\mathsf{a}}_{\square}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$ and $v = {}^{\tau}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$ we have $v_1 \bullet v_2 = v$, whereas $v_1 \bullet v$ is not defined. Notably the merge operation is commutative and associative.

Some names in a link chain can be restricted as non observable and transformed into silent actions τ. This is possible only if they are matched by some adjacent link. Restriction is denoted by $(\nu\,\mathsf{a})v$. For example, given $v = {}^{\tau}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$ as above, we have $(\nu\,\mathsf{a})v = {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$, whereas $(\nu\,\mathsf{b})v$ is not defined.

Syntax. The set of cCNA processes, denoted as $\mathcal{P}$ and ranged over by P, Q, is defined by the following grammar:

$$P, Q \quad ::= \quad \mathbf{0} \;\mid\; v.P \;\mid\; P + Q \;\mid\; P|Q \;\mid\; (\nu\,\mathsf{a})P \;\mid\; A$$

where v is a link chain, and A is a process identifier. The syntax of cCNA extends that of CNA [8] by allowing to use link chains as prefixes instead of links, i.e. we allow to write $v.P$ instead of $\ell.P$. For the rest it features nondeterministic choice $P + Q$ (also called sum), parallel composition $P|Q$, restriction $(\nu\,\mathsf{a})P$, and possibly recursively defined process identifiers A. Here we do not consider name mobility, which is present instead in the `link`-calculus and we omit the relabelling operator of CNA (not needed in the encoding).

As common in process algebras we restrict to consider prefix-guarded sums $v_1.P_1 + v_2.P_2$ and, exploiting associativity, we use the shorthand $\sum_{i \in I} v_i.P_i$ for a finite set of indexes $I = \{i_1, ..., i_k\}$ instead of $v_{i_1}.P_{i_1} + \cdots + v_{i_k}.P_{i_k}$. The inactive process $\mathbf{0}$ is thus just the empty summation.

In a prefix $v.P$ we can always assume that the links at the extremities of v are solid: if v needs to be used in larger chains, the operational semantics will add as many virtual links as needed by exploiting the equivalence $\bowtie$ (see rule

$$\frac{\upsilon \bowtie \upsilon_j \quad j \in I}{\sum_{i\in I} \upsilon_i.P_i \xrightarrow{\upsilon} P_j}\ (Sum) \qquad \frac{P \xrightarrow{\upsilon} P' \qquad (A \triangleq P) \in \Delta}{A \xrightarrow{\upsilon} P'}\ (Ide)$$

$$\frac{P \xrightarrow{\upsilon} P'}{(\nu\,\mathsf{a})P \xrightarrow{(\nu\,\mathsf{a})\upsilon} (\nu\,\mathsf{a})P'}\ (Res) \qquad \frac{P \xrightarrow{\upsilon} P'}{P|Q \xrightarrow{\upsilon} P'|Q}\ (Lpar) \qquad \frac{Q \xrightarrow{\upsilon} Q'}{P|Q \xrightarrow{\upsilon} P|Q'}\ (Rpar)$$

$$\frac{P \xrightarrow{\upsilon'} P' \qquad Q \xrightarrow{\upsilon} Q'}{P|Q \xrightarrow{\upsilon \bullet \upsilon'} P'|Q'}\ (Com)$$

Figure 1: SOS semantics of cCNA processes.

Sum in Fig. 1). For example, the process ${}^{\mathsf{a}}\backslash_{\mathsf{b}}.P$ and ${}^{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}.P$ are completely equivalent.

Regarding process constants, we rely on a given set $\Delta = \{A_i \triangleq P_i\}_{i\in I}$ of (possibly recursive) process definitions.

Semantics. The operational semantics of cCNA is defined in the SOS style by the inference rules in Fig. 1. The rules are reminiscent of those for Milner's CCS and they essentially coincide with those of CNA in [8]. The only difference is due to the presence of prefixes that are link chains. Briefly: rule (*Sum*) selects one alternative and puts as label a possible contraction/expansion of the link chain in the selected prefix; rule (*Ide*) selects one transition of the defining process for a constant; rule (*Res*) restricts some names in the label (it cannot be applied when $(\nu\,\mathsf{a})\upsilon$ is not defined); rules (*Lpar*) and (*Rpar*) account for interleaving in parallel composition; rule (*Com*) synchronises interactions (it cannot be applied when $\upsilon \bullet \upsilon'$ is not defined).

Example 7. As some simple examples, consider the recursive process definitions $H \triangleq {}^{\mathsf{a}}\backslash_{\mathsf{b}}.{}^{\mathsf{a}}\backslash_{\mathsf{c}}.H$ and $K \triangleq {}^{\mathsf{a}}\backslash_{\mathsf{b}}.K + {}^{\mathsf{a}}\backslash_{\mathsf{c}}.K$: the former recursively provides a link from a to b and then, at the next step, from a to c; the latter provides at each step a link from a to b or from a to c, nondeterministically.

Analogously to CNA, the operational semantics of cCNA satisfies the so called Accordion Lemma: whenever $P \xrightarrow{\upsilon} P'$ and $\upsilon' \bowtie \upsilon$ then $P \xrightarrow{\upsilon'} P'$.

3.1. Notation for link chains

Hereafter we make use of some new notations for link chains that will facilitate the presentation of our encoding.

Definition 8 (Replication). Let υ be a valid link chain such that $\upsilon\upsilon$ is also a valid link chain. The n times replication of υ, written υ^n, is defined recursively by letting $\upsilon^0 = \epsilon$ (i.e. the empty chain) and $\upsilon^n = \upsilon\upsilon^{n-1}$.

For example, the expression $({}^{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash_{\square})^3$ denotes the chain ${}^{\mathsf{a}}\backslash^{\square}_{\mathsf{b}}\backslash^{\mathsf{a}}_{\square}\backslash^{\square}_{\mathsf{b}}\backslash^{\mathsf{a}}_{\square}\backslash^{\square}_{\mathsf{b}}\backslash_{\square}$. Instead the expression $({}^{\mathsf{a}}\backslash_{\mathsf{b}})^2$ is ill-defined because a does not match with b.

Then, we introduce the notation for *half links* that will be used in conjunction with the *open block of chain* to form regular link chains.

Definition 9 (Half links). Let a be a channel name, we denote by ${}^{\mathsf{a}}\backslash$ the *half left link*, and by $\backslash_{\mathsf{a}}$ the *half right link*.

In the encoding of RS we will make extensive use of subscripted names: each name a will come in two variants a_i and a_o. This is just a technical issue to prevent accidental matching between links. To see why this is important, compare the chain ${}^{\tau}\backslash^{\square}_{\mathsf{a}}\backslash^{\mathsf{a}}_{\square}\backslash_{\tau}$ with ${}^{\tau}\backslash^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\backslash_{\tau}$: the former is $\bowtie$ equivalent to the solid chain ${}^{\tau}\backslash^{\mathsf{a}}_{\mathsf{a}}\backslash_{\tau}$; the latter cannot become solid unless merged with a chain that links a_i to a_o, like ${}^{\square}\backslash^{\mathsf{a}_i}_{\square}\backslash^{\square}_{\mathsf{a}_o}\backslash_{\square}$. This form of subscripting is exploited in the definition of open blocks.

Definition 10 (Open block). Let σ be a finite sequence of names. We define an *open block* as $\Big(\mathop{\backslash\!\!\backslash}\limits_{\mathsf{a}\,\in\,\sigma} {}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\Big)$, where a_i and a_o are annotated version of the name a (as explained above), by letting

$$\Big(\mathop{\backslash\!\!\backslash}\limits_{\mathsf{a}\,\in\,\sigma} {}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\Big) \triangleq \begin{cases} \epsilon & \text{if } \sigma = \epsilon \text{ is the empty sequence} \\ {}^{\square}_{\mathsf{b}_i}\backslash^{\mathsf{b}_o}_{\square}\Big(\mathop{\backslash\!\!\backslash}\limits_{\mathsf{a}\,\in\,\rho} {}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\Big) & \text{if } \sigma = \mathsf{b}\rho \end{cases}$$

Abusing the notation, we will use the open block notation for sets of names rather than sequences, assuming the names in the set are taken according to some default order (e.g. the lexicographic one).

We then combine half links and open blocks to form valid link chains.

For example, for $X = \{\mathsf{a}, \mathsf{b}\}$ the expression $\Big(\mathop{\backslash\!\!\backslash}\limits_{\mathsf{c}\,\in\,X} {}^{\square}_{\mathsf{c}_i}\backslash^{\mathsf{c}_o}_{\square}\Big)$ denotes the block ${}^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\backslash^{\square}_{\mathsf{b}_i}\backslash^{\mathsf{b}_o}_{\square}$; and the expression ${}^{\mathsf{r}_1}\backslash\Big(\mathop{\backslash\!\!\backslash}\limits_{\mathsf{c}\,\in\,X} {}^{\square}_{\mathsf{c}_i}\backslash^{\mathsf{c}_o}_{\square}\Big)\backslash_{\mathsf{r}_2}$ denotes the chain ${}^{\mathsf{r}_1}\backslash^{\square}_{\mathsf{a}_i}\backslash^{\mathsf{a}_o}_{\square}\backslash^{\square}_{\mathsf{b}_i}\backslash^{\mathsf{b}_o}_{\square}\backslash_{\mathsf{r}_2}$.

4. From Reaction Systems to cCNA

Here we define an encoding of Reaction Systems into cCNA. The idea is to define separated processes for representing the behaviour of each entity, each reaction, and for the provisioning of each entity by the context.

In the following we refer to a given set of entities S and a set of reactions $A \subseteq rac(S)$, i.e. that the Reaction System $\mathcal{A} = (S, A)$ is known.

Processes for entities. Given an entity $s \in S$, we exploit five different pairs of channel names for the interactions over s:

- names s_i, s_o are used to test the presence of s in the system;
- names $\widehat{s}_i$, $\widehat{s}_o$ are used to test the provisioning of s from the context;

- names $\widetilde{s}_i$, $\widetilde{s}_o$ are used to test the production of s by some reaction;
- names $\overline{s}_i$, $\overline{s}_o$ are used to test the absence of s in the system;
- names $\underline{s}_i$, $\underline{s}_o$ are used to test the absence of s from the context.

We let P_s be the process implementing the presence of s in the system, and $\overline{P_s}$ be the one for its absence. They can be seen as instances of the same template, which is given below.

$$
\begin{array}{rcl}
P_s & \triangleq & E(s, \widetilde{s}, \widehat{s}, \underline{s}) \qquad\qquad \overline{P_s} \;\triangleq\; E(\overline{s}, \widetilde{s}, \widehat{s}, \underline{s}) \\
E(s, \widetilde{s}, \widehat{s}, \underline{s}) & \triangleq & \sum_{h,k \geq 0} ({}^{s_i}\backslash^{\Box}_{s_o}\backslash_{\Box})^h \; {}^{\widehat{s}_i}\backslash^{\Box}_{\widehat{s}_o}\backslash_{\Box} ({}^{\widetilde{s}_i}\backslash^{\Box}_{\widetilde{s}_o}\backslash_{\Box})^k . P_s \\
& & + \\
& & \sum_{h \geq 0, k \geq 1} ({}^{s_i}\backslash^{\Box}_{s_o}\backslash_{\Box})^h \; {}^{\underline{s}_i}\backslash^{\Box}_{\underline{s}_o}\backslash_{\Box} ({}^{\widetilde{s}_i}\backslash^{\Box}_{\widetilde{s}_o}\backslash_{\Box})^k . P_s \\
& & + \\
& & \sum_{h \geq 0} ({}^{s_i}\backslash^{\Box}_{s_o}\backslash_{\Box})^h \; {}^{\underline{s}_i}\backslash_{\underline{s}_o} . \overline{P_s}
\end{array}
$$

The first line of $E(s, \widetilde{s}, \widehat{s}, \underline{s})$ accounts for the case where s is tested for presence by h reactions and produced by k reactions, while being provided by the context (${}^{\widehat{s}_i}\backslash_{\widehat{s}_o}$). Thus, s will be present at the next step (the continuation is P_s). Here h and k are not known a priori and therefore any combination is possible. In practice, by knowing the number of reactions that test s, we can bound the maximum values of h and k. The second line accounts for the analogous case where s is not provided by the context (${}^{\underline{s}_i}\backslash_{\underline{s}_o}$). The condition $k \geq 1$ guarantees that s will remain present (the continuation is P_s). The third line accounts for the case where s is tested for presence, but it is neither produced nor provided by the context. Therefore, in the next step s will be absent in the system (the continuation is $\overline{P_s}$). Note that in the case of $\overline{P_s}$ the test for presence of s in the system is just replaced by the test for its absence.

Processes for reactions. Here we focus on the encoding of the set of reactions $A \subseteq rac(S)$. We assume that all the reactions a are numbered and use j as an index for reactions. We introduce two channel names for each reaction aj:

- r_j to mark the occurrence of the reaction;
- p_j to mark the product set of the reaction.

We shall exploit names r_j, p_j to join the chains provided by the application of all the reactions. The process for the jth reaction $aj = (R_j, I_j, P_j)$ must assert either the possibility to apply the reaction or its impossibility. The first case happens when all its reactants are present (the link ${}^{s_i}\backslash_{s_o}$ is requested for any $s \in R_j$) and all its inhibitors are absent (the link ${}^{\overline{e}_i}\backslash_{\overline{e}_o}$ is requested for any $e \in I_j$), then the product set is released (the link ${}^{\widetilde{c}_i}\backslash_{\widetilde{c}_o}$ is requested for any $c \in P_j$). The second case can happen for two reasons: one of the reactants is absent (the link ${}^{\overline{s}_i}\backslash_{\overline{s}_o}$ is requested for some $s \in R_j$) or one of the inhibitors is present (the link ${}^{e_i}\backslash_{e_o}$ is requested for some $e \in I_j$). The process is recursive so that reactions can be applied at any step.

$$
\begin{array}{ll}
P_{aj} \triangleq & \\
{}^{r_j}\backslash \left(\mathop{\Big\|}_{s\,\in\,R_j} {}^{\square}_{s_i}\backslash^{s_o}_{\square} \right) \backslash \left(\mathop{\Big\|}_{e\,\in\,I_j} {}^{\square}_{\overline{e}_i}\backslash^{\overline{e}_o}_{\square} \right) \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash \left(\mathop{\Big\|}_{c\,\in\,P_j} {}^{\square}_{\widetilde{c}_i}\backslash^{\widetilde{c}_o}_{\square} \right) \backslash_{p_{j+1}}.P_{aj} & \{aj \text{ is applicable}\} \\
+ & \\
\sum_{s\in R_j} {}^{r_j}\backslash^{\square}_{\overline{s}_i}\backslash^{\overline{s}_o}_{\square}\backslash^{\square}_{r_{j+1}}\backslash^{p_j}_{\square}\backslash_{p_{j+1}}.P_{aj} & \{aj \text{ is not applicable}\} \\
+ & \\
\sum_{e\in I_j} {}^{r_j}\backslash^{\square}_{e_i}\backslash^{e_o}_{\square}\backslash^{\square}_{r_{i+1}}\backslash^{p_j}_{\square}\backslash_{p_{j+1}}.P_{aj} & \{aj \text{ is not applicable}\}
\end{array}
$$

Channels r_j and r_{j+1} enclose the enabling/disabling condition of reaction aj. Channels p_j and p_{j+1} enclose the links related to the entities produced by aj. Each reaction defines a pattern to be satisfied, i.e. each reaction inserts as many virtual links as the number of reactants, inhibitors, and products, as required by the corresponding reaction.

We will see that all the link chain labels of transitions follow the same schema: first we find all the reactions limited to the reactants and inhibitors (chained using r_j channels), then all the supplies by the contexts (chained using the channel *cxt*, to be introduced next), and finally the products for all the reactions (chained using p_j channels). For notational convenience, we fix that $r_{|A|+1} = cxt$ and $p_{|A|+1} = \tau$. This schema will be later illustrated in detail in Example 15.

Processes for contexts. For marking the part of the chain provided by the context, we exploit the name *cxt*. In RSs, the context sequence γ provides a set of entities C_n at each instant of time n: for each entity $s \in S$, the context must say if the entity is provided or not. Correspondingly, we introduce another process Cxt_n defined as follows:

$$
Cxt_n \quad \triangleq \quad {}^{cxt}\backslash \left(\mathop{\Big\|}_{s\,\in\,C_n} {}^{\square}_{\widehat{s}_i}\backslash^{\widehat{s}_o}_{\square} \right) \backslash \left(\mathop{\Big\|}_{e\,\notin\,C_n} {}^{\square}_{\underline{e}_i}\backslash^{\underline{e}_o}_{\square} \right) \backslash_{p_1}.Cxt_{n+1}
$$

We only consider Cxt_n with $n > 0$, as the entities that are present at step zero are considered to be present in the initial system (if $s \in C_0$ the process P_s will be present initially, otherwise $\overline{P_s}$ will be present).

Encoding. In the following we use the following conventions for denoting different categories of names:

- *decs* $\triangleq \{s, \overline{s}, \widetilde{s}, \widehat{s}, \underline{s} \mid s \in S\}$ is the set of channel names for decorated entities (without subscripts $_i$ and $_o$);
- *ents* $\triangleq \{d_i, d_o \mid d \in decs\}$ is the set of channel names for entities;
- *reacts* $\triangleq \{r_1, ..., r_{|A|+1}\}$ is the set of channel names r_j associated with each reaction aj (we remind that $r_{|A|+1} = cxt$);

- *prods* $\triangleq \{p_1, ..., p_{|A|}\}$ is the set of channel names p_j for product sets associated with each reaction aj (we remind that $p_{|A|+1} = \tau$).

Definition 11 (Encoding). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in\mathbb{N}}$. We define its cCNA encoding $[\![\mathcal{A}, \gamma]\!]$ as follows:

$$[\![\mathcal{A}, \gamma]\!] \triangleq (\nu\, names) \left(I \mid \prod_{a\in A} P_a \mid Cxt_1 \mid \prod_{s\in C_0} P_s \mid \prod_{s\notin C_0} \overline{P_s} \right)$$

where $names = reacts \cup ents \cup prods \cup \{cxt\}$. For technical reasons, we introduce the (trivially recursive) init process $I \triangleq {}^{\tau}\backslash_{r_1}.I$: it is needed to allow the name r_1 to be matched at the start of any chain at any instant of time.

It is important to observe that, for each transition, our cCNA encoding requires all the processes running in parallel to interact in that transition. This is due to the fact that all the channel names r_j, p_j, cxt, including those for decorated names $s_i, s_o, \overline{s}_i, \overline{s}_o, ...$ are restricted.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$. Let $P = [\![\mathcal{A}, \gamma]\!]$ its cCNA encoding. If exists P' such that $P \xrightarrow{(\nu\, names)\upsilon} P'$ is a transition of P, then*

1. *for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in υ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in υ; the channel cxt appears in υ;*
2. *for each reaction $aj \in A$, each virtual link offered by processes P_a and Cxt_1 is overlapped by exactly one solid link offered by processes representing entities.*

The topmost restriction $(\nu\, names)$ appearing in the process $[\![\mathcal{A}, \gamma]\!]$ serves to guarantee that all names appearing in a link of the chain labelling a transition are matched. Since all names appearing in any prefix of $[\![\mathcal{A}, \gamma]\!]$ are restricted, in the transition $[\![\mathcal{A}, \gamma]\!] \xrightarrow{(\nu\, names)\upsilon} P'$ it means that the observation $(\nu\, names)\upsilon$ has the form ${}^{\tau}\backslash_{\tau} \ldots {}^{\tau}\backslash_{\tau}$, i.e., it is silent, and that υ is solid. Later on we will be interested in reasoning about the actual chain υ used in the transition. It has the peculiarity to start and end with silent actions and to include all names in $reacts \cup prods \cup \{cxt\}$. As a matter of notation we call such chain υ *complete*.

Definition 13 (Complete Chain). A chain υ is called *complete* if it is solid (i.e., it contains no virtual link) and has silent actions τ at its extremes. We write $P \stackrel{\upsilon}{\Rightarrow} P'$ to mean that $P \stackrel{\upsilon}{\rightarrow} P'$ with υ complete.

We will then use $(\!|\mathcal{A}, \gamma|\!)$ to refer to the encoding without topmost name restrictions, i.e.,

$$(\!|\mathcal{A}, \gamma|\!) \triangleq I \mid \prod_{a\in A} P_a \mid Cxt_1 \mid \prod_{s\in C_0} P_s \mid \prod_{s\notin C_0} \overline{P_s}.$$

and we will focus on the complete transitions of $\llparenthesis \mathcal{A}, \gamma \rrparenthesis$.

The following Corollary immediately follows from Lemma 12.

Corollary 14. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$. Let $Q = \llparenthesis \mathcal{A}, \gamma \rrparenthesis$ and $P = \llbracket \mathcal{A}, \gamma \rrbracket = (\nu\, names)Q$. Then $P \xrightarrow{(\nu\, names)v} P'$ iff $Q \overset{v}{\Rightarrow} Q'$ and $P' = (\nu\, names)Q'$.*

Example 15. Let $\mathcal{A}$ be a RS whose specification contains two entities, $s1$ and $s2$, and the reactions $r_1 = (s1, \emptyset, s2)$ and $r_2 = (s2, \emptyset, s1)$ that produce $s2$ if $s1$ is present and $s1$ if $s2$ is present. For simplicity, we consider empty sets of inhibitors, which are not allowed by Definition 1, but the reader can assume a void inhibitor is present in both reactions. Then, we assume an extended interactive process $\pi = (\gamma, \delta)$ where the context γ provides $s1$ and $s2$ at every step, but we assume that only $s1$ is initially present. Since the context sequence is constant, we omit the subscript from Cxt. The corresponding cCNA process is $\llbracket \mathcal{A}, \gamma \rrbracket \triangleq (\nu\, names)\llparenthesis \mathcal{A}, \gamma \rrparenthesis$, with

$$\llparenthesis \mathcal{A}, \gamma \rrparenthesis \triangleq I \mid P_{s1} \mid \overline{P}_{s2} \mid P_{r1} \mid P_{r2} \mid Cxt$$

where:

$$\begin{array}{rcl}
P_{r_1} & \triangleq & {}^{r_1}\backslash^{\square}_{s1_i}\backslash^{s1_o}_{\square}\backslash^{\square}_{r_2}\backslash^{p_1}_{\square}\backslash^{\square}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\square}\backslash_{p_2}.P_{r_1} + \dots; \\
P_{r_2} & \triangleq & \dots + {}^{r_2}\backslash^{\square}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_2}_{\square}\backslash_{\tau}.P_{r_2} + \dots; \\
P_{s1} & \triangleq & {}^{s1_i}\backslash^{\square}_{s1_o}\backslash^{\widehat{s1}_i}_{\square}\backslash_{\widehat{s1}_o}.P_{s1} + \dots; \\
\overline{P}_{s2} & \triangleq & {}^{\overline{s2}_i}\backslash^{\square}_{\overline{s2}_o}\backslash^{\widehat{s2}_i}_{\square}\backslash^{\square}_{\widehat{s2}_o}\backslash^{\widetilde{s2}_i}_{\square}\backslash_{\widetilde{s2}_o}.P_{s2} + \dots; \\
Cxt & \triangleq & {}^{cxt}\backslash^{\square}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\square}\backslash^{\square}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\square}\backslash_{p_1}.Cxt
\end{array}$$

For clarity of exposition, we show the code of the processes just in part, to focus on the prefixes that will be involved in the first transition of the system. In Figure 2 we show the structure of a link chain label related to the execution of such a transition. The yellow blocks are referred to init process I, to the processes encoding the reactions, P_{r_1} and P_{r_2}, and to the context Cxt. As the figure puts in evidence, these two kinds of processes determine the structure of the link chain, from end to end, i.e. from the left τ to the right one. We could say that these processes form the *backbone* of the interaction. In contrast, the processes encoding the entities, P_{s1}, $\overline{P}_{s2}$, provide the solid links to be merged with the virtual links of the backbone (i.e. to be plugged in the backbone). In Figure 2, at the bottom of the chain, we have underlined with brackets the origin of the solid links that appear in the chain: the notation $P_1(P_2, P_3)$ means that the segment of the link chain is delimited by the process P_1 and it leaves "holes" where processes between the brackets, in this case P_2 and P_3, insert their links. Formally, we have

$$\llparenthesis \mathcal{A}, \gamma \rrparenthesis \xrightarrow{{}^{\tau}\backslash^{r_1}_{r_1}\backslash^{s1_i}_{s1_i}\backslash^{s1_o}_{s1_o}\backslash^{r_2}_{r_2}\backslash^{\overline{s2}_i}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\overline{s2}_o}\backslash^{cxt}_{cxt}\backslash^{\widehat{s1}_i}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\widehat{s1}_o}\backslash^{\widehat{s2}_i}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\widehat{s2}_o}\backslash^{p_1}_{p_1}\backslash^{\widetilde{s2}_i}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\widetilde{s2}_o}\backslash^{p_2}_{p_2}\backslash_{\tau}} P'$$

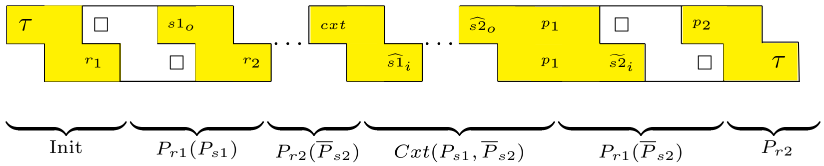

Figure 2: The link chain structure arising from reactions and context processes.

Since the chain in the transition label is complete we can also write

$$(\!|\mathcal{A}, \gamma|\!) \xRightarrow{{}^{\tau}\backslash^{r_1}_{r_1}\backslash^{s1_i}_{s1_i}\backslash^{s1_o}_{s1_o}\backslash^{r_2}_{r_2}\backslash^{\overline{s2}_i}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\overline{s2}_o}\backslash^{cxt}_{cxt}\backslash^{\widehat{s1}_i}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\widehat{s1}_o}\backslash^{\widehat{s2}_i}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\widehat{s2}_o}\backslash^{p_1}_{p_1}\backslash^{\widetilde{s2}_i}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\widetilde{s2}_o}\backslash^{p_2}_{p_2}\backslash_{\tau}} P'$$

Example 15 outlines two different roles of the processes defining the translation of an interactive process: those processes encoding the reactions and the context provide the backbone of each transition, whereas the processes encoding the entities provide the resources needed for the communication to take place.

The flat function. Our transition labels are quite verbose; then, to simplify their processing, we introduce a function that takes a solid link chain and returns a simple string by eliminating all the channel matching pairs and leaving just one placeholder for them. This transformation is harmless, in the sense that it retains all the information in the chain, because it is applied to complete chains only. The function $\mathit{flat}(\cdot)$ is defined inductively as follows:

$$\mathit{flat}(\epsilon) \triangleq \epsilon \qquad \mathit{flat}({}^{\alpha}\backslash_{\beta}) \triangleq \begin{cases} \beta & \text{if } \beta \in \mathit{reacts} \cup \{\mathit{cxt}\} \cup \mathit{prods} \\ d & \text{if } \beta = d_i \text{ with } d \in \mathit{decs} \\ \epsilon & \text{otherwise} \end{cases}$$

$$\mathit{flat}({}^{\alpha}\backslash_{\beta}\upsilon) \quad \triangleq \quad \mathit{flat}({}^{\alpha}\backslash_{\beta})\mathit{flat}(\upsilon)$$

where the usual string concatenation if represented by juxtaposition.

For example, if we consider again the complete label

$$\upsilon = {}^{\tau}\backslash^{r_1}_{r_1}\backslash^{s1_i}_{s1_i}\backslash^{s1_o}_{s1_o}\backslash^{r_2}_{r_2}\backslash^{\overline{s2}_i}_{\overline{s2}_i}\backslash^{\overline{s2}_o}_{\overline{s2}_o}\backslash^{cxt}_{cxt}\backslash^{\widehat{s1}_i}_{\widehat{s1}_i}\backslash^{\widehat{s1}_o}_{\widehat{s1}_o}\backslash^{\widehat{s2}_i}_{\widehat{s2}_i}\backslash^{\widehat{s2}_o}_{\widehat{s2}_o}\backslash^{p_1}_{p_1}\backslash^{\widetilde{s2}_i}_{\widetilde{s2}_i}\backslash^{\widetilde{s2}_o}_{\widetilde{s2}_o}\backslash^{p_2}_{p_2}\backslash_{\tau}$$

from Example 15, we have

$$\mathit{flat}(\upsilon) = r_1 \ s1 \ r_2 \ \overline{s2} \ \mathit{cxt} \ \widehat{s1} \ \widehat{s2} \ p_1 \ \widetilde{s2} \ p_2.$$

It is then immediate to define the function *unflat* to rebuild the complete label from the compact string (here we exploit again the half link and block notation):

$$\mathit{unflat}(x) \triangleq \begin{cases} {}^{x}_{x} & \text{if } x \in \mathit{reacts} \cup \{\mathit{cxt}\} \cup \mathit{prods} \\ {}^{x_i}_{x_i}\backslash^{x_o}_{x_o} & \text{if } x \in \mathit{decs} \end{cases}$$

$$unflat(x_1 \dots x_n) \triangleq {}^{\tau}\backslash unflat(x_1)\backslash \dots \backslash unflat(x_n)\backslash_{\tau}$$

It is immediate to check that for any complete label υ of our processes we have $\upsilon = unflat(flat(\upsilon))$.

With the next proposition, we analyse the structure of a cCNA process encoding of a reactive process after one transition step. In the following four statements, for brevity, we let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in A, with $\gamma = \{C_i\}_{i\in\mathbb{N}}$ and $\delta = \{D_i\}_{i\in\mathbb{N}}$.

Proposition 16 (Correctness 1). *Let $P = [\![\mathcal{A}, \gamma]\!]$ with*

$$P = (\nu\, names) \left(I \mid \prod_{a\in A} P_a \mid Cxt_1 \mid \prod_{s\in C_0} P_s \mid \prod_{s\notin C_0} \overline{P}_s \right).$$

If there exists P' such that $P \xrightarrow{\upsilon} P'$, it holds that:

1. $\upsilon = {}^{\tau}\backslash_{\tau} \dots {}^{\tau}\backslash_{\tau}$, *and*
2. $P' = (\nu\, names) \left(I \mid \prod_{a\in A} P_a \mid Cxt_2 \mid \prod_{s\in C_1\cup D_1} P_s | \prod_{s\notin C_1\cup D_1} \overline{P}_s \right)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, we have $P' = [\![\mathcal{A}, \gamma^1]\!]$.

Now, we extend the previous result to a series of transitions.

Corollary 17 (Correctness 2). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{{}^{\tau}\backslash_{\tau}\dots{}^{\tau}\backslash_{\tau}}{}^{j} P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = [\![\mathcal{A}, \gamma^j]\!]$.*

With the following propositions, we prove that, given a RS $\mathcal{A} = (S, A)$ and an extended interactive process $\pi = (\gamma, \delta)$, then the cCNA process $[\![\mathcal{A}, \gamma]\!]$ can simulate all the evolutions of π.

Proposition 18 (Completeness 1). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $\pi^1 = (\gamma^1, \delta^1)$. Then, $P \xrightarrow{{}^{\tau}\backslash_{\tau}\dots{}^{\tau}\backslash_{\tau}} P' = [\![\mathcal{A}, \gamma^1]\!]$.*

Now, we extend the previous result to a series of transitions.

Corollary 19 (Completeness 2). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $\pi^j = (\gamma^j, \delta^j)$. Then, $P \xrightarrow{{}^{\tau}\backslash_{\tau}\dots{}^{\tau}\backslash_{\tau}}{}^{j} P'' = [\![\mathcal{A}, \gamma^j]\!]$.*

5. Examples

Semantically, the topmost restriction ($\nu\, names$) filters out any interaction with virtual links, and releases a private interaction among all participants where all the channel names in the transition labels are hidden (their occurrences are all replaced by τ). This amounts to require that only complete chains are computed by the interaction. In this section, for simplicity, we shall often omit topmost restrictions ($\nu\, names$) from our encoding, but we shall take into account only transitions whose labels are complete chains, i.e. they do not contain virtual links and start/end with the τ action symbol. This way it is possible to observe all channel names that occur in the interaction.

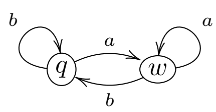

Figure 3: Minimal deterministic labelled transition system.

5.1. Labelled transition system

This example is inspired by the example in [1], where a deterministic transition system is encoded in the Reaction System framework. Here we consider the minimal deterministic transition system in Figure 3.

At the level of RSs, the set of entities to consider is the union of sets of states and of labels of the transition system. Moreover, there is one reaction for each transition: its reactant set consists of the source state and transition label, its inhibitor set includes every other state and label, and its product set is the singleton with the target state. For the transition system in Fig. 3, we take $S = \{q, w, a, b\}$ and the reactions are as follows:

$$\begin{array}{llll} 1 & (\{q,a\},\{w,b\},\{w\}) & 2 & (\{q,b\},\{w,a\},\{q\}) \\ 3 & (\{w,a\},\{q,b\},\{w\}) & 4 & (\{w,b\},\{q,a\},\{q\}) \end{array}$$

Next we show how the above RS is encoded in cCNA.

Encoding of the reactions. The encoding of the reactions is given in a parametric way, with $n \in \{1, 2, 3, 4\}$:

$$\begin{array}{lcl} P_n(q,b,w,a,q) & \triangleq & \upsilon_n(q,b,w,a,q).P_n(q,b,w,a,q) \\ & & + \\ & & \sum_{x\in\{\overline{q},\overline{b},w,a\}} \; \upsilon'_n(x).P_n(q,b,w,a,q) \end{array}$$

where

$$\begin{array}{lcl} \upsilon_n(q,b,w,a,q) & \triangleq & {}^{r_n}\backslash^{\square}_{q_i}\backslash^{q_o}_{\square}\backslash^{\square}_{b_i}\backslash^{b_o}_{\square}\backslash^{\square}_{\overline{w}_i}\backslash^{\overline{w}_o}_{\square}\backslash^{\square}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash^{\square}_{\widetilde{q}_i}\backslash^{\widetilde{q}_o}_{\square}\backslash_{p_{n+1}} \\ \upsilon'_n(x) & \triangleq & {}^{r_n}\backslash^{\square}_{x_i}\backslash^{x_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash_{p_{n+1}} \end{array}$$

Then, we have

$$\begin{array}{lcllcl} P_1 & \triangleq & P_1(q,a,w,b,w) & P_3 & \triangleq & P_3(w,a,q,b,w) \\ P_2 & \triangleq & P_2(q,b,w,a,q) & P_4 & \triangleq & P_4(w,b,q,a,q) \end{array}$$

and we put, as usual, $r_5 = cxt$ and $p_5 = \tau$.

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate the encoding for the entities that are not provided by the context and that can be produced by the reactions, and the ones that can be provided by the context and that are not produced by the reactions.

Here, for the entities q and w that are not provided by the context, we let:

$$\begin{array}{lcllcl} P_q & \triangleq & E(q,\widetilde{q}) & \qquad \overline{P}_q & \triangleq & E(\overline{q},\widetilde{q}) \\ P_w & \triangleq & E(w,\widetilde{w}) & \qquad \overline{P}_w & \triangleq & E(\overline{w},\widetilde{w}) \end{array}$$

where:

$$\begin{array}{rcl} E(q,\widetilde{q}) & \triangleq & \sum_{h=1}^{3} ({}^{q_i}\backslash^{\square}_{q_o}\backslash_{\square})^h \; {}^{\widetilde{q}_i}\backslash_{\widetilde{q}_o}.P_q \\ & & + \\ & & \sum_{h=1}^{3} ({}^{q_i}\backslash^{\square}_{q_o}\backslash_{\square})^h.\overline{P}_q \end{array}$$

In fact the presence/absence of q and w will be exploited by at least one reaction and at most three reactions.

Here, for the entities a and b that can be provided by the context but not produced by reactions, we let:

$$\begin{array}{lcllcl} P_a & \triangleq & E(a,\widehat{a},\underline{a}) & \qquad \overline{P}_a & \triangleq & E(\overline{a},\widehat{a},\underline{a}) \\ P_b & \triangleq & E(b,\widehat{b},\underline{b}) & \qquad \overline{P}_b & \triangleq & E(\overline{b},\widehat{b},\underline{b}) \end{array}$$

where:

$$\begin{array}{rcl} E(a,\widehat{a},\underline{a}) & \triangleq & \sum_{h=1}^{3} ({}^{a_i}\backslash^{\square}_{a_o}\backslash_{\square})^h \; {}^{\widehat{a}_i}\backslash_{\widehat{a}_o}.P_a \\ & & + \\ & & \sum_{h=1}^{3} ({}^{a_i}\backslash^{\square}_{a_o}\backslash_{\square})^h \; {}^{\underline{a}_i}\backslash_{\underline{a}_o}.\overline{P}_a \end{array}$$

Finally, for the context, the encoding follows:

$$Cxt \quad \triangleq \quad {}^{cxt}\backslash^{\square}_{\widehat{a}_i}\backslash^{\widehat{a}_o}_{\square}\backslash^{\square}_{\underline{b}_i}\backslash^{\underline{b}_o}_{\square}\backslash_{p_1}.Cxt \quad + \quad {}^{cxt}\backslash^{\square}_{\widehat{b}_i}\backslash^{\widehat{b}_o}_{\square}\backslash^{\square}_{\underline{a}_i}\backslash^{\underline{a}_o}_{\square}\backslash_{p_1}.Cxt$$

Notice that we exploit here the capabilities of the process algebraic framework to define a nondeterministic, recursive context. We model the context to always offer either a or b, but never both the entities together. The reason is that in the other cases (providing both a and b or neither of them) would lead the system to be stuck because of the simplifications we have adopted in the other processes.

Now, we assume that we have an initial configuration containing the entities q and b:

$$Sys \triangleq \; I \mid P_q \mid \overline{P}_w \mid \overline{P}_a \mid P_b \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt.$$

Then, only the second reaction can be applied, and the transition carries the complete label υ below

$${}^{\tau}\backslash^{r_1}_{r_1}\backslash^{\overline{\mathbf{a}}_{\mathbf{i}}}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\overline{\mathbf{a}}_{\mathbf{o}}}\backslash^{r_2}_{r_2}\backslash^{\mathbf{q}_{\mathbf{i}}}_{q_i}\backslash^{q_o}_{\mathbf{q}_{\mathbf{o}}}\backslash^{\mathbf{b}_{\mathbf{i}}}_{b_i}\backslash^{b_o}_{\mathbf{b}_{\mathbf{o}}}\backslash^{\overline{\mathbf{w}}_{\mathbf{i}}}_{\overline{w}_i}\backslash^{\overline{w}_o}_{\overline{\mathbf{w}}_{\mathbf{o}}}\backslash^{\overline{\mathbf{a}}_{\mathbf{i}}}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\overline{\mathbf{a}}_{\mathbf{o}}}\backslash^{r_3}_{r_3}\backslash^{\overline{\mathbf{a}}_{\mathbf{i}}}_{\overline{a}_i}\backslash^{\overline{a}_o}_{\overline{\mathbf{a}}_{\mathbf{o}}}\backslash^{r_4}_{r_4}\backslash^{\overline{\mathbf{w}}_{\mathbf{i}}}_{\overline{w}_i}\backslash^{\overline{w}_o}_{\overline{\mathbf{w}}_{\mathbf{o}}}\backslash^{cxt}_{cxt}\backslash^{\widehat{\mathbf{a}}_{\mathbf{i}}}_{\widehat{a}_i}\backslash^{\widehat{a}_o}_{\widehat{\mathbf{a}}_{\mathbf{o}}}\backslash^{\underline{\mathbf{b}}_{\mathbf{i}}}_{\underline{b}_i}\backslash^{\underline{b}_o}_{\underline{\mathbf{b}}_{\mathbf{o}}}\backslash^{p_1}_{p_1}\backslash^{p_2}_{p_2}\backslash^{\widetilde{\mathbf{q}}_{\mathbf{i}}}_{\widetilde{q}_i}\backslash^{\widetilde{q}_o}_{\widetilde{\mathbf{q}}_{\mathbf{o}}}\backslash^{p_3}_{p_3}\backslash^{p_4}_{p_4}\backslash_{\tau}$$

The parts in bold are provided by the entity processes, the other parts are provided by the processes encoding the reactions and by the process encoding the context (starting at cxt and ending at p_1). In the label we can read that reactions 1 and 4 have been not executed because the entity a is absent, the reaction 3 has been not applied because the entity w is absent, then only reaction 2 has been applied, and it has produced the entity q. Also, the context provides entity a, that will be available in the next state, and not the entity b. Now, to let the label more readable, we show the result of the application of the function $flat(\cdot)$ to it:

$$r_1 \; \overline{a} \; r_2 \; q \; b \; \overline{w} \; \overline{a} \; r_3 \; \overline{a} \; r_4 \; \overline{w} \; cxt \; \widehat{a} \; \underline{b} \; p_1 \; p_2 \; \widetilde{q} \; p_3 \; p_4.$$

5.2. A biological toy example of gene expression

We consider a biological toy example in the style of gene's alternative splicing [20]. Alternative splicing is a regulated process during gene expression that results in a single gene coding for multiple proteins. In practice, particular exons of a gene may be included within or excluded from the final processed messenger RNA (mRNA) produced from that gene. In our example, a gene a codes for a protein T when molecules G is present and C is absent, and in the opposite situation a codes for protein T'. This behavior is encoded in reactions 1 and 2. Then, reaction 3 codes for the production of C when proteins T and F are present, and T' absent; reaction 4 codes for the production of G when proteins T' is present and F is absent.

Encoding of the reactions. The encoding of reactions is given in a parametric way:

$$\begin{array}{rcl} P_n(a,G,C,T) & \triangleq & \pi_n(a,G,C,T).P_n(a,G,C,T) \\ & & + \\ & & \sum_{x\in\{a,G,C\}} \pi'_n(x).P_n(a,G,C,T) \end{array}$$

where

$$\begin{array}{rcl} \pi_n(a,G,C,T) & \triangleq & {}^{r_n}\backslash^{\square}_{a_i}\backslash^{a_o}_{\square}\backslash^{\square}_{G_i}\backslash^{G_o}_{\square}\backslash^{\square}_{\overline{C}_i}\backslash^{\overline{C}_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash^{\square}_{\widetilde{T}_i}\backslash^{\widetilde{T}_o}_{\square}\backslash_{p_{n+1}} \\ \pi'_n(x) & \triangleq & {}^{r_n}\backslash^{\square}_{x_i}\backslash^{x_o}_{\square}\backslash^{\square}_{r_{n+1}}\backslash^{p_n}_{\square}\backslash_{p_{n+1}} \end{array}$$

Then we have

$$P_1 \;\triangleq\; P_1(a,G,C,T) \qquad P_2 \;\triangleq\; P_2(a,C,G,T') \qquad P_3 \;\triangleq\; P_3(F,T,T',C)$$

$$\begin{array}{rcl} P_4 & \triangleq & {}^{r_4}\backslash^{\square}_{T'_i}\backslash^{T'_o}_{\square}\backslash^{\square}_{\overline{F}_i}\backslash^{\overline{F}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_4}_{\square}\backslash^{\square}_{\widetilde{G}_i}\backslash^{\widetilde{G}_o}_{\square}\backslash_{\tau}.P_4 \\ & & + \\ & & {}^{r_4}\backslash^{\square}_{\overline{T'}_i}\backslash^{\overline{T'}_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_4}_{\square}\backslash_{\tau}.P_4 \;+\; {}^{r_4}\backslash^{\square}_{F_i}\backslash^{F_o}_{\square}\backslash^{\square}_{cxt}\backslash^{p_4}_{\square}\backslash_{\tau}.P_4 \end{array}$$

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate three types of encodings: (1) for the entities that are not provided by the context and can be produced by the reactions; (2) for the entities that can be provided by the context and can be produced by the reactions; (3) for the entities that are only provided by the context.

Here, the entities T and T' that can be produced by the reactions and that are not provided by the context:

$$P(T,\widetilde{T}) \;\triangleq\; \sum_{h=0}^{1}({}^{T_i}\backslash^{\square}_{T_o}\backslash_{\square})^h\;{}^{\widetilde{T}_i}\backslash_{\widetilde{T}_o}.P(T,\widetilde{T}) \quad + \quad {}^{T_i}\backslash_{T_o}.P(\overline{T},\widetilde{T})$$

Then, we have

$$P_T \triangleq P(T,\widetilde{T}) \qquad \overline{P}_T \triangleq P(\overline{T},\widetilde{T}) \qquad P_{T'} \triangleq P(T',\widetilde{T'}) \qquad \overline{P}_{T'} \triangleq P(\overline{T'},\widetilde{T'})$$

The entities that can be produced by the reactions and that can be provided by the context are as follows:

$$
\begin{array}{rcl}
P(C,\widehat{C},\underline{C},\widetilde{C}) & \triangleq & \sum_{h=0}^{1}({}^{C_i}\backslash^{\square}_{C_o}\backslash_{\square})^h \; {}^{\widehat{C}_i}\backslash^{\square}_{\widehat{C}_o}\backslash_{\square}({}^{\widetilde{C}_i}\backslash^{\square}_{\widetilde{C}_o}\backslash_{\square})^h.P(C,\widehat{C},\underline{C},\widetilde{C}) \\
 & & + \\
 & & \sum_{h=0}^{1}({}^{C_i}\backslash^{\square}_{C_o}\backslash_{\square})^h \; {}^{\underline{C}_i}\backslash_{\underline{C}_o}.P(\overline{C},\widehat{C},\underline{C},\widetilde{C}) \\
 & & + \\
 & & \sum_{h=0}^{1}({}^{C_i}\backslash^{\square}_{C_o}\backslash_{\square})^h \; {}^{\underline{C}_i}\backslash_{\underline{C}_o}{}^{\square}\backslash^{\widehat{C}_i}_{\square}\backslash_{\widehat{C}_o}.P(C,\widehat{C},\underline{C},\widetilde{C})
\end{array}
$$

Then, we have

$$
\begin{array}{rclcrcl}
P_C & \triangleq & P(C,\widehat{C},\underline{C},\widetilde{C}) & \qquad & P_G & \triangleq & P(G,\widehat{G},\underline{G},\widetilde{G}) \\
\overline{P}_C & \triangleq & P(\overline{C},\widehat{C},\underline{C},\widetilde{C}) & \qquad & \overline{P}_G & \triangleq & P(\overline{G},\widehat{G},\underline{G},\widetilde{G})
\end{array}
$$

The encoding of the entity F that can only be provided by the context follows:

$$
\begin{array}{rcl}
P_F & \triangleq & \sum_{h=0}^{1}({}^{F_i}\backslash^{\square}_{F_o}\backslash_{\square})^h \; {}^{\widehat{F}_i}\backslash_{\widehat{F}_o}.P_F \quad + \quad \sum_{h=0}^{1}({}^{F_i}\backslash^{\square}_{F_o}\backslash_{\square})^h \; {}^{\underline{F}_i}\backslash_{\underline{F}_o}.\overline{P}_F \\
\overline{P}_F & \triangleq & \sum_{h=0}^{1}({}^{\overline{F}_i}\backslash^{\square}_{\overline{F}_o}\backslash_{\square})^h \; {}^{\widehat{F}_i}\backslash_{\widehat{F}_o}.P_F \quad + \quad \sum_{h=0}^{1}({}^{\overline{F}_i}\backslash^{\square}_{\overline{F}_o}\backslash_{\square})^h \; {}^{\underline{F}_i}\backslash_{\underline{F}_o}.\overline{P}_F
\end{array}
$$

Also in this example we account for a nondeterministic context that can (nondeterministically) provide any combination of the entities C, G F:

$$
Cxt \;\triangleq\; \sum_{\substack{C^\star \in \{C,\overline{C}\} \\ G^\star \in \{G,\overline{G}\} \\ F^\star \in \{F,\overline{F}\}}} {}^{cxt}\backslash^{\square}_{C_i^\star}\backslash^{C_o^\star}_{\square}\backslash^{\square}_{G_i^\star}\backslash^{G_o^\star}_{\square}\backslash^{\square}_{F_i^\star}\backslash^{F_o^\star}_{\square}\backslash_{p_1}.Cxt
$$

Now, to show a possible composition of a transition label, we assume a system where only the entities a, T', and G are present:

$$
Sys \;\triangleq\; I \mid P_a \mid \overline{P}_C \mid P_G \mid \overline{P}_F \mid \overline{P}_T \mid P_{T'} \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt
$$

In the above configuration, reactions 1 and 4 can be applied, and also we assume that the context will provide the entity F, that will be available in the target configuration. Instead of showing the complete transition label, we give its flattened version obtained by applying the function *flat*$(\cdot)$:

$$
r_1 \; a \; G \; \overline{C} \; r_2 \; G \; r_3 \; T' \; r_4 \; T' \; \overline{F} \; cxt \; \underline{C} \; \underline{G} \; \widehat{F} \; p_1 \; \widetilde{T} \; p_2 \; p_3 \; p_4 \; \widetilde{G}.
$$

The original label can then be reconstructed just applying the function *unflat*$(\cdot)$ to the string above.

In [7] we have shown a more complex example, by modeling a RS of a regulatory network for *lac* operon, presented in [3].

6. Bio-simulation

The classical notion of bisimulation for process algebras equates two processes when one process can simulate all the instructions executed by the other one and viceversa. In its weak formulation, internal instructions, i.e. non visible by external observers, are abstracted away. There are many variants of the bisimulation for process algebras, for example the barbed bisimulation [21] only considers the execution of invisible actions, and then equates two processes when they expose the same prefixes; for the mobile ambients [11], a process algebra equipped with a reduction semantics, a notion of behavioural equivalence equates two processes when they expose the same ambients [22].

There are some previous works based on bisimulation applied to models for biological systems. Barbuti et al [23] define a classical setting for bisimulation for two formalisms: the Calculus of Looping Sequences, which is a rewriting system, and the Brane Calculi, which is based on process calculi. Bisimulation is used to verify properties of the regulation of lactose degradation in Escherichia coli and the EGF signalling pathway. These calculi allow the authors to model membranes' behaviour. Cardelli et al [24] present two quantitative behavioral equivalences over species of a chemical reaction network with semantics based on ordinary differential equations. Bisimulation identifies a partition where each equivalence class represents the exact sum of the concentrations of the species belonging to that class. Bisimulation also relates species that have identical solutions at all time points when starting from the same initial conditions. Both the fore mentioned formalisms [23, 24] adopt a classical approach to bisimulation. Albeit the bisimulation is a powerful tool for verifying if the behaviour of two different software programs is indistinguishable, in the case of biological systems the classical bisimulation seems to be inappropriate, as the labels of the transitions systems are too concrete. In fact, in a biological soup, a high number of interactions occur at every computational stage, and generally, biologists are only interested to analyse a small subset of them and to focus just on some entities.

For this reason, we propose an alternative notion of bisimulation, that hereafter we call *bio-simulation*, that allows us to compare two biological systems by restricting the observation to only a limited set of events of interest, which can be chosen according to the property one wants to investigate in an experiment. This allows one to tailor the equivalence to different applications and purposes.

The transition labels of our systems record detailed information about all the reactions that have been applied in one transition, about the elements that acted as reagents, as inhibitors or that have been produced, or that have been provided by the context. All these information are stored in the label because they are necessary to compose a transition in a modular way. Depending on the application, only a suitable abstraction over the label can be of interest.

In a way, we want to query our transition labels to extract only the information we care about. To this goal, we introduce a simple language that allows us to formulate detailed and partial queries about what happened in a single transition.

Example 20. For instance we would like to express properties about each step of the bio-simulation of a system like the ones below:

1. Has the entity s_i been used by reaction r_j as reagent?
2. Has the entity s_i been blocked the application of reaction r_j?
3. Has the entity s_i been produced by reaction r_j?
4. Has the entity s_i been produced by some reaction?
5. Has the entity s_i been provided by the context?
6. Has the reaction r_j not been applied?

As detailed before, in the following we assume that: (i) the context can be non-deterministic, otherwise it makes little sense to rely on bisimulation to observe the branching structure of system dynamics; (ii) we are interested in observing the names of the entities involved in the transitions and also the reactions that have been applied, thus we assume top level restrictions are absent and rely on solid transitions only (with leftmost and rightmost silent actions).

6.1. Assertion language

Next, we introduce an assertion language that operates on strings and that combines regular expression operators with conjunction and disjunction. We let *names* be the set of symbolic names used in our assertion language.

Definition 21 (Assertion Language). Atomic assertions ζ and general assertions F are built from the syntax below:

$$\begin{array}{rcl} \zeta & ::= & \eta \ \mid \ ? \ \mid \ [N] \\ \mathsf{F} & ::= & \epsilon \ \mid \ \zeta \ \mid \ \mathsf{F} :: \mathsf{F} \ \mid \ \mathsf{F}^+ \ \mid \ \mathsf{F}^* \ \mid \ \mathsf{F} \vee \mathsf{F} \ \mid \ \mathsf{F} \wedge \mathsf{F} \end{array}$$

where $\eta \in$ *names* and $N \subseteq$ *names*.

Roughly, an atomic assertion ζ denotes either a string composed by a single name η (one of the symbols in the set *names* for denoting a particular entity, reaction, production or context), or the wildcard ? that stands for any symbol, or the pattern $[N]$ that stands for any string composed by a single symbol in the set N. Clearly ? is just a shorthand for $[\mathit{names}]$. We write $\mathbf{0}$ for $[\emptyset]$ and $[s_1, ..., s_n]$ instead of $[\{s_1, ..., s_n\}]$.

An assertion F is either the empty string ϵ, an atomic assertion ζ, the concatenation of two assertions $\mathsf{F}_1 :: \mathsf{F}_2$, the replication of F for 1 or more times F^+, the replication of F for 0 or more times F^*, the disjunction of two assertions $\mathsf{F}_1 \vee \mathsf{F}_2$ or their conjunction $\mathsf{F}_1 \wedge \mathsf{F}_2$. We denote by $\star$ the assertion $?^*$.

An assertion denotes a set of strings over the alphabet *names* as expected. Below we let $\wp(X)$ denote the powerset of a set X.

Definition 22 (Semantics of Assertions). We define $[\![\mathsf{F}]\!] \subseteq \wp(\mathit{names}^*)$ by induction on the structure of F:

$$\begin{array}{rclcrcl} [\![\epsilon]\!] & \triangleq & \{\epsilon\} & \quad & [\![\mathsf{F}_1 :: \mathsf{F}_2]\!] & \triangleq & \{\omega_1 :: \omega_2 \ \mid \ \omega_1 \in [\![\mathsf{F}_1]\!] \wedge \omega_2 \in [\![\mathsf{F}_2]\!]\} \\ [\![\alpha]\!] & \triangleq & \{\alpha\} & & [\![\mathsf{F}^+]\!] & \triangleq & [\![\mathsf{F}]\!]^+ \\ [\![?]\!] & \triangleq & \mathit{names} & & [\![\mathsf{F}^*]\!] & \triangleq & [\![\mathsf{F}]\!]^* \\ [\![[N]]\!] & \triangleq & N & & [\![\mathsf{F}_1 \vee \mathsf{F}_2]\!] & \triangleq & [\![\mathsf{F}_1]\!] \cup [\![\mathsf{F}_2]\!] \\ & & & & [\![\mathsf{F}_1 \wedge \mathsf{F}_2]\!] & \triangleq & [\![\mathsf{F}_1]\!] \cap [\![\mathsf{F}_2]\!] \end{array}$$

Definition 23 (Satisfaction as Membership). Let υ be a transition label, and F be an assertion. We write $\upsilon \models \mathsf{F}$ (read as the transition label υ satisfies the assertion F) if $\mathit{flat}(\upsilon) \in [\![\mathsf{F}]\!]$, otherwise we write $\upsilon \not\models \mathsf{F}$ (or also $\upsilon \models \neg\mathsf{F}$) and say that F does not hold at υ.

Given two transition labels υ, w we write $\upsilon \equiv_{\mathsf{F}} w$ if $\upsilon \models \mathsf{F} \iff w \models \mathsf{F}$, i.e. if both υ, w satisfy F or they do not.

Example 24. The assertions corresponding to the sample queries listed in Example 20 are as follows:

1. $\star :: r_j :: [s_1, ..., s_n]^* :: s_i :: [s_1, ..., s_n]^* :: [\overline{s}_1, ..., \overline{s}_n]^+ :: \star$
2. $\star :: r_j :: [s_i, \overline{s}_i] :: r_{j+1} :: \star$
3. $\star :: p_j :: [\mathit{ents}]^* :: \widetilde{s}_i :: \star$
4. $\star :: \widetilde{s}_i :: \star$
5. $\star :: \widehat{s}_i :: \star$
6. $\star :: r_j ::? :: r_{j+1} :: \star$

where in 1, 2, 6 we exploit the fact that in a reaction (R, I, P) the sets R of reactants and I of inhibitors are non empty, so that if there is only one symbol between the occurrence of r_j and r_{j+1} it means the reaction r_j has not been applied. Viceversa, if the reaction r_j has been applied the occurrence of r_j must be followed by at least one of the symbols in $\{s_1, ..., s_n\}$ and then by at least one of the symbols in $\{\overline{s}_1, ..., \overline{s}_n\}$.

6.2. Bio-similarity and bio-logical equivalence

The notion of bio-simulation builds on the above language of assertions to parameterize the induced equivalence on the property of interest. Please recall that we have defined the behaviour of the context in a non deterministic way, thus at each step, different possible sets of entities can be provided to the system and different sets of reactions can be enabled/disabled. Bio-simulation can thus be used to compare the behaviour of different systems that share some of the reactions or entities or also to compare the behaviour of the same set of reactions when different contexts are provided.

Definition 25 (Bio-similarity $\sim_{\mathsf{F}}$). Given an assertion F, a *bio-simulation* $\mathbf{R}_{\mathsf{F}}$ that respects F is a binary relation over cCNA processes such that, whenever $P \;\mathbf{R}_{\mathsf{F}}\; Q$ then:

- for any υ, P' such that $P \stackrel{\upsilon}{\Rightarrow} P'$ then there exist w, Q' such that $Q \stackrel{w}{\Rightarrow} Q'$ with $\upsilon \equiv_{\mathsf{F}} w$ and $P' \,\mathbf{R}_{\mathsf{F}}\, Q'$.

- for any w, Q' such that $Q \stackrel{w}{\Rightarrow} Q'$ then there exist υ, P' such that $P \stackrel{\upsilon}{\Rightarrow} P'$ with $\upsilon \equiv_{\mathsf{F}} w$ and $P' \,\mathbf{R}_{\mathsf{F}}\, Q'$.

We let $\sim_{\mathsf{F}}$ denote the largest bio-simulation and we say that P is *bio-similar* to Q, with respect to F, if $P \sim_{\mathsf{F}} Q$.

Remark 26. Please remember that the notation $P \overset{v}{\Rightarrow} P'$ refers to ordinary transitions $P \xrightarrow{v} P'$ where v is a complete chain (solid and with τ actions at the extremes). The double arrow notation should not be confused with the notation for weak transitions commonly found in the literature on process algebras.

Remark 27. An alternative way to look at a bio-simulation that respects F is to define it as an ordinary bisimulation over the transition system labelled over $\{\mathsf{F}, \neg\mathsf{F}\}$ obtained by transforming each transition $P \overset{v}{\Rightarrow} P'$ such that $v \models \mathsf{F}$ into $P \xrightarrow{\mathsf{F}} P'$ and each transition $P \overset{v}{\Rightarrow} P'$ such that $v \not\models \mathsf{F}$ into $P \xrightarrow{\neg\mathsf{F}} P'$.

It can be easily shown that the identity relation is a bio-simulation and that bio-simulations are closed under (relational) inverse, composition and union and that, as a consequence, bio-similarity is an equivalence relation.

Now, we introduce a slightly modified version of the Hennessy Milner Logic (HML) [15], called bioHML; due to the reasons we explained above, we do not want to look at the complete transition labels, thus we rely on our simple assertion language to make it parametric w.r.t the assertion F of interest:

Definition 28 (BioHML). Let F be an assertion, then the set of bioHML formulas G that respects F are built by the following syntax:

$$\begin{array}{rcl} \chi & ::= & \mathsf{F} \mid \neg\mathsf{F} \\ \mathsf{G}, \mathsf{H} & ::= & \mathtt{t} \mid \mathtt{f} \mid \mathsf{G} \wedge \mathsf{G} \mid \mathsf{G} \vee \mathsf{G} \mid \langle\chi\rangle\mathsf{G} \mid [\chi]\mathsf{G} \end{array}$$

Remark 29. An alternative way to look at bioHML formulas is as ordinary HML formulas over the set of labels $\{\mathsf{F}, \neg\mathsf{F}\}$.

As usual, the semantics of a bioHML formula is the set of cCNA processes that satisfy it.

Definition 30 (Semantics of BioHML). We define $[\![\mathsf{G}]\!] \subseteq \mathcal{P}$ by induction on the structure of G:

$$\begin{array}{rclcrcl} [\![\mathtt{t}]\!] & \triangleq & \mathcal{P} & \qquad & [\![\mathsf{G} \wedge \mathsf{H}]\!] & \triangleq & [\![\mathsf{G}]\!] \cap [\![\mathsf{H}]\!] \\ [\![\mathtt{f}]\!] & \triangleq & \emptyset & & [\![\mathsf{G} \vee \mathsf{H}]\!] & \triangleq & [\![\mathsf{G}]\!] \cup [\![\mathsf{H}]\!] \end{array}$$

$$\begin{array}{rcl} [\![\langle\chi\rangle\mathsf{G}]\!] & \triangleq & \{P \in \mathcal{P} : \exists v, P'.\ P \overset{v}{\Rightarrow} P' \text{ with } v \models \chi \text{ and } P' \in [\![\mathsf{G}]\!]\} \\ [\![[\chi]\mathsf{G}]\!] & \triangleq & \{P \in \mathcal{P} : \forall v, P'.\ P \overset{v}{\Rightarrow} P' \text{ implies } v \models \chi \text{ and } P' \in [\![\mathsf{G}]\!]\} \end{array}$$

We write $P \models \mathsf{G}$ (P satisfies G) if and only if $P \in [\![\mathsf{G}]\!]$.

Negation is not included in the syntax, but the converse $\overline{\mathsf{G}}$ of a bioHML formula G can be easily defined inductively in the same way as for HML logic.

Definition 31 (Converse). Given a bioHML formula G we define its converse $\overline{\mathsf{G}}$ as follows:

$$\begin{array}{rclcrclcrcl} \overline{\mathtt{t}} & \triangleq & \mathtt{f} & \qquad & \overline{\mathsf{G} \wedge \mathsf{H}} & \triangleq & \overline{\mathsf{G}} \vee \overline{\mathsf{H}} & \qquad & \overline{\langle\chi\rangle\mathsf{G}} & \triangleq & [\chi]\overline{\mathsf{G}} \\ \overline{\mathtt{f}} & \triangleq & \mathtt{t} & & \overline{\mathsf{G} \vee \mathsf{H}} & \triangleq & \overline{\mathsf{G}} \wedge \overline{\mathsf{H}} & & \overline{[\chi]\mathsf{G}} & \triangleq & \langle\chi\rangle\overline{\mathsf{G}} \end{array}$$

We observe that, as expected, for any bioHML formula G and process P we have $\overline{\overline{\mathsf{G}}} = \mathsf{G}$ and $P \models \overline{\mathsf{G}}$ iff $P \not\models \mathsf{G}$.

Definition 32 (Bio-logical equivalence). We let $\mathcal{L}_{\mathsf{F}}$ be the set of all bioHML formulas that respects F and we say that two processes P, Q are *bio-logically equivalent w.r.t.* F, written $P \equiv_{\mathcal{L}_{\mathsf{F}}} Q$, when P and Q satisfy exactly the same bioHML formulas in $\mathcal{L}_{\mathsf{F}}$, i.e. when for any $\mathsf{G} \in \mathcal{L}_{\mathsf{F}}$ we have $P \models \mathsf{G} \Leftrightarrow Q \models \mathsf{G}$.

Finally, we extend the classical result establishing the correspondence between the logical equivalence induced by HML with bisimilarity for proving that bio-similarity coincides with bio-logical equivalence.

Theorem 33 (Correspondence). $\sim_{\mathsf{F}} = \equiv_{\mathcal{L}_{\mathsf{F}}}$

6.3. Bio-simulation at work

We will show how bio-simulation works. For the sake of space, we consider a very simple example with only two reactions. Reaction P_1 requires G to produce C; reaction P_2 requires C to produce G; both reactions have H as inhibitor. Now, we set two systems defined by the same two reactions, with the two different initial configuration, and with two different context definitions. The two reactions work as follows (where we omit to specify the cases where H is present, as they will never happen):

$$\begin{array}{lcl} P_1 & \triangleq & {}^{\tau}\backslash^{\Box}_{G_i}\backslash^{G_o}_{\Box}\backslash^{\Box}_{\overline{H}_i}\backslash^{\overline{H}_o}_{\Box}\backslash^{\Box}_{r_2}\backslash^{p_1}_{\Box}\backslash^{\Box}_{\widetilde{C}_i}\backslash^{\widetilde{C}_o}_{\Box}\backslash_{p_2}.P_1 \;+\; {}^{\tau}\backslash^{\Box}_{\overline{G}_i}\backslash^{\overline{G}_o}_{\Box}\backslash^{\Box}_{r_2}\backslash^{p_1}_{\Box}\backslash_{p_2}.P_1 \\ P_2 & \triangleq & {}^{r_2}\backslash^{\Box}_{C_i}\backslash^{C_o}_{\Box}\backslash^{\Box}_{\overline{H}_i}\backslash^{\overline{H}_o}_{\Box}\backslash^{\Box}_{cxt}\backslash^{p_2}_{\Box}\backslash^{\Box}_{\widetilde{G}_i}\backslash^{\widetilde{G}_o}_{\Box}\backslash_{\tau}.P_2 \;+\; {}^{r_2}\backslash^{\Box}_{\overline{C}_i}\backslash^{\overline{C}_o}_{\Box}\backslash^{\Box}_{cxt}\backslash^{p_2}_{\Box}\backslash_{\tau}.P_2 \end{array}$$

The two contexts follow :

$$Cxt \triangleq {}^{cxt}\backslash^{\Box}_{\widehat{C}_i}\backslash^{\widehat{C}_o}_{\Box}\backslash^{\Box}_{\underline{G}_i}\backslash^{\underline{G}_o}_{\Box}\backslash^{\Box}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\Box}\backslash_{p_1}.Cxt + {}^{cxt}\backslash^{\Box}_{\underline{C}_i}\backslash^{\underline{C}_o}_{\Box}\backslash^{\Box}_{\underline{G}_i}\backslash^{\underline{G}_o}_{\Box}\backslash^{\Box}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\Box}\backslash_{p_1}.Cxt$$

$$Cxt' \triangleq {}^{cxt}\backslash^{\Box}_{\widehat{G}_i}\backslash^{\widehat{G}_o}_{\Box}\backslash^{\Box}_{\underline{C}_i}\backslash^{\underline{C}_o}_{\Box}\backslash^{\Box}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\Box}\backslash_{p_1}.Cxt' + {}^{cxt}\backslash^{\Box}_{\underline{G}_i}\backslash^{\underline{G}_o}_{\Box}\backslash^{\Box}_{\underline{C}_i}\backslash^{\underline{C}_o}_{\Box}\backslash^{\Box}_{\underline{H}_i}\backslash^{\underline{H}_o}_{\Box}\backslash_{p_1}.Cxt'$$

The definition of the processes encoding G and C is similar, and it is given in a parametric way:

$$P(G) \;\triangleq\; {}^{G_i}\backslash_{G_o}.\overline{P}(G) \qquad\qquad \overline{P}(G) \;\triangleq\; {}^{\overline{G}_i}\backslash^{\Box}_{\overline{G}_o}\backslash^{\widetilde{G}_i}_{\Box}\backslash_{\widetilde{G}_o}.P(G)$$

and we have $P_G \triangleq P(G)$, $P_C \triangleq P(C)$, then $\overline{P}_H \triangleq {}^{\overline{H}_i}\backslash_{\overline{H}_o}.\overline{P}_H + {}^{\overline{H}_i}\backslash^{\Box}_{\overline{H}_o}\backslash^{\overline{H}_i}_{\Box}\backslash_{\overline{H}_i}.\overline{P}_H$, as H is neither produced nor provided by the context. Then, the initial configuration of system Sys_1 includes G and not C and the context can only provide C, the initial configuration of system Sys_2 includes C and not G and the context can only provide G:

$$\begin{array}{lcl} Sys_1 & \triangleq & I \mid P_1 \mid P_2 \mid P_G \mid \overline{P}_C \mid \overline{P}_H \mid Cxt \\ Sys_2 & \triangleq & I \mid P_1 \mid P_2 \mid \overline{P}_G \mid P_C \mid \overline{P}_H \mid Cxt' \end{array}$$

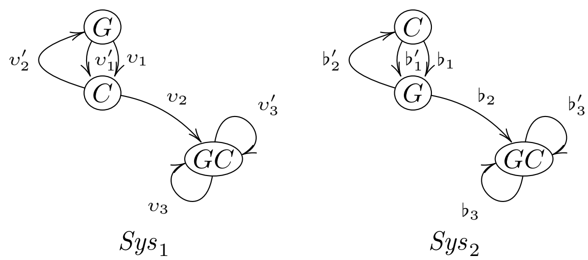

Figure 4: The operational semantics of Sys_1 and Sys_2.

In Figure 4 we show the operational semantics of Sys_1 (on the left) and Sys_2 (on the right), limited to the transitions with complete and solid labels. To improve readability, we named the states of the transition system with the entities that are present in the state. For example, in the leftmost figure, since in Sys_1 only the entity G is available, we name the topmost state G instead of Sys_1. Similarly, for the rightmost figure, where we write, e.g. C instead of Sys_2. As before we show the output of the $\mathit{flat}(\cdot)$ function applied to the transition labels:

$$\begin{array}{lcl}
\mathit{flat}(\upsilon_1) & \triangleq & r_1\ G\ \overline{H}\ r_2\ \overline{C}\ \mathit{cxt}\ \widehat{C}\ \underline{G}\ \underline{H}\ p_1\ \widetilde{C}\ p_2 \\
\mathit{flat}(\upsilon_1') & \triangleq & r_1\ G\ \overline{H}\ r_2\ \overline{C}\ \mathit{cxt}\ \underline{C}\ \underline{G}\ \underline{H}\ p_1\ \widetilde{C}\ p_2
\end{array}$$

$$\begin{array}{lcl}
\mathit{flat}(\upsilon_2) & \triangleq & r_1\ \overline{G}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \widehat{C}\ \underline{G}\ \underline{H}\ p_1\ p_2\ \widetilde{G} \\
\mathit{flat}(\upsilon_2') & \triangleq & r_1\ \overline{G}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \underline{C}\ \underline{G}\ \underline{H}\ p_1\ p_2\ \widetilde{G}
\end{array}$$

$$\begin{array}{lcl}
\mathit{flat}(\upsilon_3) & \triangleq & r_1\ G\ \overline{H}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \widehat{C}\ \underline{G}\ \overline{H}\ p_1\ \widetilde{C}\ p_2\ \widetilde{G} \\
\mathit{flat}(\upsilon_3') & \triangleq & r_1\ G\ \overline{H}\ r_2\ C\ \overline{H}\ \mathit{cxt}\ \underline{C}\ \underline{G}\ \overline{H}\ p_1\ \widetilde{C}\ p_2\ \widetilde{G}
\end{array}$$

The labels $\flat_i\ \flat_i'$, with $i \in \{1,2,3\}$ can be obtained by labels υ_i, υ_i' by substituting G with C and viceversa.

Now, it easy to check that Sys_1 and Sys_2 are bio-similar w.r.t the property F saying that G and C are simultaneously produced, formally: $Sys_1 \sim_{\mathsf{F}} Sys_2$ with $\mathsf{F} = \star :: \widetilde{G} :: \star \wedge \star :: \widetilde{C} :: \star$.

On the contrary, $Sys_1 \not\sim_{\mathsf{F}'} Sys_2$ with $\mathsf{F}' = \star :: \widetilde{C} :: \star$, because it happens that both transition labels ℓ_1 and ℓ_1', in Sys_1, record the production of C, whereas transition labels $\flat_1$ and $\flat_1'$ do not. In fact, the bioHML formula $\mathsf{G} \triangleq \langle \mathsf{F}' \rangle \mathtt{t}$ can be used to distinguish Sys_1 from Sys_2, as $Sys_1 \models \mathsf{G}$ and $Sys_2 \not\models \mathsf{G}$.

7. Towards Enhanced Reaction Systems

Our encoding increases the expressivity of RS concerning: the possibility of alternative behaviour of mutated entities, and the communication between two different RSs. It is important to note that, when the context is deterministic, our encoding guarantees that from each state, in the cCNA transition system, only one state is reachable, as the dynamics is totally deterministic.

7.1. Mutating entities

In RS, when an entity is present, it can potentially be involved in each reactions where it is required. With a few more lines of code, in cCNA it is possible to describe the behaviour of a mutation of an entity, in a way that the mutated version of the entity can take part to only a subset of the reactions requiring the *normal version* of the entity. For example, let us assume that entity $s1$ is consumed by reactions $a1$ and $a2$. Reaction $a1$ produces also $s1$ if $s2$ is present, otherwise $a1$ produces a mutated version of $s1$, say $s1'$. When $s1'$ is produced, reaction $a2$ behaves in the same way as if $s1$ would be absent, whereas $a2$ recognises the presence of $s1'$ and behaves in the same way as if $s1$ would be present. Technically, in both cases it is enough to add one more nondeterministic choice in the code of P_{a1} and P_{a2}.

7.2. Communicating Reaction Systems

We sketch how it is possible to program two RSs encodings, in a way that the entities that usually come from the context of one RS will be provided instead from the other RS.

Example 34. Let $rs1$ and $rs2$ be two RSs, defined, respectively, by the reactions $a_1 = (s, \emptyset, x)$ and $a_2 = (y, \emptyset, s)$. Now, we set our example such that the two contexts, for $rs1$ and $rs2$, do not provide any entities. We also assume that entity s in $rs1$ is provided by $rs2$, as $rs2$ produces a quantity of s that is enough for $rs1$ and $rs2$. For technical reasons, we can not use the same name for s in both the two RSs, then we use the name ss in $rs2$. We need to modify our encoding technique to suit this new setting. As we do not model contexts, we introduce *dummy* channel names dx and dss to model the absence of entities. Also, thanks to the simplicity of the example, we can leave out the use of the p_i channels. This streamlining does not affect the programming technique we propose to make two RSs communicate. First, we translate the reaction in $rs1$, by setting $[\![a_1]\!] \triangleq P_{a_1}$ where:

$$P_{a_1} \quad \triangleq \quad {}^{\tau}\backslash^{\square}_{s_i}\backslash^{s_o}_{\square}\backslash^{\square}_{\widetilde{x}_i}\backslash^{\widetilde{x}_o}_{\square}\backslash_{a_2}.P_{a_1} + {}^{\tau}\backslash^{\square}_{\overline{s}_i}\backslash^{\overline{s}_o}_{\square}\backslash^{\square}_{dx_i}\backslash^{dx_o}_{\square}\backslash_{a_2}.P_{a_1}$$

Please note, that prefixes of process P_{a_1} end with the channel name a_2, as the link chain is now connected with the reaction of $rs2$. The encoding for the entities is given by setting $[\![s]\!] \triangleq P_s$ and $[\![x]\!] \triangleq P_x$, where:

$$\begin{array}{rcl}
P_s & \triangleq & {}^{s_i}\backslash^{\square}_{s_o}\backslash^{\widehat{s}_i}_{\square}\backslash_{\widehat{s}_o}.P_s + {}^{s_i}\backslash_{s_o}.\overline{P_s} \\
\overline{P_s} & \triangleq & {}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash^{\widehat{s}_i}_{\square}\backslash_{\widehat{s}_o}.P_s + {}^{\overline{s}_i}\backslash_{\overline{s}_i}.\overline{P_s} \\
P_x & \triangleq & {}^{\widetilde{x}_i}\backslash_{\widetilde{x}_o}.P_x + {}^{dx_i}\backslash_{dx_o}.\overline{P_x} \\
\overline{P_x} & \triangleq & {}^{\widetilde{x}_i}\backslash_{\widetilde{x}_o}.P_x + {}^{dx_i}\backslash_{dx_o}.\overline{P_x}
\end{array}$$

The encoding for $rs2$ is given by $[\![a_2]\!] \triangleq P_{a_2}$, where:

$$P_{a_2} \quad \triangleq \quad {}^{a_2}\backslash^{\square}_{y_i}\backslash^{y_o}_{\square}\backslash^{\square}_{\widetilde{s}s_i}\backslash^{\widetilde{s}s_o}_{\square}\backslash_{\tau}.P_{a_2} + {}^{a_2}\backslash^{\square}_{\overline{y}_i}\backslash^{\overline{y}_o}_{\square}\backslash^{\square}_{dss_i}\backslash^{dss_o}_{\square}\backslash_{\tau}.P_{a_2}$$

In the encoding of the entities in $rs2$, we introduce the mechanism that allows the entity s (ss in $rs2$) to be provided in $rs1$. Every time ss is produced in $rs2$, a virtual link is created to synchronise with $rs1$ on link ${}^{\widehat{s}_i}\backslash_{\widehat{s}_o}$. To this purpose we define $[\![ss]\!] \triangleq P_{ss}$ and $[\![y]\!] \triangleq P_y$, where:

$$
\begin{array}{lcl}
P_{ss} & \triangleq & {}^{\widetilde{s}s_i}\backslash^{\square}_{\widehat{s}_i}\backslash^{\widehat{s}_o}_{\square}\backslash_{\widetilde{s}s_o}.P_{ss} + {}^{dss_i}\backslash_{dss_o}.\overline{P_{ss}} \\
\overline{P_{ss}} & \triangleq & {}^{\color{red}\widetilde{s}s_i}\backslash^{\color{blue}\square}_{\color{red}\widehat{s}_i}\backslash^{\color{red}\widehat{s}_o}_{\color{blue}\square}\backslash_{\color{red}\widetilde{s}s_o}.P_{ss} + {}^{dss_i}\backslash_{dss_o}.\overline{P_{ss}} \\
P_y & \triangleq & {}^{\color{gray}y_i}\backslash_{\color{gray}y_o}.\overline{P_y} \\
\overline{P_y} & \triangleq & {}^{\overline{y}_i}\backslash_{\overline{y}_o}.\overline{P_y}
\end{array}
$$

We now assume that the initial system is $S \triangleq (\nu\, names)(P_{a_1}|P_{a_2}|P_s|P_y|\overline{P_x}|\overline{P_{ss}})$, i.e. only entities s and y are present. Now, the only possible transition has the following label (that we report without restriction):

$$
{}^{\tau}\backslash^{\color{blue}s_i}_{s_i}\backslash^{s_o}_{\color{blue}s_o}\backslash^{\color{gray}\widetilde{x}_i}_{\widetilde{x}_i}\backslash^{\widetilde{x}_o}_{\color{gray}\widetilde{x}_o}\backslash^{a_2}_{a_2}\backslash^{\color{gray}y_i}_{y_i}\backslash^{y_o}_{\color{gray}y_o}\backslash^{\color{red}\widetilde{s}s_i}_{\widetilde{s}s_i}\backslash^{\color{blue}\widehat{s}_i}_{\color{red}\widehat{s}_i}\backslash^{\color{red}\widehat{s}_o}_{\color{blue}\widehat{s}_o}\backslash^{\widetilde{s}s_o}_{\color{red}\widetilde{s}s_o}\backslash_{\tau},
$$

where the black links belong to the prefixes of P_{a_1}, and P_{a_2}, the blue links belong to P_s, the gray links belong to P_y, and $\overline{P_x}$ and the red links belong to $\overline{P_{ss}}$. After the execution, the entity s is still present in $rs1$ as it has been provided by $rs2$.

As we have briefly sketched, our model of two *communicating Reaction Systems* can enable the study of the behaviour of one RS in relation to another one. Thus, the products of the reactions of one RS can become the input for another one. This could allow for a modular approach to modeling complex systems, by composing different Reaction Systems.

8. Conclusion

In this paper we have introduced cCNA, that generalises CNA by allowing the use of prefixes that are link chains and not just single links. This extension was initially described in the future work section of [8]. Thanks to this enhancement, cCNA allowed us to define a faithful encoding of Reaction Systems in a process algebraic framework. This encoding shows several benefits. First, contexts of RSs can be easily defined recursively and exhibit non deterministic behaviour. Second, the operational semantics is defined in a compositional way by a set of SOS inference rules. Third, we have defined a new assertion language, which allows us to specify the properties to be verified over the labels of the operational semantics. Assertions can be used to tailor the classical notion of bisimilarity and Hennessy-Milner logic to focus on some particular aspects or experiments. We have called *bio-similarity* the induced notion of equivalence.

We are currently investigating how to integrate our methodology with other formal techniques to prove properties of the modeled systems, along the lines in [25, 26, 27, 28]. Moreover, we are considering possible enhancements of RSs based on entity mutation and on the possibility for two RSs to exchange entities.

As future work, we plan to implement a prototype of our embedding, with an automatic translation from RSs to `link`-calculus, so to exploit the implementation of the symbolic semantics of the `link`-calculus [29] that can be found in [30].

Acknowledgments We thank the anonymous reviewers for their detailed and very useful criticisms and recommendations that helped us to improve our paper.

References

[1] R. Brijder, A. Ehrenfeucht, M. Main, G. Rozenberg, A tour of reaction systems, International Journal of Foundations of Computer Science 22 (07) (2011) 1499–1517.

[2] S. Azimi, B. Iancu, I. Petre, Reaction system models for the heat shock response, Fundamenta Informaticae 131 (3-4) (2014) 299–312.

[3] L. Corolli, C. Maj, F. Marinia, D. Besozzi, G. Mauri, An excursion in reaction systems: From computer science to biology, Theoretical Computer Science 454 (2012) 95–108.

[4] S. Azimi, Steady states of constrained reaction systems, Theor. Comput. Sci. 701 (C) (2017) 20–26.

[5] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Investigating dynamic causalities in reaction systems, Theor. Comput. Sci. 623 (2016) 114–145.

[6] F. Okubo, T. Yokomori, The computational capability of chemical reaction automata, Natural Computing 15 (2) (2016) 215–224.

[7] L. Brodo, R. Bruni, M. Falaschi, Enhancing reaction systems: A process algebraic approach, in: M. Alvim, K. Chatzikokolakis, C. Olarte, F. Valencia (Eds.), The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy, Vol. 11760 of Lecture Notes in Computer Science, Springer Berlin, 2019, pp. 68–85.

[8] C. Bodei, L. Brodo, R. Bruni, A formal approach to open multiparty interactions, Theoretical Computer Science 763 (2019) 38–65.

[9] C. Bodei, L. Brodo, R. Bruni, Open multiparty interaction, in: Recent Trends in Algebraic Development Techniques, 21st International Workshop, WADT 2012, Vol. 7841 of Lecture Notes in Computer Science, Springer, 2012, pp. 1–23.

[10] C. Bodei, L. Brodo, R. Bruni, The link-calculus for open multiparty interactions, Information and Computation (2020) 104587.

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

[12] G. Ciobanu, V. A. Zakharov, Encoding mobile ambients into the π-calculus, in: Perspectives of Systems Informatics, Vol. 4378, Springer Berlin Heidelberg, 2007, pp. 148–165.

[13] L. Brodo, On the expressiveness of pi-calculus for encoding mobile ambients, Mathematical Structures in Computer Science 28 (2) (2018) 202–240.

[14] C. Bodei, L. Brodo, R. Bruni, D. Chiarugi, A flat process calculus for nested membrane interactions, Sci. Ann. Comp. Sci. 24 (1) (2014) 91–136.

[15] M. Hennessy, R. Milner, On observing nondeterminism and concurrency, in: J. de Bakker, J. van Leeuwen (Eds.), Automata, Languages and Programming, Vol. 85 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 1980, pp. 299–309.

[16] A. Bernini, L. Brodo, P. Degano, M. Falaschi, D. Hermith, Process calculi for biological processes, Natural Computing 17 (2) (2018) 345–373.

[17] B. Aman, G. Ciobanu, Controlled reversibility in reaction systems, in: M. Gheorghe, G. Rozenberg, A. Salomaa, Z. Claudio (Eds.), Membrane Computing, Springer International Publishing, 2018, pp. 40–53.

[18] P. Bottoni, A. Labella, G. Rozenberg, Networks of reaction systems, International Journal of Foundations of Computer Science 31 (01) (2020) 53–71.

[19] A. Męski, W. Penczek, G. Rozenberg, Model checking temporal properties of reaction systems, Information Sciences 313 (2015) 22–42.

[20] J. D. Watson, T. A. Baker, S. P. Bell, Molecular Biology of the Gene, Pearson Education, USA, 2013.

[21] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 685–695.

[22] A. Gordon, L. Cardelli, Equational properties of mobile ambients, Mathematical Structures in Computer Science 13 (3) (2003) 371–408.

[23] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Bisimulations in calculi modelling membranes, Formal Aspects of Computing 20 (4) (2008) 351–377.

[24] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Forward and backward bisimulations for chemical reaction networks, in: 26th International Conference on Concurrency Theory, CONCUR 2015, Vol. 42, Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2015, pp. 226–239.

[25] D. Chiarugi, M. Falaschi, D. Hermith, C. Olarte, L. Torella, Modelling non-markovian dynamics in biochemical reactions, BMC Systems Biology 9 (S-3) (2015) S8.

[26] C. Olarte, D. Chiarugi, M. Falaschi, D. Hermith, A proof theoretic view of spatial and temporal dependencies in biochemical systems, Theor. Comput. Sci. 641 (2016) 25–42.

[27] C. Bodei, L. Brodo, R. Gori, F. Levi, A. Bernini, D. Hermith, A static analysis for Brane Calculi providing global occurrence counting information, Theoretical Computer Science 696 (2017) 11–51.

[28] C. Bodei, L. Brodo, R. Focardi, Static evidences for attack reconstruction, in: Proc. of Programming Languages with Applications to Biology and Security, Vol. 9465 of Lecture Notes in Computer Science, Springer, 2015, pp. 162–182.

[29] L. Brodo, C. Olarte, Symbolic semantics for multiparty interactions in the link-calculus, in: Proc. of SOFSEM'17, Vol. 10139 of Lecture Notes in Computer Science, Springer, 2017, pp. 62–75.

[30] C. Olarte, SiLVer: Symbolic links verifier, `http://subsell.logic.at/links/links-web/index.html` (Dec. 2018).

Appendix A. Omitted Proofs

In this section we report the proofs for the results in Section 4 and in Section 6.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in $\mathcal{A}$. Let $P = [\![\mathcal{A}, \gamma]\!]$ its cCNA encoding. If exists P' such that $P \xrightarrow{(\nu\, names)\upsilon} P'$ is a transition of P, then*

1. *for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in υ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in υ; the channel cxt appears in υ;*
2. *for each reaction $aj \in A$, each virtual link offered by processes P_a and Cxt_1 is overlapped by exactly one solid link offered by processes representing entities.*

Proof. We prove the two items separately:

1. by Definition 11, all the names that appear in the prefixes of any subprocess are in the set *names* and thus restricted. They include all the reaction names r_j, all the production names p_j, all the entity names s_i, s_o, in all their decorated versions, and the name cxt. Therefore the chain υ must start and end with a τ action and cannot contain virtual links. The only prefix that starts with τ is the one of the recursive init process I (prefix ${}^{\tau}\backslash_{r_1}$) and the only prefixes that end with τ are those associated to reaction a_u (as we have assumed that $p_{u+1} = \tau$, where u is the number of reactions). Then each prefix that starts with r_j involves p_j and r_{j+1}. Thus all the prefixes associated with reactions must be concatenated and also the prefix associated with the context (remember that $r_{u+1} = cxt$), forming the backbone of the label. Since the context processes is involved, then all entities processes are also involved. Then, all the processes I, P_a, P_s (or $\overline{P_s}$), and Cxt_1 must participate to each transition.
2. for each reaction $aj \in A$, the cCNA code of P_{aj} leaves one virtual link between two solid links of the types ${}^{r_j}\backslash_{\dots}\backslash^{\square}_{s_i}\backslash^{s_o}_{\square}\backslash_{\dots}\backslash_{r_{j+1}} \dots {}^{p_j}\backslash_{\dots}\backslash_{p_{j+1}}$. Then, it derives that the process P_s, encoding the behaviour of entity s, can participate by filling the virtual link in the above transition by only offering one solid link of the form ${}^{s_i}\backslash_{s_o}$. In fact, there is no other way to generate a solid chain from s_i to s_o. The same reasoning holds for the processes Cxt_1 and for all the decorated versions of s_i, s_o.

Proposition 16 (Correctness 1). *Let $P = [\![\mathcal{A}, \gamma]\!]$ with*

$$P = (\nu\, names) \left(I \mid \prod_{a \in A} P_a \mid Cxt_1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P}_s \right).$$

If there exists P' such that $P \xrightarrow{\upsilon} P'$, it holds that:

1. $v = {}^{\tau}\backslash_{\tau} \ldots {}^{\tau}\backslash_{\tau}$, *and*
2. $P' = (\nu\ names)\big(I \mid \prod_{a \in A} P_a \mid Cxt_2 \mid \prod_{s \in C_1 \cup D_1} P_s | \prod_{s \notin C_1 \cup D_1} \overline{P}_s\big)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, *we have* $P' = [\![\mathcal{A}, \gamma^1]\!]$.

Proof. First, we note that all the channels in the system are restricted, see Definition 11, then it holds that the transition labels are of the form $v = {}^{\tau}\backslash_{\tau} \ldots {}^{\tau}\backslash_{\tau}$. Now, by Definition 11 and by Lemma 12.1, all the channels r_j, p_j, with $j \in [1, \ldots, u]$, and cxt, and all the annotated versions of s_i, s_o are restricted. Also, processes Cxt_1 always requires the interaction with P_s on either on channels $\widehat{s}_i$, $\widehat{s}_o$ or on channels $\underline{s}_i$, $\underline{s}_o$. It derives that all the processes: P_a (coding the behaviour of reaction $a \in A$), P_s (coding the behaviour of entity $s \in S$), and Cxt_1 (coding the behaviour of the context regarding all the entities) have been involved in the transition.

For any process P_{aj} encoding a reaction aj we have the following cases:

(a) if aj is applicable and it produces the entity s, the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash^{\square}_{\widetilde{s}_i} \backslash^{\widetilde{s}_o}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}}.P_{aj};$$

(b) if aj is applicable and it consumes the entity s, the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{s_i} \backslash^{s_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}}.P_{aj};$$

(c) if aj is applicable and it requires the absence of the entity s, the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{\overline{s}_i} \backslash^{\overline{s}_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}}.P_{aj};$$

(d) if aj is not applicable, the process P_{aj} executes a code capturing either the absence of one of its reactants (case 1), or the presence of one of its inhibitors (case 2):

1. $P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{\overline{s}_i} \backslash^{\overline{s}_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}}.P_{aj};$
2. $P_{aj} \triangleq {}^{r_j}\backslash_{\ldots} \backslash^{\square}_{s_i} \backslash^{s_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{r_{j+1}} \backslash^{p_j}_{\square} \backslash_{\ldots} \backslash_{p_{j+1}}.P_{aj}.$

Now, we consider the structure of the process Cxt_1. By Definition 11, Cxt_1 is the unique process encoding the behaviour of the context regulating the supply of any entity.

(e) The code of the process Cxt_1 that provides s and not e has the following structure:

$$Cxt_1 \triangleq {}^{cxt}\backslash_{\ldots} \backslash^{\square}_{\widehat{s}_i} \backslash^{\widehat{s}_o}_{\square} \backslash_{\ldots} \backslash^{\square}_{\underline{e}_i} \backslash_{\square}{}^{\underline{e}_o} \backslash_{\ldots} \backslash_{p_1}.Cxt_2.$$

The code executed by P_s has the following structure:

(f) $P_s \triangleq \sum_{h,k\geq 0}({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h\, {}^{\widehat{s}_i}\backslash^{\square}_{\widehat{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k\,.P_s$, if $s \in C_{i+1}$;

(g) $P_s \triangleq \sum_{h\geq 0,k\geq 1}({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h\, {}^{\underline{s}_i}\backslash^{\square}_{\underline{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k.P_s$, if $s \notin C_{i+1}$

(h) $P_s \triangleq \sum_{h\geq 0}({}^{s_i}\backslash^{\square}_{s_o}\backslash_{\square})^h\, {}^{\underline{s}_i}\backslash_{\underline{s}_o}.\overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the presence of s plus possibly some reactions not requiring s; and k is the number of reactions producing s.

Similarly, the code executed by $\overline{P}_s$ has the following structure:

(f') $\overline{P}_s \triangleq \sum_{h,k\geq 0}({}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash_{\square})^h\, {}^{\widehat{s}_i}\backslash^{\square}_{\widehat{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k\,.P_s$, if $s \in C_{i+1}$;

(g') $\overline{P}_s \triangleq \sum_{h\geq 0,k\geq 1}({}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash_{\square})^h\, {}^{\underline{s}_i}\backslash^{\square}_{\underline{s}_o}\backslash_{\square}({}^{\widetilde{s}_i}\backslash^{\square}_{\widetilde{s}_o}\backslash_{\square})^k.P_s$, if $s \notin C_{i+1}$

(h') $\overline{P}_s \triangleq \sum_{h\geq 0}({}^{\overline{s}_i}\backslash^{\square}_{\overline{s}_o}\backslash_{\square})^h\, {}^{\underline{s}_i}\backslash_{\underline{s}_o}.\overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the absence of s plus possibly some reactions requiring s; and k is the number of reactions producing s.

It is worth nothing that, depending on the presence (P_s) or the absence ($\overline{P}_s$) of each entity s, for each process P_a (encoding a reaction a) the choice between the execution of the reaction code (points **(a)**, **(b)**, **(c)**) or the code expressing that reaction a is not applicable (point **(d)**) is deterministic. Also, the building of the code of process Cxt (points **(e)**, **(f)**), is univocally determined by the evolution of γ. It derives that the trend followed by the processes P_s (or $\overline{P}_s$) is also deterministic (points **(f)**, **(g)**, **(h)** or **(f')**, **(g')**, **(h')**), leading to $P' = [\![\mathcal{A}, \gamma^1]\!]$.

Corollary 17 (Correctness 2). *Let $P = [\![\mathcal{A}, \gamma]\!]$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}}{}^{j}\ P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = [\![\mathcal{A}, \gamma^j]\!]$.*

Proof. We proceed by induction on the transition number $j \geq 1$.

base case $j = 1$: This case falls into the case of Proposition 16.

inductive case: We assume that the property holds for some generic $j \geq 1$ and we prove that it holds for $j + 1$. Suppose that

$$P \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}}{}^{j+1}\ P''.$$

Then, it must exist P' such that

$$P \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}}{}^{j}\ P' \xrightarrow{{}^{\tau}\backslash_{\tau}\cdots{}^{\tau}\backslash_{\tau}} P''.$$

By inductive hypothesis, $P' = [\![\mathcal{A}, \gamma^j]\!]$. As P' is the encoding of an extended interactive process, by Proposition 16, $P'' = [\![\mathcal{A}, (\gamma^j)^1]\!] = [\![\mathcal{A}, \gamma^{j+1}]\!]$.

Proposition 18 (Completeness 1). *Let* $P = [\![\mathcal{A}, \gamma]\!]$ *and* $\pi^1 = (\gamma^1, \delta^1)$. *Then,* $P \xrightarrow{\tau\backslash_\tau \ldots \tau\backslash_\tau} P' = [\![\mathcal{A}, \gamma^1]\!]$.

Proof. By Proposition 16, if there exists P' such that $P \xrightarrow{\tau\backslash_\tau \ldots \tau\backslash_\tau} P'$, then the structure of P' is deterministically computed.
Now, to prove that always exists P', we observe that even in the case no reaction a is applicable in the interactive process π in A, then process P can always execute a step transition, as its subprocesses P_a can always execute one of the *alternative code for when reaction a is not applicable* (see Definition 11, code for P_a processes).

Corollary 19 (Completeness 2). *Let* $P = [\![\mathcal{A}, \gamma]\!]$ *and* $\pi^j = (\gamma^j, \delta^j)$. *Then,* $P \xrightarrow{\tau\backslash_\tau \ldots \tau\backslash_\tau}{}^{j} P'' = [\![\mathcal{A}, \gamma^j]\!]$.

Proof. The proof proceeds by induction on the number j, and it is similar to the one of Corollary 17.

Theorem 33 (Correspondence). $\sim_\mathsf{F} = \equiv_{\mathcal{L}_\mathsf{F}}$

Proof. The proof is just an adaptation of the classical result. The two implications are proved separately.

$\sim_\mathsf{F} \subseteq \equiv_{\mathcal{L}_\mathsf{F}}$: Given any two processes $P \sim_\mathsf{F} Q$ we need to prove that for any bioHML formula $\mathsf{G} \in \mathcal{L}_\mathsf{F}$ we have $P \models \mathsf{G}$ iff $Q \models \mathsf{G}$. Without loss of generality, we prove that $P \models \mathsf{G}$ implies $Q \models \mathsf{G}$. The proof is by structural induction on G.

- if $\mathsf{G} = \mathtt{t}$, then $Q \models \mathsf{G}$.
- if $\mathsf{G} = \mathtt{f}$, then the assumption $P \models \mathsf{G}$ is false and the implication holds.
- if $\mathsf{G} = \mathsf{G}_1 \wedge \mathsf{G}_2$ we take as inductive hypotheses that

$$\forall R, S.\ R \sim_\mathsf{F} S \wedge R \models \mathsf{G}_1 \Rightarrow S \models \mathsf{G}_1$$
$$\forall R, S.\ R \sim_\mathsf{F} S \wedge R \models \mathsf{G}_2 \Rightarrow S \models \mathsf{G}_2$$

  We need to prove that $Q \models \mathsf{G}$. Since $P \models \mathsf{G} = \mathsf{G}_1 \wedge \mathsf{G}_2$ we have $P \models \mathsf{G}_1$ and $P \models \mathsf{G}_2$. Since $P \sim_\mathsf{F} Q$, by inductive hypotheses we get $Q \models \mathsf{G}_1$ and $Q \models \mathsf{G}_2$. Hence $Q \models \mathsf{G}_1 \wedge \mathsf{G}_2 = \mathsf{G}$.
- if $\mathsf{G} = \mathsf{G}_1 \vee \mathsf{G}_2$ we take as inductive hypotheses that

$$\forall R, S.\ R \sim_\mathsf{F} S \wedge R \models \mathsf{G}_1 \Rightarrow S \models \mathsf{G}_1$$
$$\forall R, S.\ R \sim_\mathsf{F} S \wedge R \models \mathsf{G}_2 \Rightarrow S \models \mathsf{G}_2$$

  We need to prove that $Q \models \mathsf{G}$. Since $P \models \mathsf{G} = \mathsf{G}_1 \vee \mathsf{G}_2$ we have $P \models \mathsf{G}_1$ or $P \models \mathsf{G}_2$. If $P \models \mathsf{G}_1$, since $P \sim_\mathsf{F} Q$, by inductive hypotheses we get $Q \models \mathsf{G}_1$ and thus $Q \models \mathsf{G}_1 \vee \mathsf{G}_2 = \mathsf{G}$. If $P \models \mathsf{G}_2$, since $P \sim_\mathsf{F} Q$, by inductive hypotheses we get $Q \models \mathsf{G}_2$ and thus $Q \models \mathsf{G}_1 \vee \mathsf{G}_2 = \mathsf{G}$.

- if $\mathsf{G} = \langle\chi\rangle\mathsf{H}$ we take as inductive hypothesis that

$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{H} \Rightarrow S \models \mathsf{H}$$

  We need to prove that $Q \models \mathsf{G}$. Since $P \models \langle\chi\rangle\mathsf{H}$ it means that there exists υ, P' such that $P \stackrel{\upsilon}{\Rightarrow} P'$ with $\upsilon \models \chi$ and $P' \models \mathsf{H}$. Since $P \sim_{\mathsf{F}} Q$, there exists w, Q' such that $Q \stackrel{w}{\Rightarrow} Q'$ with $w \models \chi$ and $P' \sim_{\mathsf{F}} Q'$. Then, by inductive hypothesis, $Q' \models \mathsf{H}$ and thus $Q \models \langle\chi\rangle\mathsf{H} = \mathsf{G}$.

- if $\mathsf{G} = [\chi]\mathsf{H}$ we take as inductive hypothesis that

$$\forall R, S.\ R \sim_{\mathsf{F}} S \wedge R \models \mathsf{H} \Rightarrow S \models \mathsf{H}$$

  We need to prove that $Q \models \mathsf{G}$. If there is no $\upsilon \models \chi$ such that $Q \stackrel{\upsilon}{\Rightarrow} Q'$ for some Q', then $Q \models [\chi]\mathsf{H} = \mathsf{G}$ trivially. For any υ, Q' such that $Q \stackrel{\upsilon}{\Rightarrow} Q'$ with $\upsilon \models \chi$, then as $P \sim_{\mathsf{F}} Q$ there must exist w, P' such that $P \stackrel{w}{\Rightarrow} P'$ with $w \models \chi$ and $P' \sim_{\mathsf{F}} Q'$. Since $P \models \mathsf{G} = [\chi]\mathsf{H}$ then it must be $P' \models \mathsf{H}$. Since $P' \sim_{\mathsf{F}} Q'$, by inductive hypothesis $Q' \models \mathsf{H}$. Hence $Q \models [\chi]\mathsf{H} = \mathsf{G}$.

$\equiv_{\mathcal{L}_{\mathsf{F}}} \subseteq \sim_{\mathsf{F}}$: We prove that $\equiv_{\mathcal{L}_{\mathsf{F}}}$ is a bio-simulation and thus included in $\sim_{\mathsf{F}}$. Take two generic processes $P \equiv_{\mathcal{L}_{\mathsf{F}}} Q$ and suppose $P \stackrel{\upsilon}{\Rightarrow} P'$ for some υ, P'.

- If $\upsilon \models \mathsf{F}$ we want to prove that there exists some w, Q' such that $Q \stackrel{w}{\Rightarrow} Q'$, with $w \models \mathsf{F}$ and $P' \equiv_{\mathcal{L}_{\mathsf{F}}} Q'$.

  Towards a contradiction, assume that we cannot find such w, Q'.

  If there is no transition $Q \stackrel{w}{\Rightarrow} Q'$ such that $w \models \mathsf{F}$, then the bioHML formula $\mathsf{G} \triangleq \langle\mathsf{F}\rangle\mathtt{t}$ is such that $P \models \mathsf{G}$ and $Q \not\models \mathsf{G}$, contradicting the assumption $P \equiv_{\mathcal{L}_{\mathsf{F}}} Q$.

  Otherwise, let $\mathcal{Q} \triangleq \{Q' \mid \exists w.\ Q \stackrel{w}{\Rightarrow} Q' \wedge w \models \mathsf{F}\}$ be the (non-empty) set of processes reachable from Q via a transition with a (complete) label that satisfies F. Since our processes are with guarded recursion, the set $\mathcal{Q}$ is finite. Let $\mathcal{Q} = \{Q'_1, ..., Q'_n\}$. By hypothesis all processes in $\mathcal{Q}$ must not be bio-logically equivalent to P', hence for any $i \in [1, n]$ there exists a bioHML formula $\mathsf{G}_i \in \mathcal{L}_{\mathsf{F}}$ such that $P' \models \mathsf{G}_i$ and $Q'_i \not\models \mathsf{G}_i$ (if it was the opposite, $P' \not\models \mathsf{H}_i$ and $Q'_i \models \mathsf{H}_i$ for some H_i, we can use the converse formula $\mathsf{G}_i \triangleq \overline{\mathsf{H}}_i$). But then the formula $\mathsf{G} \triangleq \langle\mathsf{F}\rangle(\mathsf{G}_1 \wedge \cdots \wedge \mathsf{G}_n)$ is such that $P \models \mathsf{G}$ and $Q \not\models \mathsf{G}$, contradicting the assumption $P \equiv_{\mathcal{L}_{\mathsf{F}}} Q$.

- If $\upsilon \not\models \mathsf{F}$ then the proof is analogous to the previous case (by exploiting $\neg\mathsf{F}$) and thus omitted.

Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
elsarticle.cls

Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
elsarticle-num-names.bst

Declaration of interests

X The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: