

# Enhancing reaction systems: a process algebraic approach

Linda Brodo<sup>1</sup>, Roberto Bruni<sup>2</sup>, and Moreno Falaschi<sup>3</sup>

<sup>1</sup> Dipartimento di Scienze economiche e aziendali, Università di Sassari, Italy

<sup>2</sup> Dipartimento di Informatica, Università di Pisa, Italy

<sup>3</sup> Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche  
Università di Siena, Italy

**Abstract.** In the area of Natural Computing, reaction systems are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. This model has already been applied and extended successfully to various areas of research. Reaction systems interact with the environment represented by the context, and pose problems of implementation, as it is a new computation model. In this paper we consider the `link`-calculus, which allows to model multiparty interaction in concurrent systems, and show that it allows to embed reaction systems, by representing the behaviour of each entity and preserving faithfully their features. We show the correctness and completeness of our embedding. We illustrate our framework by showing how to embed a *lac* operon regulatory network. Finally, our framework can contribute to increase the expressiveness of reaction systems, by exploiting the interaction among different reaction systems.

**Keywords:** process algebras, reaction systems, multi-party interaction

## 1 Introduction

Natural Computing is an emerging area of research which has two main aspects: human designed computing inspired by nature, and computation performed in nature. Reaction Systems (RSs) [9] 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, such as computer science [15], biology [2,14,1,3], molecular chemistry [16]. Reaction Systems 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 or not present, and likewise they consider if an inhibiting molecule is or not present. 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, we present an encoding from RSs, to the open multiparty process algebra `cCNA`,<sup>4</sup> a variant of the `link`-calculus [5,6] without name mobility.

<sup>4</sup> After ‘chained Core Network Algebra’.

This formalism allows several processes to synchronise and communicate altogether, at the same time, with a new communicating mechanism based on links and link chains. Our initial motivation for introducing this mechanism was to encode Mobile Ambients [12], getting a much stronger operational correspondence than any available in the literature, such as the one in [10]. This allowed us to easily encode calculi for biology equipped with membranes, as in [7]. Process calculi have been used successfully to model biological processes, see [4] for a recent survey. We illustrate our embedding by means of some simple basic examples, and then we consider a more complex example, by modeling a RS representing a regulatory network for *lac* operon, presented in [14]. We also show that our embedding preserves the main features of RSs, and prove its correctness and completeness. Our main contributions are as follows:

- the behaviour of the context, for each single entity, can be specified in a recursive way as an ordinary process;
- we can express the behaviour of entity mutation, in such a way that the mutated entity  $s'$  can take part to only a subset of rules requiring entity  $s$ ;
- with a little coding effort, two RSs can communicate; i.e. a subset of those entities that the context can provide, are then provided by a second RS;
- as our translation results in a cCNA system, from each state only one transition can be generated, thus the cCNA computation is fully deterministic.

The main drawback of our proposal, is that the cCNA translation is verbose. Nevertheless it is clear that our translation can be automatised by means of a proper front-end in an implementation of the `link`-calculus.

As we have remarked, in our translation, Reaction Systems get the ability to interact between them in a synchronized manner. This interaction is not foreseen in the basic RS framework, as it can only happen with the context. By exploiting recursion, the kind of interactions which can be defined can be complex and expressive. Example 2 and more in general the discussion in Section 6 show that the interaction between RSs can help to model new scenarios.

*Structure of the paper.* Section 2 describes RSs and their semantics (interactive processes). Section 3 describes briefly 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 shows a more complex example taken from the literature on RSs and illustrating a *lac* operon. Section 6 presents some features and advantages of our embedding for the compositionality of RSs. Finally, Section 7 discusses future work, and concludes.

## 2 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 [9] 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 [15], theory of computing, mathematics, biology [2,14,1,3], and molecular chemistry [16]. Here we briefly review the basic notions of RSs, see [9] 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 Reaction Systems.

**Definition 1 (Reaction).** A reaction is a triplet  $a = (R, I, P)$ , where  $R, I, P$  are finite, non empty sets and  $R \cap I = \emptyset$ . If  $S$  is a set such that  $R, I, P \subseteq S$ , then  $a$  is a reaction in  $S$ .

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, and  $A \subseteq rac(S)$ .

The set  $S$  is called the *background set* of  $\mathcal{A}$ , its elements are called *entities*, they 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).** Let  $T$  be a finite set.

1. Let  $a$  be a reaction. Then  $a$  is enabled by  $T$ , denoted by  $en_a(T)$ , if  $R_a \subseteq T$  and  $I_a \cap T = \emptyset$ . The result of  $a$  on  $T$ , denoted by  $res_a(T)$ , is defined by:  $res_a(T) = P_a$ , if  $en_a(T)$ , and  $res_a(T) = \emptyset$  otherwise.
2. Let  $A$  be a finite set of reactions. The result of  $A$  on  $T$ , denoted by  $res_A(T)$ , is defined by:  $res_A(T) = \bigcup_{a \in A} res_a(T)$ .

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

- **No permanency.** An entity of a set  $T$  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  $n \geq 0$ .  
 115 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  $C_i, D_i \subseteq S$  for any  $i \in [0, n]$ ,  $D_0 = \emptyset$ ,  
 and  $D_i = \text{res}_{\mathcal{A}}(D_{i-1} \cup C_{i-1})$  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  $\gamma$  is the *context sequence* of  $\pi$   
 and represents the influence of the environment on the Reaction System. The  
 120 sequence  $\delta$  is the *result sequence* of  $\pi$  and it is entirely determined by  $\gamma$  and  $A$ .  
 The sequence  $\tau = W_0, \dots, W_n$  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 of the previous step.

125 For technical reasons, 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, let  $\pi' = (\gamma', \delta')$  be the extended interactive process of  $\pi = (\gamma, \delta)$ ,  
 130 defined as  $\gamma' = \{C'_i\}_{i \in \mathbb{N}}$ ,  $\delta' = \{D'_i\}_{i \in \mathbb{N}}$ , where  $C'_j = C_j$  for  $j \in [0, n]$  and  $C'_j = \emptyset$   
 for  $j > n$ ,  $D'_0 = D_0$  and  $D'_j = \text{res}_{\mathcal{A}}(D'_{j-1} \cup C'_{j-1})$  for  $j \geq 1$ .

### 3 Chained CNA (cCNA)

In this section we introduce the syntax and operational semantics of a variant  
 of the link-calculus [5], the cCNA (chained CNA) where the prefixes are link  
 135 chains.

*Link Chains.* Let  $\mathcal{C}$  be the set of channels, ranged over by  $a, b, \dots$ , and let  $\mathcal{A} =$   
 $\mathcal{C} \cup \{\tau\} \cup \{\square\}$  be the set of actions, ranged over by  $\alpha, \beta, \dots$ , where the symbol  
 $\tau$  denotes a *silent* action, while the symbol  $\square$  denotes a *virtual* (non-specified)  
 action. A *link* is a pair  $\ell = \alpha \setminus \beta$ ; it is *solid* if  $\alpha, \beta \neq \square$ ; the link  $\square \setminus \square$  is called  
 140 *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  $v = \ell_1 \dots \ell_n$  of (valid) links  $\ell_i = \alpha_i \setminus \beta_i$  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.  $\exists i \in [1, n]. \ell_i \neq \square \setminus \square$ .

Virtual links represent missing elements of a chain. The equivalence  $\blacktriangleleft$  mod-  
 145 els expansion/contraction of virtual links to adjust the length of a link chain.

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

$$\begin{array}{lcl} v \square \setminus \square \blacktriangleleft v & v_1 \square \setminus \square \setminus \square \setminus \square v_2 \blacktriangleleft v_1 \square \setminus \square v_2 & \\ \square \setminus \square v \blacktriangleleft v & v_1 \alpha \setminus \alpha \setminus \beta v_2 \blacktriangleleft v_1 \alpha \setminus \alpha \setminus \square \setminus \beta v_2 & \end{array}$$

$$\begin{array}{c}
\frac{v \blacktriangleright v_j}{\sum_{i \in I} v_i.P_i \xrightarrow{v} P_j} \text{ (Sum)} \quad \frac{P \xrightarrow{v} P' \quad (A \triangleq P) \in \Delta}{A \xrightarrow{v} P'} \text{ (Ide)} \\
\\
\frac{P \xrightarrow{v} P'}{(\nu a)P \xrightarrow{(\nu a)v} (\nu a)P'} \text{ (Res)} \quad \frac{P \xrightarrow{v} P'}{P|Q \xrightarrow{v} P'|Q} \text{ (Lpar)} \quad \frac{P \xrightarrow{v'} P' \quad Q \xrightarrow{v} Q'}{P|Q \xrightarrow{v \bullet v'} P'|Q'} \text{ (Com)}
\end{array}$$

Fig. 1: SOS semantics of the cCNA (rules *(Rel)* and *(Rpar)* omitted).

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  
150  $v_1 = a \setminus_b \square \setminus_c \square \setminus \square$  and  $v_2 = \square \setminus_b \square \setminus_c \square \setminus \square$  we have  $v_1 \bullet v_2 = a \setminus_b \square \setminus_c \square \setminus \square$ .

Some names in a link chain can be restricted as non observable and transformed into silent actions  $\tau$ . This is possible only if they are matched by some adjacent link. Restriction is denoted by  $(\nu a)v$ . For example, given  $v = a \setminus_b \square \setminus_c \square \setminus \square$  we have  $(\nu b)v = a \setminus_\tau \square \setminus_c \square \setminus \square$ .

*Syntax.* The cCNA processes are generated by the following grammar:

$$P, Q ::= \sum_{i \in I} v_i.P_i \mid P|Q \mid (\nu a)P \mid P[\phi] \mid A$$

155 where  $v_i$  is a link chain,  $\phi$  is a channel renaming function, and  $A$  is a process identifier for which we assume a definition  $A \triangleq P$  is available in a given set  $\Delta$  of (possibly recursive) process definitions. We let  $\mathbf{0}$ , the inactive process, denote the empty summation.

160 The syntax of cCNA extends that of CNA [6] by allowing to use link chains as prefixes instead of links. For the rest it features nondeterministic choice, parallel composition, restriction, relabelling and possibly recursive definitions. Here we do not consider name mobility, which is present instead in the **link-calculus**.

*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  
165 and they essentially coincide with those of CNA in [6]. 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  
170 when  $(\nu a)v$  is not defined); rules *(Lpar)* and *(Rpar)* account for interleaving in parallel composition; rule *(Com)* synchronises interactions (it cannot be applied when  $v \bullet v'$  is not defined).

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

175 **3.1 Notation for link chains**

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

**Definition 7 (Replication).** Let  $v$  be a link chain. Its  $n$  times replication  $v^n$  is defined recursively by letting  $v^0 = \epsilon$  (i.e. the empty chain) and  $v^n = v^{n-1}v$ , with the hypothesis that all the links in the resulting link chains match.

For example, the expression  $(a \setminus_b \square \setminus \square)^3$  denotes the chain  $a \setminus_b \square \setminus_a \square \setminus_b \square \setminus_a \square \setminus_b \square \setminus \square$ . We introduce the *half link* that will be used in conjunction with the *open block of chain* to form regular link chains.

**Definition 8 (Half links).** Let  $a$  be a channel name, we define the half left link:  $a \setminus$ , and the half right link:  $\setminus a$ .

**Definition 9 (Open block).** Let  $R$  be a set of names. We define an open block as  $(\bigvee_{a \in R} \square \setminus_a \square)$ , where  $a_i$  and  $a_o$  are annotated version of the name  $a$ , as

set	block of chain	result
$R = \emptyset$	$(\bigvee_{a \in R} \square \setminus_a \square)$	$\epsilon$
$R = \{b\}$	$(\bigvee_{a \in R} \square \setminus_a \square)$	$\square \setminus b_o$
$R = \{b\} \cup R'$	$(\bigvee_{a \in R} \square \setminus_a \square)$	$\square \setminus b_o \setminus (\bigvee_{a \in R'} \square \setminus_a \square)$

We then combine half links and open blocks to form regular link chains. For example, for  $R = \{a, b\}$  the expression  $(\bigvee_{c \in R} \square \setminus_c \square)$  denotes the block of chains  $\square \setminus_a \square \setminus_b \square$ ; and the expression  $r_1 \setminus (\bigvee_{c \in R} \square \setminus_c \square) \setminus r_2$  denotes the chain  $r_1 \setminus_a \square \setminus_b \square \setminus r_2$ .

190 **4 From Reaction Systems to cCNA**

Here we present a translation from Reaction Systems to 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.

*Processes for entities.* Given an entity  $s \in S$ , we exploit four different names for the interactions over  $s$ : names  $s_i, s_o$  are used to test the presence of  $s$  in the system; names  $\hat{s}_i, \hat{s}_o$  are used to test the provisioning of  $s$  from the context; names  $\bar{s}_i, \bar{s}_o$  are used to test the production of  $s$  by some reaction; names  $\underline{s}_i, \underline{s}_o$  are used to test the absence of  $s$  from the context; and names  $\underline{s}_i, \underline{s}_o$  are used to test the absence of  $s$  in the system. 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.

$$P_s \triangleq P(s, \bar{s}, \hat{s}, \underline{s}) \quad \overline{P}_s \triangleq P(\bar{s}, \tilde{s}, \hat{s}, \underline{s})$$



*Processes for contexts.* For each entity  $s \in S$ , we introduce another process  $Cxt_s$ , participating in each transition and saying if, the entity  $s$  is provided by the context or not. As done for the reactions, we assume that entities are enumerated and use the names  $cxt_j$  to concatenate the chains formed by the application of all the contexts. For each entity  $s$  with number  $j$ , at step  $n > 0$  there are two possible behaviours:

$$Cxt_s^n \triangleq \begin{cases} cxt_j \backslash \square_{\hat{s}_i} \backslash \square_{\hat{s}_o} \backslash cxt_{j+1} \cdot Cxt_s^{n+1} & \text{if the context provides } s \text{ at the } n\text{-th step} \\ cxt_j \backslash \square_{\underline{s}_i} \backslash \square_{\underline{s}_o} \backslash cxt_{j+1} \cdot Cxt_s^{n+1} & \text{otherwise} \end{cases}$$

$$Cxt_s \triangleq Cxt_s^1$$

We only consider  $Cxt_s^n$  with  $n > 0$ , as the entities  $s$  that are present at step zero are considered to be present in the initial system (process  $P_s$  instead of  $\bar{P}_s$ ).

225 **Definition 10 (Translation).** 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 translation  $\llbracket \mathcal{A}, \gamma \rrbracket$  as follows:

$$\llbracket \mathcal{A}, \gamma \rrbracket = (\nu \text{ reacts}, cxts, ents, prods)(\Pi_{s \in C_0} P_s | \Pi_{s \notin C_0} \bar{P}_s | \Pi_{a \in A} P_a | \Pi_{s \in S} Cxt_s),$$

230 with *reacts* be the set of reaction names  $r_j$ , *cxts* the set of context names  $cxt_j$ , *ents* the set of decorated entity names  $\{s_i, s_o, \hat{s}_i, \hat{s}_o, \tilde{s}_i, \tilde{s}_o, \bar{s}_i, \bar{s}_o, \underline{s}_i, \underline{s}_o | s \in S\}$ , and *prods* be the set of names  $p_j$  associated to each reaction. In the following, we set *names* = *reacts*  $\cup$  *cxts*  $\cup$  *ents*  $\cup$  *prods*. For notational convenience, we fix that  $r_1 = \tau$ ,  $r_{u+1} = cxt_1$  for  $u$  the number of reacts, and  $cxt_{w+1} = p_1$   $p_{u+1} = \tau$  for  $w$  the number of entities.

235 It is important to observe that, for each transition, our cCNA encoding requires all the processes  $P_a$ , with  $a \in A$ , and  $Cxt_s$  and  $P_s$ , with  $s \in S$ , be interacting in that transition. This is due to the fact that all the channels  $r_j$ ,  $p_j$ ,  $cxt_h$ , and  $s_{hi}$ , and  $s_{ho}$  are restricted. 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.

240 **Lemma 1.** Let  $\mathcal{A} = (S, A)$  be a RS and let  $\pi = (\gamma, \delta)$  be an extended interactive process in  $\mathcal{A}$ . Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  its cCNA translation. If exists  $P'$  such that  $t = (P \xrightarrow{(\nu \text{ names})\nu} P')$  is a transition of  $P$ , then

- 245 1. for each reaction  $a_j \in A$ , the corresponding channels  $r_j$  and  $p_j$  appear in  $\nu$ ; for each entity  $s_h \in S$  (where  $h$  is the identifying number of  $s$ ), the corresponding channel  $s_h$  (suitably decorated), and the corresponding channel  $cxt_h$  appear in  $\nu$ ;
- 250 2. for each reaction  $a \in A$  and each entity  $s \in S$ , each virtual link offered by processes  $P_a$  and  $Cxt_s$  is overlapped by exactly one solid link offered by processes representing entities.

*Example 1.* Let  $\mathcal{A}$  be a RS whose specification contains two entities,  $s1$  and  $s2$ , and, among the others, the reaction  $a = (s1, \cdot, s1)$  that guarantees the presence of the  $s1$  in the system. Then, we assume an extended interactive process

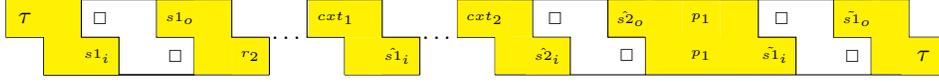


Fig. 2: The link chain structure arising from reactions and context processes.

$\pi = (\gamma, \delta)$  where the context  $\gamma$  provides  $s1$  and  $s2$ . Our translation includes the processes:

$$\begin{aligned}
P_a &\triangleq \tau \backslash_{s1_i} \square \backslash_{s1_o} \square \backslash_{r2} \square \backslash_{s1_i} \square \backslash_{p2} \square \backslash_{p1} \square \backslash_{s1_o} \square \backslash_{p2} \cdot P_a + \dots; \\
P_{s1} &\triangleq s1_i \backslash_{s1_o} \square \backslash_{s1_i} \square \backslash_{s1_o} \square \backslash_{s1_i} \square \backslash_{s1_o} \cdot P_{s1} + \dots; & P_{s2} &\triangleq \hat{s}2_i \backslash_{s2_o} \cdot P_{s2} + \dots; \\
Cxt_{s1} &\triangleq cxt1 \backslash_{s1_i} \square \backslash_{s1_o} \square \backslash_{cxt2} \cdot Cxt_{s1} & Cxt_{s2} &\triangleq cxt2 \backslash_{s2_i} \square \backslash_{s2_o} \square \backslash_{p1} \cdot Cxt_{s2}
\end{aligned}$$

Now, we assume that  $s1$  is in the initial state of  $\mathcal{A}$ , and in Figure 2 we show the structure of a link chain label related to the execution of a transition of the cCNA system:  $(\nu names)(P_{s1}|P_{s2}|P_a|\dots|Cxt_{s1}|Cxt_{s2})$ . The yellow blocks are referred to the processes encoding the reactions ( $P_a$ , in our case) and the contexts ( $Cxt_{s1}$  and  $Cxt_{s2}$ ). 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  $\tau$  to the right one. We could say that these processes form the *backbone* of the interaction. In contrast, the processes encoding the entities ( $P_{s1}$ , and  $P_{s2}$ , in our case) provides the solid links to overlap the virtual links of the backbone.

Example 1 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.

With the next proposition, we analyse the structure of a cCNA process encoding of a reactive process after one step transition. 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  $\delta = \{D_i\}_{i \in \mathbb{N}}$ . Moreover, we denote by  $\pi^j$  the shift of  $\pi$  starting at the  $j$ -th state sequence; formally we let  $\pi^j = (\gamma^j, \delta^j)$  with  $\gamma^j = \{C'_i\}_{i \in \mathbb{N}}$ ,  $\delta^j = \{D'_i\}_{i \in \mathbb{N}}$  with  $C'_0 = C_j \cup D_j$ , and  $C'_i = C_{i+j}$ ,  $D'_i = D_{i+j}$  for any  $i \geq 1$ .

**Proposition 1 (Correctness 1).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  with*

$$P = (\nu names)(\Pi_{a \in A} P_a | \Pi_{s \in S} Cxt_s | \Pi_{s \in C_0} P_s | \Pi_{s \notin C_0} \bar{P}_s).$$

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

$$v = \tau \backslash_{\tau} \dots \tau \backslash_{\tau} \text{ and}$$

$$P' = (\nu names)(\Pi_{a \in A} P_a | \Pi_{s \in S} Cxt_s | \Pi_{s \in C_1 \cup D_1} P_s | \Pi_{s \notin C_1 \cup D_1} \bar{P}_s).$$

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

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

**Corollary 1 (Correctness 2).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  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  $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$ .*

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  $\llbracket \mathcal{A}, \gamma \rrbracket$  can simulate all the evolutions of  $\pi$ .

**Proposition 2 (Completeness 1).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  and  $\pi^1 = (\gamma^1, \delta^1)$ . Then,*  
285  $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$ .

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

**Corollary 2 (Completeness 2).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  and  $\pi^j = (\gamma^j, \delta^j)$ . Then,*  
 $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau^j} P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$ .

## 5 Example: *lac* operon

290 In this section we present the encoding of a RS example taken from [14].

### 5.1 The *lac* operon

An operon is a cluster of genes under the control of a single promoter. The *lac* operon is involved in the metabolism of lactose in *Escherichia coli* cells; it is composed by three adjacent structural genes (plus some regulatory components):  
295 *lacZ*, *lacY* and *lacA* encoding for two enzymes *Z* and *A*, and a transporter *Y*, involved in the digestion of the *lactose*. The main regulations are:

- the gene *lacI* encodes for a repressor protein *I*;
- the DNA sequence, called *promoter*, is recognised by a RNA polymerase to initiate the transcription of the genes *lacZ*, *lacY* and *lacA*;
- 300 – a DNA segment, called the *operator* (*OP*), obstructs the RNA polymerase functionality when the repressor protein *I* is bound to it forming *I-OP*;
- a short DNA sequence, called the *CAP-binding site*, when it is bound to the complex composed by the protein *CAP* and the signal molecule *cAMP*, acts as a promoter for the interaction between the RNA polymerase and the  
305 promoter.

The functionality of the *lac* operon depends on the integration of two control mechanisms, one mediated by *lactose*, and the other one mediated by *glucose*.

In the first control mechanism, an effect of the absence of the *lactose* is that *I* is able to bind the operator sequence preventing the *lac* operon expression. If  
310 lactose is available, *I* is unable to bind the operator sequence, and the *lac* operon can be potentially expressed.

In the second control mechanism, when glucose is absent, the molecule *cAMP* and the protein *CAP* increase the *lac* operon expression, thanks to the fact that the binding between the molecular complex *cAMP-CAP* and the *CAP-binding site*  
315 increases. In summary, the condition promoting the operon gene expression is when the *lactose* is present and the *glucose* is absent.

In the following we report the description of the *lac* operon mechanism in the reaction system formalism and then show its encoding in cCNA.

## 5.2 The RS formalization

The reaction system for the *lac* operon is defined as  $A_{lac} = (S, A)$ , where the set  $S$  represents the main biochemical components involved in this genetic system, while the reaction set  $A$  contains the biochemical reactions involved in the regulation of the *lac* operon expression. Formally, the *lac* operon reaction system is defined as follows:  $S$  is the set

$\{lac, Z, Y, A, lacI, I, I-OP, cya, cAMP, crp, CAP, cAMP-CAP, lactose, glucose\}$ , and  $A$  consists of the following 10 reactions:

$$\begin{aligned} a_1 &= (\{lac\}, \{\dots\}, \{lac\}), & a_6 &= (\{cya\}, \{\dots\}, \{cAMP\}), \\ a_2 &= (\{lacI\}, \{\dots\}, \{lacI\}), & a_7 &= (\{crp\}, \{\dots\}, \{crp\}), \\ a_3 &= (\{lacI\}, \{\dots\}, \{I\}), & a_8 &= (\{crp\}, \{\dots\}, \{CAP\}), \\ a_4 &= (\{I\}, \{lactose\}, \{I-OP\}), & a_9 &= (\{cAMP, CAP\}, \{glucose\}, \{cAMP-CAP\}), \\ a_5 &= (\{cya\}, \{\dots\}, \{cya\}), & a_{10} &= (\{lac, cAMP-CAP\}, \{I-OP\}, \{Z, Y, A\}). \end{aligned}$$

320 The default context ( $DC$ ) is composed by those entities that are always present in the system  $DC = \{lac, lacI, I, cya, cAMP, crp, CAP\}$ , whereas the *lactose* and the *glucose* are given non-deterministically by the context.

## 5.3 The RS encoding

325 For the sake of readability, the encoding we propose exploits the specific features of the example in hand to perform some simplifications:

- for the entities in the default context,  $s \in DC$ , as they are persistent, we do not provide the  $\overline{P}_s$  processes and the  $Cxt_s$  processes;
- for the reactions requiring the presence of entities  $s \in DC$ , we do not provide the reaction alternative behaviour for when  $s$  is absent;
- 330 – the  $Cxt_s$  processes are specified only for those entities that are really provided by the context.

Moreover, we do not model the *dummy* entity that is specified by dots (...) by the RS reactions in Section 5.2. Finally, we exclude the *duplication reactions* ( $a_1, a_2, a_5, a_7$ ), and renumber the remaining reactions :

old	new	reactions
$a_3$	$a_1$	$= (\{lacI\}, \{\dots\}, \{I\}),$
$a_4$	$a_2$	$= (\{I\}, \{lactose\}, \{I-OP\}),$
$a_6$	$a_3$	$= (\{cya\}, \{\dots\}, \{cAMP\}),$
$a_8$	$a_4$	$= (\{crp\}, \{\dots\}, \{CAP\}),$
$a_9$	$a_5$	$= (\{cAMP, CAP\}, \{glucose\}, \{cAMP-CAP\}),$
$a_{10}$	$a_6$	$= (\{lac, cAMP-CAP\}, \{I-OP\}, \{Z, Y, A\}).$

*Duplication reactions.* As explained previously, their encoding is omitted.

*Expression reactions.* First we define the parametric process

$$P_i(s1, s2) \triangleq r_i \backslash_{s1_i} \square \backslash_{s1_o} \square \backslash_{r_{i+1}} \square \backslash_{p_i} \square \backslash_{s2_i} \square \backslash_{s2_o} \square \backslash_{p_{i+1}} . P_i(s1, s2)$$

Then, we let  $P_{a1} \triangleq P_1(lacI, I)$ ,  $P_{a3} \triangleq P_3(cya, cAMP)$ , and  $P_{a4} \triangleq P_4(crp, CAP)$ .

Regulation reactions.

$$\begin{aligned}
P_{a2} &\triangleq r_2 \frac{\square}{I_i} \setminus I_o \frac{\square}{lactose_i} \setminus \overline{lactose_o} \setminus r_3 \setminus \frac{\square}{p_2} \setminus \frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \setminus p_3 \cdot P_{a2} \\
&+ \\
&r_2 \frac{\square}{lactose_i} \setminus lactose_o \setminus r_3 \setminus \frac{\square}{p_2} \setminus p_3 \cdot P_{a2} + r_2 \frac{\square}{\widetilde{I}_i} \setminus \frac{\square}{\widetilde{I}_o} \setminus r_3 \setminus \frac{\square}{p_2} \setminus p_3 \cdot P_{a2} \\
P_{a5} &\triangleq r_5 \frac{\square}{cAMP_i} \setminus \frac{\square}{cAMP_o} \setminus \frac{\square}{CAP_i} \setminus \frac{\square}{CAP_o} \setminus \frac{\square}{glucose_i} \setminus \frac{\square}{glucose_o} \setminus r_6 \setminus \frac{\square}{p_5} \setminus \frac{\square}{cAMP-CAP_i} \setminus \frac{\square}{cAMP-CAP_o} \setminus p_6 \cdot P_{a5} \\
&+ \\
&r_5 \frac{\square}{glucose_i} \setminus glucose_o \setminus r_6 \setminus \frac{\square}{p_5} \setminus p_6 \cdot P_{a5} \\
&+ \\
&r_5 \frac{\square}{cAMP_i} \setminus \frac{\square}{cAMP_o} \setminus r_6 \setminus \frac{\square}{p_5} \setminus p_6 \cdot P_{a5} + r_5 \frac{\square}{CAP_i} \setminus \frac{\square}{CAP_o} \setminus r_6 \setminus \frac{\square}{p_5} \setminus p_6 \cdot P_{a5} \\
P_{a6} &\triangleq r_6 \frac{\square}{lac_i} \setminus lac_o \setminus \frac{\square}{cAMP_i} \setminus \frac{\square}{cAMP_o} \setminus \frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \setminus p_6 \setminus \frac{\square}{cxt_1} \setminus \frac{\square}{\tilde{z}_i} \setminus \frac{\square}{\tilde{z}_o} \setminus \frac{\square}{\tilde{y}_i} \setminus \frac{\square}{\tilde{y}_o} \setminus \frac{\square}{\tilde{A}_i} \setminus \frac{\square}{\tilde{A}_o} \setminus \tau \cdot P_{a6} \\
&+ \\
&r_6 \frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \setminus \frac{\square}{cxt_1} \setminus \frac{\square}{p_6} \setminus \tau \cdot P_{a6} \\
&+ \\
&r_6 \frac{\square}{lac_i} \setminus \frac{\square}{lac_o} \setminus \frac{\square}{cxt_1} \setminus \frac{\square}{p_6} \setminus \tau \cdot P_{a6} + \frac{\square}{cAMP_i} \setminus \frac{\square}{cAMP_o} \setminus \frac{\square}{cxt_1} \setminus \frac{\square}{p_6} \setminus \tau \cdot P_{a6}
\end{aligned}$$

335

Processes for the entities. We exploit the specificity of the example in hand to optimising the code, and we specify exactly the number of solid links that each process encoding an entity must offer. For the always present entities we let:

$$\begin{aligned}
P_{cya} &\triangleq cya_i \setminus cya_o \cdot P_{cya} & P_{crp} &\triangleq crp_i \setminus crp_o \cdot P_{crp} \\
P_{lacI} &\triangleq lacI_i \setminus lacI_o \cdot P_{lacI} & P_{lac} &\triangleq lac_i \setminus lac_o \cdot P_{lac}
\end{aligned}$$

For the entities always produced (i.e. not present only at the first step), we provide a parametric definition  $P_e(s) \triangleq s_i \setminus \frac{\square}{s_o} \setminus \frac{\square}{\tilde{s}_i} \setminus \frac{\square}{\tilde{s}_o} \cdot P_e(s) + \tilde{s}_i \setminus \frac{\square}{\tilde{s}_o} \cdot P_e(s)$ . There are three entities of the second type:

$$P_{cAMP} \triangleq P_e(cAMP) \quad P_{CAP} \triangleq P_e(CAP) \quad P_I \triangleq P_e(I).$$

The entity  $I-OP$  can be either produced (by  $a_2$ ) or tested for absence (by  $a_6$ ). Correspondingly, the process  $P_{I-OP}$  is defined as follows:

$$\begin{aligned}
P_{I-OP} &\triangleq \sum_{h=0}^1 (I-OP_i \setminus \frac{\square}{I-OP_o} \setminus \frac{\square}{\square})^h \frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \cdot P_{I-OP} + I-OP_i \setminus \frac{\square}{I-OP_o} \cdot \overline{P_{I-OP}} \\
\overline{P_{I-OP}} &\triangleq \sum_{h=0}^1 (\frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \setminus \frac{\square}{\square})^h \frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \cdot P_{I-OP} + \frac{\square}{\widetilde{I-OP}_i} \setminus \frac{\square}{\widetilde{I-OP}_o} \cdot \overline{P_{I-OP}}
\end{aligned}$$

The process  $P_{cAMP-CAP}$  is similar to  $P_{I-OP}$ , as it is produced by  $a_5$  and tested for presence by  $a_6$ . Its code is in Table 2, in the Appendix. The *lactose* is provided by the context and tested for absence by  $a_2$ .

340

$$\begin{aligned}
P_{lactose} &\triangleq \sum_{h=0}^1 (lactose_i \setminus \frac{\square}{lactose_o} \setminus \frac{\square}{\square})^h \frac{\square}{lactose_i} \setminus \frac{\square}{lactose_o} \cdot P_{lactose} \\
&+ \\
&\sum_{h=0}^1 (lactose_i \setminus \frac{\square}{lactose_o} \setminus \frac{\square}{\square})^h \frac{\square}{lactose_i} \setminus \frac{\square}{lactose_o} \cdot \overline{P_{lactose}} \\
\overline{P_{lactose}} &\triangleq \sum_{h=0}^1 (\frac{\square}{lactose_i} \setminus \frac{\square}{lactose_o} \setminus \frac{\square}{\square})^h \frac{\square}{lactose_i} \setminus \frac{\square}{lactose_o} \cdot P_{lactose} \\
&+ \\
&\sum_{h=0}^1 (\frac{\square}{lactose_i} \setminus \frac{\square}{lactose_o} \setminus \frac{\square}{\square})^h \frac{\square}{lactose_i} \setminus \frac{\square}{lactose_o} \cdot \overline{P_{lactose}}
\end{aligned}$$

The process  $P_{glucose}$  is similar to  $P_{lactose}$  and tested for absence by  $a_5$ . Its code is in Table 3, in the Appendix. The entity  $z$  can only be produced by rule  $a_6$ , while it is never provided by the context. Moreover, there is no rule for testing its presence or absence.

$$P_z \triangleq \tilde{z}_i \backslash_{\tilde{z}_o} \square \backslash_{\square} \tilde{z}_i \backslash_{\tilde{z}_o} . P_z + \tilde{z}_i \backslash_{\tilde{z}_o} . \overline{P}_z \quad \overline{P}_z \triangleq \tilde{z}_i \backslash_{\tilde{z}_o} \square \backslash_{\square} \tilde{z}_i \backslash_{\tilde{z}_o} . \overline{P}_z + \tilde{z}_i \backslash_{\tilde{z}_o} . \overline{P}_z$$

The entities  $y$  and  $A$  are treated in the same way as  $z$ . Their processes are in Table 4 in the Appendix.

*Context.* The entities in  $DC$  are assumed always present by default, so no context process is needed for them. The entities  $z$ ,  $y$ , and  $A$  are assumed never provided by the context. Their processes are

$$\begin{aligned} Cxt_z &\triangleq cxt_1 \backslash_{\tilde{z}_i} \square \backslash_{\square} \tilde{z}_o \backslash_{\tilde{z}_o} . Cxt_z & Cxt_y &\triangleq cxt_2 \backslash_{\tilde{y}_i} \square \backslash_{\square} \tilde{y}_o \backslash_{\tilde{y}_o} . Cxt_y \\ Cxt_A &\triangleq cxt_3 \backslash_{\tilde{A}_i} \square \backslash_{\square} \tilde{A}_o \backslash_{\tilde{A}_o} . Cxt_A \end{aligned}$$

For the sake of presentation, we assume that the *lactose* is always provided by the context, in contrast, *glucose* is never provided.

$$\begin{aligned} Cxt_{lactose} &\triangleq cxt_4 \backslash_{lactose_i} \square \backslash_{\square} \widehat{lactose}_o \backslash_{cxt_5} . Cxt_{lactose} \\ Cxt_{glucose} &\triangleq cxt_5 \backslash_{glucose_i} \square \backslash_{\square} \widehat{glucose}_o \backslash_{p_1} . Cxt_{glucose} \end{aligned}$$

In the following we let  $CXT \triangleq Cxt_z | Cxt_y | Cxt_A | Cxt_{lactose} | Cxt_{glucose}$  be the processes for context. The whole system is as follows:

$$lacOp \triangleq (\nu names)(\Pi_{i=1}^6 P_{ai} | \Pi_{s \in DC} P_s | \Pi_{s \in S \setminus DC} \overline{P}_s | CXT)$$

*Execution.* Now, we show two transitions. After the first transition the entity *cAMP-CAP* is produced due to the absence of *glucose*, while the presence of *lactose* inhibits the production of *I-OP*:  $lacOp \xrightarrow{(\nu names)v} lacOp'$  where:

$$\begin{aligned} v &= \tau \backslash_{lac_i} \dots \widehat{lactose}_o \backslash_{r_3} \dots r_5 \backslash_{cAMP_i} \dots \widehat{glucose}_o \backslash_{r_6} \dots p_5 \backslash_{cAMP-CAP_i} \dots p_6 \backslash_{\tau} \\ lacOp' &\triangleq (\nu names)(\Pi_{i=1}^6 P_{ai} | \Pi_{s \in AP} P_s | \Pi_{s \in S \setminus AP} \overline{P}_s | CXT) \end{aligned}$$

with  $AP = DC \cup \{cAMP-CAP\}$  the actual context.

After the second step the entities  $z$ ,  $y$  and  $A$  are produced, due to the presence of *cAMP-CAP* and the absence of *I-OP*, thus  $lacOp' \xrightarrow{(\nu names)v'} lacOp''$  where:

$$\begin{aligned} v' &= \tau \backslash_{lac_i} \dots lac_o \backslash_{cAMP-CAP_i} \dots \overline{I-OP}_o \backslash_{cxt_1} \dots \widehat{glucose}_o \backslash_{p_1} \backslash_{p_6} \dots \tilde{z}_i \backslash_{\tilde{z}_o} \tilde{y}_i \backslash_{\tilde{y}_o} \tilde{A}_i \backslash_{\tilde{A}_o} \backslash_{\tau} \\ lacOp'' &\triangleq (\nu names)(\Pi_{i=1}^6 P_{ai} | \Pi_{s \in AP'} P_s | \Pi_{s \in S \setminus AP'} \overline{P}_s | CXT) \end{aligned}$$

with  $AP' = DC \cup \{z, y, A\}$ .

## 6 Enhanced Reaction Systems

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

### 350 6.1 Recursive contexts

In RS, the behaviour of the context is finite. For the first  $n$  steps, it is specified which are the entities that are provided from the context. Using cCNA we can describe in a natural way the behaviour of the context in a recursive way. Then, the context behaviour would not necessarily end after  $n$  steps, and could be infinite. For example, in an extended interactive process, we may want that the entity  $s$  is intermittently provided by the context every two steps:

$$\begin{aligned} Cxt_s &\triangleq cxt_j \setminus_{\square}^{\hat{s}_i} \setminus_{\square}^{\hat{s}_o} \setminus cxt_{j+1}. Cxt_{s'}, && \text{context provides } s; \\ Cxt_{s'} &\triangleq cxt_j \setminus_{\square}^{\hat{s}_i} \setminus_{\square}^{\hat{s}_o} \setminus cxt_{j+1}. Cxt_{s''}, && \text{context doesn't provide } s; \\ Cxt_{s''} &\triangleq cxt_j \setminus_{\square}^{\hat{s}_i} \setminus_{\square}^{\hat{s}_o} \setminus cxt_{j+1}. Cxt_s, && \text{context doesn't provide } s. \end{aligned}$$

### 6.2 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  
355 version of the entity can take part to only a subset of the rules 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  
360 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 non deterministic choice in the code of  $P_{a1}$  and  $P_{a2}$ .

### 6.3 Communicating reaction systems

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

*Example 2.* Let  $rs1$  and  $rs2$  be two RSs, defined by the rules in Table 1. 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

$rs1$	$rs2$
$a_1 = (s, , x)$	$a_2 = (y, , s)$

Table 1: The two reaction systems  $rs1$  and  $rs2$ .

the name  $ss$  in  $rs2$ . We need to modify our translation technique to suite 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  $rs1$ :

$$[[a_1]] \triangleq P_{a_1} \triangleq \tau \backslash_{s_i} \square \backslash_{s_o} \square \backslash_{\tilde{x}_i} \square \backslash_{\tilde{x}_o} \square \backslash_{a_2} \cdot P_{a_1} + \tau \backslash_{\bar{s}_i} \square \backslash_{\bar{s}_o} \square \backslash_{dx_i} \square \backslash_{dx_o} \square \backslash_{a_2} \cdot P_{a_1}$$

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 translation for the entities follows.

$$[[s]] \triangleq \frac{P_s \triangleq s_i \backslash_{s_o} \square \backslash_{\hat{s}_i} \square \backslash_{\hat{s}_o} \cdot P_s + s_i \backslash_{s_o} \cdot \overline{P_s}}{P_s \triangleq \bar{s}_i \backslash_{\bar{s}_o} \square \backslash_{\hat{s}_i} \square \backslash_{\hat{s}_o} \cdot P_s + \bar{s}_i \backslash_{\bar{s}_o} \cdot \overline{P_s}} \quad [[x]] \triangleq \frac{P_x \triangleq \tilde{x}_i \backslash_{\tilde{x}_o} \cdot P_x + dx_i \backslash_{dx_o} \cdot \overline{P_x}}{P_x \triangleq \tilde{x}_i \backslash_{\tilde{x}_o} \cdot P_x + dx_i \backslash_{dx_o} \cdot \overline{P_x}}$$

The translation for the  $rs2$  follows.

$$[[a_2]] \triangleq P_{a_2} \triangleq a_2 \backslash_{y_i} \square \backslash_{y_o} \square \backslash_{\bar{s}_i} \square \backslash_{\bar{s}_o} \square \backslash_{\tau} \cdot P_{a_2} + a_2 \backslash_{\bar{y}_i} \square \backslash_{\bar{y}_o} \square \backslash_{dss_i} \square \backslash_{dss_o} \square \backslash_{\tau} \cdot P_{a_2}$$

In the translation 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  $\hat{s}_i \backslash_{\hat{s}_o}$ :

$$[[ss]] \triangleq \frac{P_{ss} \triangleq \bar{s}_i \backslash_{\bar{s}_o} \square \backslash_{\hat{s}_i} \square \backslash_{\hat{s}_o} \cdot P_{ss} + dss_i \backslash_{dss_o} \cdot \overline{P_{ss}}}{P_{ss} \triangleq \bar{s}_i \backslash_{\bar{s}_o} \square \backslash_{\hat{s}_i} \square \backslash_{\hat{s}_o} \cdot P_{ss} + dss_i \backslash_{dss_o} \cdot \overline{P_{ss}}} \quad [[y]] \triangleq \frac{P_y \triangleq y_i \backslash_{y_o} \cdot \overline{P_y}}{P_y \triangleq \bar{y}_i \backslash_{\bar{y}_o} \cdot \overline{P_y}}$$

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):

$$\tau \backslash_{s_i} \backslash_{s_o} \square \backslash_{\tilde{x}_i} \square \backslash_{\tilde{x}_o} \square \backslash_{a_2} \backslash_{y_i} \backslash_{y_o} \square \backslash_{\bar{s}_i} \square \backslash_{\bar{s}_o} \square \backslash_{\hat{s}_i} \square \backslash_{\hat{s}_o} \square \backslash_{\bar{s}_i} \square \backslash_{\bar{s}_o} \square \backslash_{\tau},$$

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. In Example 2 describing the behaviour of the *lac* operon, the two entities lactose and glucose are controlled non deterministically by the context. In our framework, instead, by exploiting the expressivity of cCNA, the *lac* operon system has been connected with the two systems producing the lactose and the glucose. This way, the presence of these two entities in the *lac* operon system can be regulated by realistic mechanisms.

## 7 Conclusion

In this paper we have introduced a variant of the `link`-calculus where prefixes are link chains and no more single links, as it was briefly described in the future work section in [6]. This variant allowed us to define an elegant embedding of reaction systems, an emerging formalism to model computationally biochemical systems. This translation shows several benefits. For instance, the context behaviour can also be expressed recursively. Entity mutations can be expressed easily. Reaction systems can communicate between them.

We believe that our embedding can contribute to extend the applications of reaction systems to diverse fields of computer science, and life sciences. As we have already mentioned, the evolution of each process resulting from our embedding is deterministic, thus we do not have the problem of having infinitely many transitions in the produced labelled transition system. In any case, we can exploit the implementation of the symbolic semantics of the `link`-calculus [11] that can be found in [17].

As future work, we plan to implement a prototype of our framework, with an automatic translation from RSs to `link`-calculus. We believe that our work can also help to extend the framework of RSs towards a model which can improve the communication between different RSs. We also believe that our work can make possible to investigate how to apply formal techniques to prove properties of the modeled systems [13,18,8].

## References

1. S. Azimi. Steady states of constrained reaction systems. *Theor. Comput. Sci.*, 701(C):20–26, 2017.
2. S. Azimi, B. Iancu, and I. Petre. Reaction system models for the heat shock response. *Fundamenta Informaticae*, 131(3-4):299–312, 2014.
3. R. Barbuti, R. Gori, F. Levi, and P. Milazzo. Investigating dynamic causalities in reaction systems. *Theor. Comput. Sci.*, 623:114–145, 2016.
4. A. Bernini, L. Brodo, P. Degano, M. Falaschi, and D. Hermith. Process calculi for biological processes. *Natural Computing*, 17(2):345–373, 2018.
5. C. Bodei, L. Brodo, and R. Bruni. Open multiparty interaction. In *Recent Trends in Algebraic Development Techniques, 21st International Workshop, WADT 2012*, volume 7841 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2012.
6. C. Bodei, L. Brodo, and R. Bruni. A formal approach to open multiparty interactions. *Theoretical Computer Science*, 763:38–65, 2019.
7. C. Bodei, L. Brodo, R. Bruni, and D. Chiarugi. A flat process calculus for nested membrane interactions. *Sci. Ann. Comp. Sci.*, 24(1):91–136, 2014.
8. C. Bodei, L. Brodo, R. Gori, F. Levi, A. Bernini, and D. Hermith. A static analysis for Brane Calculi providing global occurrence counting information. *Theoretical Computer Science*, 696:11–51, 2017.
9. R. Brijder, A. Ehrenfeucht, M. Main, and G. Rozenberg. A tour of reaction systems. *International Journal of Foundations of Computer Science*, 22(07):1499–1517, 2011.
10. L. Brodo. On the expressiveness of pi-calculus for encoding mobile ambients. *Mathematical Structures in Computer Science*, 28(2):202–240, 2018.

11. L. Brodo and C. Olarte. Symbolic semantics for multiparty interactions in the link-calculus. In *Proc. of SOFSEM'17*, volume 10139 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2017.
12. L. Cardelli and A. D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.
13. D. Chiarugi, M. Falaschi, D. Hermith, C. Olarte, and L. Torella. Modelling non-markovian dynamics in biochemical reactions. *BMC Systems Biology*, 9(S-3):S8, 2015.
14. L. Corolli, C. Maj, F. Marina, D. Besozzi, and G. Mauri. An excursion in reaction systems: From computer science to biology. *Theoretical Computer Science*, 454:95–108, 2012.
15. A. Męski, W. Penczek, and G. Rozenberg. Model checking temporal properties of reaction systems. *Information Sciences*, 313:22–42, 2015.
16. F. Okubo and T. Yokomori. The computational capability of chemical reaction automata. *Natural Computing*, 15(2):215–224, 2016.
17. C. Olarte. SiLVer: Symbolic links verifier, Dec. 2018. <http://subsell.logic.at/links/links-web/index.html>.
18. C. Olarte, D. Chiarugi, M. Falaschi, and D. Hermith. A proof theoretic view of spatial and temporal dependencies in biochemical systems. *Theor. Comput. Sci.*, 641:25–42, 2016.

## A Omitted Proofs

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

**Lemma 1.** *Let  $\mathcal{A} = (S, A)$  be a RS and let  $\pi = (\gamma, \delta)$  be an extended interactive process in  $\mathcal{A}$ . Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  its cCNA translation. If exists  $P'$  such that  $t =$   
 450  $(P \xrightarrow{(\nu \text{names})v} P')$  is a transition of  $P$ , then*

1. for each reaction  $a_j \in A$ , the corresponding channels  $r_j$  and  $p_j$  appear in  $v$ ; for each entity  $s_h \in S$  (where  $h$  is the identifying number of  $s$ ), the corresponding channel  $s_h$  (suitably decorated), and the corresponding channel  $ctx_h$  appear in  $v$ ;
- 455 2. for each reaction  $a \in A$  and each entity  $s \in S$ , each virtual link offered by processes  $P_a$  and  $Cxt_s$  is overlapped by exactly one solid link offered by processes representing entities.

*Proof.* We prove the two items separately:

1. by Definition 10, all the names  $r_j, p_j$ ; all the names  $s_i, s_o$ , in all their annotated versions, and all the names  $ctx_h$ , are restricted. Moreover, the only  
 460 prefixes that start with  $\tau$  are those associated to reaction  $a_1$  (as  $r_1 = \tau$ ) and the only prefixes that end with  $\tau$  are those associated to reaction  $a_u$  (as  $p_{u+1} = \tau$ , where  $u$  is the number of reactions). Thus all the prefixes associated with reactions and contexts must be concatenated (remember that  
 465  $r_{u+1} = ctx_1$  and  $ctx_{w+1} = p_1$ , where  $w$  is the number of entities), forming the backbone of the label. Since all context processes are involved, then all entities processes are also involved. Then, all the processes  $P_a$ ,  $P_s$  (or  $\overline{P_s}$ ), and  $Cxt_h$  must participate to each transition.
2. for each reaction  $a_j \in A$ , the cCNA code of  $P_{a_j}$  leaves one virtual link between  
 470 two solid links of the types  $r_j \setminus \dots \setminus \square \setminus s_i \setminus \square \setminus \dots \setminus r_{j+1} \setminus \dots \setminus p_j \setminus \dots \setminus p_{j+1}$ . 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 type  $s^i \setminus s_o$ . 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  $Cxt_h$  and for all  
 475 the annotated versions of  $s_i, s_o$ .

**Proposition 1 (Correctness 1).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  with*

$$P = (\nu \text{names})(\Pi_{a \in A} P_a | \Pi_{s \in S} Cxt_s | \Pi_{s \in C_0} P_s | \Pi_{s \notin C_0} \overline{P_s}).$$

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

$$v = \tau \setminus \tau \dots \tau \setminus \tau \text{ and}$$

$$480 P' = (\nu \text{names})(\Pi_{a \in A} P_a | \Pi_{s \in S} Cxt_s | \Pi_{s \in C_1 \cup D_1} P_s | \Pi_{s \notin C_1 \cup D_1} \overline{P_s}).$$

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

*Proof.* First, we note that all the channels in the system are restricted, see Def. 10, then it holds that the transition labels are of the form  $v = \tau \setminus \tau \dots \tau \setminus \tau$ . Now, by Definition 10 and by Lemma 1.1, all the channels  $r_j, p_j$ , with  $j \in$   
 485  $[1, \dots, u]$ , and  $ctx_h$ , with  $h \in [1, \dots, w]$ , and all the annotated versions of  $s_i, s_o$

are restricted. Also, processes  $Cxt_s$  always requires the interaction with  $P_s$  on either on channels  $\hat{s}_i, \hat{s}_o$  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  $Cxt_s$  (coding the behaviour of the context regarding the entity  $s$ ) have been involved in the transition. We have the following cases:

- (a) processes  $P_a$  encoding a reaction  $a$ , with serial number  $j$ , producing the entity  $s$  provide a code of this type:  $P_a \triangleq r_j \backslash \dots \backslash_{r_{j+1}} \backslash_{\square}^{p_j} \backslash \dots \backslash_{\hat{s}_i} \backslash_{\square}^{\bar{s}_o} \backslash \dots \backslash_{p_{j+1}} .P_a$ ;
- (b) processes  $P_a$  encoding a reaction  $a$ , with serial number  $j$ , consuming the entity  $s$  provide a code of this type:  $P_a \triangleq r_j \backslash \dots \backslash_{\hat{s}_i} \backslash_{\square}^{s_o} \backslash \dots \backslash_{r_{j+1}} \backslash_{\square}^{p_j} \backslash \dots \backslash_{p_{j+1}} .P_a$ ;
- 495 (c) processes  $P_a$  encoding a reaction  $a$ , with serial number  $j$ , requiring the absence of the entity  $s$  provide a code of this type:  
 $P_a \triangleq r_j \backslash \dots \backslash_{\hat{s}_i} \backslash_{\square}^{\bar{s}_o} \backslash \dots \backslash_{r_{j+1}} \backslash_{\square}^{p_j} \backslash \dots \backslash_{p_{j+1}} .P_a$ ;
- (d) processes  $P_a$  encoding a reaction  $a$  (that cannot be applied), with serial number  $j$ , execute a code capturing either the absence of one of its reactants (case 1), or the presence of one of its inhibitors (case 2):  
500 1.  $P_a \triangleq r_j \backslash \dots \backslash_{\hat{s}_i} \backslash_{\square}^{\bar{s}_o} \backslash \dots \backslash_{r_{j+1}} \backslash_{\square}^{p_j} \backslash \dots \backslash_{p_{j+1}} .P_a$ .  
2.  $P_a \triangleq r_j \backslash \dots \backslash_{\hat{s}_i} \backslash_{\square}^{s_o} \backslash \dots \backslash_{r_{j+1}} \backslash_{\square}^{p_j} \backslash \dots \backslash_{p_{j+1}} .P_a$ .

Now, we consider the structure of the process  $Cxt_s = Cxt_s^0$ . By Definition 10,  $Cxt_s^0$  is the unique process encoding the behaviour of the context regulating the supply of  $s$ . The code of the process  $Cxt_s^t$ , with  $t \geq 0$ , has the following structure:

- (e)  $Cxt_s^i \triangleq cxt_h \backslash_{\hat{s}_i} \backslash_{\square}^{\bar{s}_o} \backslash cxt_{h+1} .Cxt_s^{i+1}$ , if  $s \in C_{i+1}$ ;
- (f)  $Cxt_s^i \triangleq cxt_h \backslash_{\underline{s}_i} \backslash_{\square}^{\bar{s}_o} \backslash cxt_{h+1} .Cxt_s^{i+1}$ , if  $s \notin C_{i+1}$ ;

The code executed by  $P_s$  has the following structure:

- (g)  $P_s \triangleq \sum_{h,k \geq 0} (s_i \backslash_{s_o} \backslash_{\square})^h \hat{s}_i \backslash_{\hat{s}_o} \backslash_{\square} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^k .P_s$ , if  $s \in C_{i+1}$ ;
- 510 (h)  $P_s \triangleq \sum_{h \geq 0, k \geq 1} (s_i \backslash_{s_o} \backslash_{\square})^h \underline{s}_i \backslash_{\underline{s}_o} \backslash_{\square} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^k .P_s$ , if  $s \notin C_{i+1}$
- (i)  $P_s \triangleq \sum_{h \geq 0} (s_i \backslash_{s_o} \backslash_{\square})^h \underline{s}_i \backslash_{\underline{s}_o} .\bar{P}_s$ , if  $s \notin C_{i+1}$

where, by Lemma 1.2,  $h$  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  $s$ . The code executed by  $\bar{P}_s$  has the following structure:

- 515 (g')  $\bar{P}_s \triangleq \sum_{h,k \geq 0} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^h \hat{s}_i \backslash_{\hat{s}_o} \backslash_{\square} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^k .P_s$ , if  $s \in C_{i+1}$ ;
- (h')  $\bar{P}_s \triangleq \sum_{h \geq 0, k \geq 1} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^h \underline{s}_i \backslash_{\underline{s}_o} \backslash_{\square} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^k .P_s$ , if  $s \notin C_{i+1}$
- (i')  $\bar{P}_s \triangleq \sum_{h \geq 0} (\bar{s}_i \backslash_{\bar{s}_o} \backslash_{\square})^h \underline{s}_i \backslash_{\underline{s}_o} .\bar{P}_s$ , if  $s \notin C_{i+1}$

where, by Lemma 1.2,  $h$  is the number of reactions requiring the absence of  $s$  plus possibly some reactions requiring  $s$ ; and  $k$  is the number of reactions producing  $s$ .

520 It is worth nothing that, depending on the presence ( $P_s$ ) or the absence ( $\bar{P}_s$ ) 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 processes  $Cxt_s$  (points (e), (f)), follows the evolution of  $\gamma$ . It derives that the trend followed by the processes  $P_s$  (or  $\bar{P}_s$ ) is also deterministic (points 525 (g), (h), (i) or (g'), (h'), (i')), leading to  $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$ .

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

530 *Proof.* We proceed by induction on the transition number  $j \geq 0$ .  
 case  $j = 1$

This case falls into the case of Proposition 1.

case  $j > 1$

By inductive hypothesis, it holds that  $\exists P'$  such that  $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau}^{j-1} P'$  and  
 535  $P' = \llbracket \mathcal{A}, \gamma^{j-1} \rrbracket$ .

As  $P'$  is the encoding of an extended interactive process, by Proposition 1, it exists  $P''$  such that  $P' \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P''$ , and  $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$ .

**Proposition 2 (Completeness 1).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  and  $\pi^1 = (\gamma^1, \delta^1)$ . Then,  $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$ .*

540 *Proof.* By Proposition 1, if there exists  $P'$  such that  $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P'$ , 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  $\pi$  in  $A$ , then process  $P$  can always execute a step transition, as its subprocesses  $P_a$  can always execute one of the  
 545 *alternative code for when reaction  $a$  is not applicable* (see Definition 10, code for  $P_a$  processes).

**Corollary 2 (Completeness 2).** *Let  $P = \llbracket \mathcal{A}, \gamma \rrbracket$  and  $\pi^j = (\gamma^j, \delta^j)$ . Then,  $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau}^j P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$ .*

550 *Proof.* The proof proceeds by induction on the number  $j$ , and it is similar to the one of Corollary 1

## B The *lac* operon encoding

$$\begin{aligned}
P_{cAMP-CAP} &\triangleq \sum_{h=0}^1 ({}^{cAMP-CAP_i} \setminus \square_{cAMP-CAP_o} \setminus \square)^h {}^{cAMP-CAP_i} \setminus \square_{cAMP-CAP_o} \cdot P_{cAMP-CAP} \\
&\quad + \overline{{}^{cAMP-CAP_i} \setminus \square_{cAMP-CAP_o} \cdot P_{cAMP-CAP}} \\
\overline{P_{cAMP-CAP}} &\triangleq \sum_{h=0}^1 (\overline{{}^{cAMP-CAP_i} \setminus \square_{cAMP-CAP_o} \setminus \square})^h \overline{{}^{cAMP-CAP_i} \setminus \square_{cAMP-CAP_o} \cdot P_{cAMP-CAP}} \\
&\quad + \overline{\overline{{}^{cAMP-CAP_i} \setminus \square_{cAMP-CAP_o} \cdot P_{cAMP-CAP}}}
\end{aligned}$$

Table 2: The process for *cAMP-CAP*.

$$\begin{aligned}
P_{glucose} &\triangleq \sum_{h=0}^1 ({}^{glucose_i} \setminus \square_{glucose_o} \setminus \square)^h {}^{glucose_i} \setminus \square_{glucose_o} \cdot P_{glucose} \\
&\quad + \sum_{h=0}^1 ({}^{glucose_i} \setminus \square_{glucose_o} \setminus \square)^h \overline{{}^{glucose_i} \setminus \square_{glucose_o} \cdot P_{glucose}} \\
\overline{P_{glucose}} &\triangleq \sum_{h=0}^1 (\overline{{}^{glucose_i} \setminus \square_{glucose_o} \setminus \square})^h \overline{{}^{glucose_i} \setminus \square_{glucose_o} \cdot P_{glucose}} \\
&\quad + \sum_{h=0}^1 (\overline{{}^{glucose_i} \setminus \square_{glucose_o} \setminus \square})^h \overline{\overline{{}^{glucose_i} \setminus \square_{glucose_o} \cdot P_{glucose}}}
\end{aligned}$$

Table 3: The process for *glucose*.

$$\begin{aligned}
P_A &\triangleq \hat{A}_i \setminus \square_{\hat{A}_o} \setminus \square_{A_o} \cdot P_A + A_i \setminus A_o \cdot \overline{P_A} & P_y &\triangleq \tilde{y}_i \setminus \square_{\tilde{y}_o} \setminus \square_{y_o} \cdot P_y + y_i \setminus y_o \cdot \overline{P_y} \\
\overline{P_A} &\triangleq \hat{A}_i \setminus \square_{\hat{A}_o} \setminus \square_{A_o} \cdot \overline{P_A} + A_i \setminus A_o \cdot P_A & \overline{P_y} &\triangleq \tilde{y}_i \setminus \square_{\tilde{y}_o} \setminus \square_{y_o} \cdot \overline{P_y} + y_i \setminus y_o \cdot P_y
\end{aligned}$$

Table 4: The processes for *y* and *A*.