SOS Rules for Equivalences of Reaction Systems*

Linda Brodo[1][0000−0002−4455−2419], Roberto Bruni[2][0000−0002−7771−4154], and Moreno Falaschi[3][0000−0002−6659−3828]

[1] Dipartimento di Scienze Economiche e Aziendali, Università di Sassari, Italy
[2] Dipartimento di Informatica, Università di Pisa, Italy
[3] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Italy

Abstract. Reaction Systems (RSs) are a successful computational framework inspired by biological systems. A RS combines a set of entities with a set of reactions over them. Entities can be provided by an external context, used to enable or inhibit each reaction, and also produced by reactions. RS semantics is defined in terms of an (unlabelled) rewrite system: given the current set of entities, a rewrite step consists of the application of all and only the enabled reactions. In this paper we define, for the first time, a compositional labelled transition system for RSs with recursive and nondeterministic contexts, in the structural operational semantics (SOS) style. This is achieved by distilling a signature whose operators directly correspond to the ingredients of RSs and by defining some simple SOS inference rules for any such operator. The rich information recorded in the labels allows us to define an assertion language to tailor behavioural equivalences on some specific properties or entities. The SOS approach is suited to drive additional enhancements of RSs along features such as quantitative measurements of entities and communication between RSs. The SOS rules have been also exploited to design a prototype implementation in logic programming.

Keywords: SOS rules, Reaction Systems, assertions, logic programming

1 Introduction

Labelled Transition Systems (LTSs) are a powerful structure to model the behaviour of interacting processes. An LTS can be conveniently defined following the Structural Operational Semantics (SOS) approach [25,27]. Given a signature, an SOS system assigns some inference rules to each operator of the language: the conclusion of each rule is the transition of a composite term, which is determined by those of its constituents (appearing as premises of the rule). The SOS approach has been particularly successful in the area of process algebras [22,26,20].

Reaction Systems (RSs) [8] are a computational framework inspired by systems of living cells. Its constituents are a finite set of entities and a finite set of

* Research supported by MIUR PRIN 201784YSZ5 *ASPRA*, by U. of Pisa PRA_2018_66 *DECLWARE*, by U. of Sassari: Fondo di Ateneo per la ricerca 2020.

reactions acting on entities. A reaction is a triple (R, I, P) where R is the set of reactants (entities whose presences is needed to enable the reaction), I is the set of inhibitors (entities whose absence is needed to enable the reaction) and P is the set of products (entities that are produced if the reaction takes place and that will be made available at the next step). After their introduction, RSs have shown to be a quite general computational model whose application ranges from the modelling of biological phenomena [2,14,1,3], and molecular chemistry [24] to theoretical foundations of computing. The semantics of RSs is defined as an unlabelled rewrite system whose states are set of entities (coming from an external context or produced at the previous step). Given the current set of entities, a rewrite step consists of the application of all and only the enabled reactions. Given a context sequence, the behaviour of an RS is uniquely determined and the corresponding (unlabelled, deterministic) transition system is finite.

Here we will define, for the first time, an LTS semantics in the SOS style for RSs with enhanced contexts. First we fix a process signature whose operators pinpoint the basic structure of a RS. We have operators for entities and reactions. For contexts we exploit some classic process algebraic operators (action prefix, sum and recursion). This way we can recursively define contexts that possibly exhibit nondeterministic behaviour, as sometimes have already appeared in the literature [21,9]. Even though we enrich the expressiveness of contexts, the overall LTS still remains finite. The SOS approach has several advantages: 1) compositionality, the behaviour of each composite system is defined in term of the behaviours of its constituents; 2) each transition label conveys all the activities connected to that computational step; 3) the definition of contexts is better integrated in the framework; 4) different context sequences can be studied at once; 5) it is now easier to extend the concept of RSs by adding new operators; 6) SOS rules facilitate implementation in a declarative language and the application of standard techniques for defining equivalences between processes.

The transition labels of our LTS are so rich of information that standard notion of behavioural equivalence (like traces or bisimulation) are too fine grain. For studying RSs, one is often interested in focusing on some entities and disregard others, like exploiting a microscope to enhance certain details and ignore others that fall out of the picture. To this aim, following the ideas in our previous paper [10], we propose an assertion language built over the transition labels, and we make the definition of behavioural and logical equivalences parametric w.r.t. such assertions. This way, it is possible to consider different RSs as equivalent for some purposes or to distinguish them for other purposes. Furthermore, we develop suitable behavioural equivalences for RS processes, and show the correspondence between a coinductive definition in terms of bisimilarity and its logical counterpart *à la* Hennessy-Milner.

We have developed a prototype implementation in logic programming of our semantic framework available online, as we describe in Section 5. Our interpreter allows the user to check automatically on the labels for a given RS the validity of formulas expressed in our variant of the Hennessy-Milner logic combined with the assertions specified in our language.

Related work. The work by Kleijn et al. [21] studies two versions of LTS for RS: one with state-oblivious context controller, and the other with state-aware context controller. In the first version, the labels only record the entities provided by the context, and in the second one the labels also provide the entities composing the actual state. The last choice allows one to decide which entities the context should provide. Differently, we give a process algebra-style definition of the RS, where the SOS rules produce informative transition labels, including context specification, allowing different kinds of analysis.

There are some previous works based on bisimulation applied to models for biological systems. Barbuti et al. [4] 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. [12] 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 mentioned formalisms [4,12] adopt a classical approach to bisimulation.

In Brodo et al. [9,10] we derived similar results to those presented here by encoding RSs into cCNA, a multi-party process algebra (a variant of the `link`-calculus [5,6]). In comparison with the encoding of RS in cCNA, we get here a much simpler computational model, closer to the syntax of RSs, preserving the expressiveness at the level of transition labels.

Structure of the paper. Section 2 recalls the basics of RSs. The original contribution starts from Section 3, where: 1) we introduce the syntax and SOS rules of a novel process algebra for RSs, 2) we show how to encode RSs as processes, 3) we prove a tight semantic correspondence between RSs and their encodings. Section 4 shows the correspondence between assertion-based coinductive equivalences and their logical counterpart *à la* Hennessy-Milner. A prototype implementation in logic programming of our semantic framework is briefly described in Section 5. Further extensions of RSs that build on our theory are sketched in Section 6. Some concluding remarks are in Section 7.

2 Reaction Systems

The theory of Reaction Systems (RSs) [8] was born in the field of Natural Computing to model the behaviour of biochemical reactions in living cells.

We use the term *entities* to denote generic molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system. The main mechanisms that regulate the functioning of a living cell are *facilitation* and *inhibition*. These mechanisms are based on the presence and absence of entities and are reflected in the basic definitions of RSs.

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

The sets R, I, P are the sets of *reactants*, *inhibitors*, and *products*, respectively. All reactants are needed for the reaction to take place. Any inhibitor blocks the reaction. Products are the outcome of the reaction. Since R and I are not empty, all products are produced from at least one reactant and every reaction can be inhibited. We let $rac(S)$ be the set of all reactions in S.

Definition 2 (Reaction System). *A Reaction System (RS) is a pair $\mathcal{A} = (S, A)$ s.t. S is a finite set, and $A \subseteq rac(S)$ is a finite set of reactions in S.*

The theory of RSs is based on three assumptions: **no permanency**, any entity vanishes unless it is sustained by a reaction. In fact, 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**, we assume that either an entity is available for all reactions, or it is not available at all.

Definition 3 (Reaction Result). *Given a (finite) set of entities S, and a subset $W \subseteq S$, we define the following:*

1. *Let $a = (R, I, P) \in rac(S)$ be a reaction in S. The result of a on W, denoted by $res_a(W)$, is defined by:*

$$res_a(W) \triangleq \begin{cases} P & \text{if } en_a(W) \\ \emptyset & \text{otherwise} \end{cases}$$

   where the enabling predicate is defined by $en_a(W) \triangleq R \subseteq W \wedge I \cap W = \emptyset$.
2. *Let $A \subseteq rac(S)$ be a finite set of reactions. The result of A on W, denoted by $res_A(W)$, is defined by: $res_A(W) \triangleq \bigcup_{a \in A} res_a(W)$.*

Living cells are seen as open systems that react with the external environment. The 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 $n \geq 0$. An n-steps* interactive process *in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$ s.t. $\gamma = \{C_i\}_{i \in [0,n]}$ is the* context sequence *and $\delta = \{D_i\}_{i \in [0,n]}$ is the* result sequence*, where $C_i, D_i \subseteq S$ for any $i \in [0, n]$, $D_0 = \emptyset$, and $D_{i+1} = res_A(D_i \cup C_i)$ for any $i \in [0, n-1]$. We call $\tau = W_0, \ldots, W_n$ with $W_i \triangleq C_i \cup D_i$, for any $i \in [0, n]$ the* state sequence*.*

The context sequence γ represents the environment. The result sequence δ is entirely determined by γ and A. Each state W_i in τ is the union of two sets: the context C_i at step i and the result set $D_i = res_A(W_{i-1})$ from the previous step.

Given a context sequence γ, we denote by γ^k the shift of γ starting at the k-th step. The shift notation will come in handy to draw a tight correspondence between the classic semantics of RS and the newly proposed SOS specification.

Definition 5 (Sequence shift). *Let $\gamma = \{C_i\}_{i \in [0,n]}$ a context sequence. Given a positive integer $k \leq n$ we let $\gamma^k = \{C_{i+k}\}_{i \in [0, n-k]}$. Note that $\gamma^0 = \gamma$.*

We conclude this section with a simple example of RS.

Example 1. Here we consider a toy RS defined as $\mathcal{A} = (S, A)$ where the set $S = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ only contains three entities, and the set of reactions $A = \{a_1\}$ only contains the reaction $a_1 = (\{\mathsf{a}, \mathsf{b}\}, \{\mathsf{c}\}, \{\mathsf{b}\})$, to be written more concisely as $(\mathsf{ab}, \mathsf{c}, \mathsf{b})$. Then, we consider a $4-$*steps* interactive process $\pi = (\gamma, \delta)$, where $\gamma = \{C_0, C_1, C_2, C_3\}$, with $C_0 = \{\mathsf{a}, \mathsf{b}\}$, $C_1 = \{\mathsf{a}\}$, $C_2 = \{\mathsf{c}\}$, and $C_3 = \{\mathsf{c}\}$; and $\delta = \{D_0, D_1, D_2, D_3\}$, with $D_0 = \emptyset$, $D_1 = \{\mathsf{b}\}$, $D_2 = \{\mathsf{b}\}$, and $D_3 = \emptyset$. Then, the resulting state sequence is

$$\tau = W_0, W_1, W_2, W_3 = \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{b}, \mathsf{c}\}, \{\mathsf{c}\}.$$

In fact, it is easy to check that, e.g., $W_0 = C_0$, $D_1 = res_A(W_0) = res_A(\{\mathsf{a}, \mathsf{b}\}) = \{\mathsf{b}\}$ because $en_a(W_0)$, and $W_1 = C_1 \cup D_1 = \{\mathsf{a}\} \cup \{\mathsf{b}\} = \{\mathsf{a}, \mathsf{b}\}$.

3 SOS Rules for Reaction Systems

Inspired by classic process algebras, such as CCS [22], we introduce a syntax for RSs that resembles their original presentation and then equip each operator with some SOS inference rules that define its behaviour. This way: (1) we establish a strong correspondence between terms of the signature and RSs; (2) we derive an LTS semantics for each RS, where the states are terms, each transition corresponds to a step of the RS and transition labels retain some information needed for compositionality; (3) we pave the way to the RS enhancements in Section 6.

Definition 6 (RS processes). *Let S be a set of entities. An* RS process P *is any term defined by the following grammar:*

$$\begin{array}{l} \mathsf{P} ::= [\mathsf{M}] \\ \mathsf{M} ::= (R, I, P) \mid D \mid \mathsf{K} \mid \mathsf{M}|\mathsf{M} \\ \mathsf{K} ::= \mathbf{0} \mid X \mid C.\mathsf{K} \mid \mathsf{K} + \mathsf{K} \mid \mathsf{rec}\ X.\ \mathsf{K} \end{array}$$

where $R, I, P \subseteq S$ are non empty sets of entities (with $R \cap I = \emptyset$), $C, D \subseteq S$ are possibly empty set of entitities, and X is a process variable.

An RS process P embeds a *mixture* process M obtained as the parallel composition of some reactions (R, I, P), some set of currently present entities D (possibly the empty set $\emptyset$), and some *context* process K. We write $\prod_{i \in I} \mathsf{M}_i$ for the parallel composition of all M_i with $i \in I$. For example, $\prod_{i \in \{1,2\}} \mathsf{M}_i = \mathsf{M}_1 \mid \mathsf{M}_2$.

A process context K is a possibly nondeterministic and recursive system: the nil context $\mathbf{0}$ stops the computation; the prefixed context $C.\mathsf{K}$ says that the entities in C are immediately available to be consumed by the reactions, and then K is the context offered at the next step; the non deterministic choice $\mathsf{K}_1 + \mathsf{K}_2$ allows the context to behave either as K_1 or K_2; X is a process variable, and $\mathsf{rec}\ X.\ \mathsf{K}$ is the usual recursive operator of process algebras. We write $\sum_{i \in I} \mathsf{K}_i$ for the nondeterministic choice between all K_i with $i \in I$.

We say that P and P' are structurally equivalent, written $\mathsf{P} \equiv \mathsf{P}'$, when they denote the same term up to the laws of commutative monoids (unit, associativity and commutativity) for parallel composition $\cdot|\cdot$, with $\emptyset$ as the unit, and the laws of idempotent and commutative monoids for choice $\cdot + \cdot$, with $\mathbf{0}$ as the unit. We also assume $D_1|D_2 \equiv D_1 \cup D_2$ for any $D_1, D_2 \subseteq S$.

Remark 1. Note that the (mixture) process $\emptyset$ and the (context) process $\mathbf{0}$ are not the same: as it will become clear from the operational semantics, the process $\emptyset$ has just a trivial transition to itself, while the process $\mathbf{0}$ has no outgoing transition and is used to stop the computation.

Definition 7 (RSs as RS processes). *Let $\mathcal{A} = (S, A)$ be a RS, and $\pi = (\gamma, \delta)$ an n-step interactive process in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$. For any step $i \in [0, n]$, the corresponding RS process $[\![\mathcal{A}, \pi]\!]_i$ is defined as follows:*

$$[\![\mathcal{A}, \pi]\!]_i \triangleq \left[\prod_{a\in A} a \mid D_i \mid \mathsf{K}_{\gamma^i}\right]$$

where the context process $\mathsf{K}_{\gamma^i} \triangleq C_i.C_{i+1}.\cdots.C_n.\mathbf{0}$ is the sequentialization of the entities offered by γ^i. We write $[\![\mathcal{A}, \pi]\!]$ as a shorthand for $[\![\mathcal{A}, \pi]\!]_0$.

Example 2. Here, we give the encoding of the reaction system, $\mathcal{A} = (S, A)$, defined in Example 1. The resulting RS process is as follows:

$$\mathsf{P} = [\![\mathcal{A}, \pi]\!] = [\![(\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, \{(\mathsf{ab}, \mathsf{c}, \mathsf{b})\}), \pi]\!] = [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \emptyset \mid \mathsf{K}_\gamma] \equiv [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \mathsf{K}_\gamma]$$

where $\mathsf{K}_\gamma = \{\mathsf{a}, \mathsf{b}\}.\{\mathsf{a}\}.\{\mathsf{c}\}.\{\mathsf{c}\}.\mathbf{0}$, written more concisely as $\mathsf{ab.a.c.c.}\mathbf{0}$. Note that $D_0 = \emptyset$ is inessential and can be discarded thanks to structural congruence.

In Definition 7 we have not exploited the entire potentialities of the syntax. In particular, the context K_γ is just a finite sequence of action prefixes induced by the set of entities provided by γ at the various steps. Our syntax allows for more general kinds of contexts as shown in the example below. Nondeterministic contexts can be used to collect several experiments, while recursion can be exploited to extract some regularity in the longterm behaviour of a Reaction System. Together they offer any combination of in-breadth/in-depth analysis.

Example 3. Let us consider our running example. Suppose we want to enhance the behaviour of the context by defining a process $\mathsf{K}' = \mathsf{K}_1 + \mathsf{K}_2$ that nondeterministically can behave as K_1 or as K_2, where $\mathsf{K}_1 = \mathsf{ab.a.c.c.}\mathbf{0}$ (as in Example 2), and $\mathsf{K}_2 = \mathsf{rec}\ X.\ \mathsf{ab.a}.X$ (which is a recursive behaviour that allows the reaction to be always enabled). Then we simply define $\mathsf{P}' \equiv [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \mathsf{K}']$.

Definition 8 (Label). *A label is a tuple $\langle W \rhd R, I, P\rangle$ with $W, R, I, P \subseteq S$.*

In a transition label $\langle W \rhd R, I, P\rangle$, we record the set W of entities currently in the system (produced in the previous step or provided by the context), the

$$\frac{}{D \xrightarrow{\langle D \triangleright \emptyset,\emptyset,\emptyset\rangle} \emptyset}\ (Ent) \qquad \frac{}{C.\mathsf{K} \xrightarrow{\langle C \triangleright \emptyset,\emptyset,\emptyset\rangle} \mathsf{K}}\ (Cxt) \qquad \frac{\mathsf{K}[^{\mathsf{rec}\ X.\ \mathsf{K}}/_X] \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{K}'}{\mathsf{rec}\ X.\ \mathsf{K} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{K}'}\ (Rec)$$

$$\frac{\mathsf{K}_1 \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{K}_1'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{K}_1'}\ (Suml) \qquad \frac{\mathsf{K}_2 \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{K}_2'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{K}_2'}\ (Sumr)$$

$$\frac{}{(R,I,P) \xrightarrow{\langle \emptyset \triangleright R,I,P\rangle} (R,I,P) \mid P}\ (Pro) \qquad \frac{J \subseteq I \quad Q \subseteq R \quad J \cup Q \neq \emptyset}{(R,I,P) \xrightarrow{\langle \emptyset \triangleright J,Q,\emptyset\rangle} (R,I,P)}\ (Inh)$$

$$\frac{\mathsf{M}_1 \xrightarrow{\langle W_1 \triangleright R_1,I_1,P_1\rangle} \mathsf{M}_1' \quad \mathsf{M}_2 \xrightarrow{\langle W_2 \triangleright R_2,I_2,P_2\rangle} \mathsf{M}_2' \quad (W_1 \cup W_2 \cup R_1 \cup R_2) \cap (I_1 \cup I_2) = \emptyset}{\mathsf{M}_1 \mid \mathsf{M}_2 \xrightarrow{\langle W_1 \cup W_2 \triangleright R_1 \cup R_2,I_1 \cup I_2,P_1 \cup P_2\rangle} \mathsf{M}_1' \mid \mathsf{M}_2'}\ (Par)$$

$$\frac{\mathsf{M} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{M}' \quad R \subseteq W}{[\mathsf{M}] \xrightarrow{\langle W \triangleright R,I,P\rangle} [\mathsf{M}']}\ (Sys)$$

Fig. 1: SOS semantics of the reaction system processes.

set R of entities whose presence is assumed (either because they are needed as reactants on an applied reaction or because their presence prevents the application of some reaction); the set I of entities whose absence is assumed (either because they appear as inhibitors for an applied reaction or because their absence prevents the application of some reaction); the set P of products of all the applied reactions.

Definition 9 (Operational semantics). *The operational semantics of processes is defined by the set of SOS inference rules in Figure 1.*

The process $\mathbf{0}$ has no transition. The rule (*Ent*) makes available the entities in the (possibly empty) set D, then reduces to $\emptyset$. As a special instance of (*Ent*), $\emptyset \xrightarrow{\langle \emptyset \triangleright \emptyset,\emptyset,\emptyset\rangle} \emptyset$. The rule (*Cxt*) says that a prefixed context process $C.\mathsf{K}$ makes available the entities in the set C and then reduces to K. The rule (*Rec*) is the classical rule for recursion. Here, $\mathsf{K}[^{\mathsf{rec}\ X.\ \mathsf{K}}/_X]$ denotes the process obtained by replacing in K every free occurrence of the variable X with its recursive definition $\mathsf{rec}\ X.\ \mathsf{K}$. For example $\mathsf{rec}\ X.\ \mathsf{a.b}.X \xrightarrow{\langle \mathsf{a} \triangleright \emptyset,\emptyset,\emptyset\rangle} \mathsf{b.rec}\ X.\ \mathsf{a.b}.X$ The rules (*Suml*) and (*Sumr*) select a move of either the left or the right component, resp., discarding the other process. The rule (*Pro*), executes the reaction (R,I,P) (its reactants, inhibitors, and products are recorded the label), which remains available at the next step together with P. The rule (*Inh*) applies when the reaction (R,I,P) should not be executed; it records in the label the possible causes for which the reaction is disabled: possibly some inhibiting entities ($J \subseteq I$) are present or some reactants ($Q \subseteq R$) are missing, with $J \cup Q \neq \emptyset$, as at least one cause is needed for

explaining why the reaction is not enabled.[4] The rule (*Par*) puts two processes in parallel by pooling their labels and joining all the set components of the labels; a sanity check is required to guarantee that there is no conflict between reactants and inhibitors of the applied reactions. Finally, the rule (*Sys*) requires that all the processes of the systems have been considered, and also checks that all the needed reactants are actually available in the system ($R \subseteq W$). In fact this constraint can only be met on top of all processes. The check that inhibitors are absent ($I \cap W = \emptyset$) is not necessary, as is guaranteed by the semantics (see Lemma 3).

Example 4. Let us consider the RS process $\mathsf{P}_0 \triangleq [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{ab.a.c.c.}\mathbf{0}]$ from Example 2. The process P_0 has a unique outgoing transition, whose formal derivation is given below:

$$\dfrac{\dfrac{\dfrac{}{(\mathsf{ab},\mathsf{c},\mathsf{b}) \xrightarrow{\langle \emptyset \rhd \mathsf{ab},\mathsf{c},\mathsf{b}\rangle} (\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b}}\,(Pro) \quad \dfrac{}{\mathsf{ab.a.c.c.}\mathbf{0} \xrightarrow{\langle \mathsf{ab} \rhd \emptyset,\emptyset,\emptyset\rangle} \mathsf{a.c.c.}\mathbf{0}}\,(Cxt)}{(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{ab.a.c.c.}\mathbf{0} \xrightarrow{\langle \mathsf{ab} \rhd \mathsf{ab},\mathsf{c},\mathsf{b}\rangle} (\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{a.c.c.}\mathbf{0}}\,(Par)}{[(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{ab.a.c.c.}\mathbf{0}] \xrightarrow{\langle \mathsf{ab} \rhd \mathsf{ab},\mathsf{c},\mathsf{b}\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{a.c.c.}\mathbf{0}]}\,(Sys)$$

The target process $\mathsf{P}_1 \triangleq [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{a.c.c.}\mathbf{0}]$ has also a unique outgoing transition, namely:

$$\mathsf{P}_1 = [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{a.c.c.}\mathbf{0}] \xrightarrow{\langle \mathsf{ab} \rhd \mathsf{ab},\mathsf{c},\mathsf{b}\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}] = \mathsf{P}_2$$

Instead the process P_2 has three outgoing transitions, each providing a different justification to the fact that the reaction $(\mathsf{ab},\mathsf{c},\mathsf{b})$ is not enabled:

1. $[(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}] \xrightarrow{\langle \mathsf{bc} \rhd \mathsf{c},\mathsf{a},\emptyset\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{c.}\mathbf{0}]$, where the label shows that the presence of c and the absence of a inhibit the reaction;
2. $[(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}] \xrightarrow{\langle \mathsf{bc} \rhd \mathsf{c},\emptyset,\emptyset\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{c.}\mathbf{0}]$, where it is only observed that the presence of c has played some role in inhibiting the reaction;
3. $[(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}] \xrightarrow{\langle \mathsf{bc} \rhd \emptyset,\mathsf{a},\emptyset\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{c.}\mathbf{0}]$, where it is only observed that the absence of a has played some role in inhibiting the reaction.

Notably, the three transitions have the same target process $\mathsf{P}_3 \triangleq [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{c.}\mathbf{0}]$.

Finally, the process P_3 has seven transitions all leading to $\mathsf{P}_4 \triangleq [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathbf{0}]$. Their labels are of the form $\langle c \rhd J, Q, \emptyset\rangle$ with $J \subseteq \mathsf{c}$, $Q \subseteq \mathsf{ab}$ and $J \cup Q \neq \emptyset$. Each label provides a different explanation why the reaction is not enabled.

The following technical lemmas express some relevant properties of the transition system and are proved by rule induction.

[4] Conceptually, one could extend labels to record J and Q in separate positions from R and I, respectively, like in $\langle W \rhd R, J, I, Q, P\rangle$. However, one would then need to rewrite the side conditions of all the rules by replacing R with $R \cup J$ and I with $I \cup Q$, because the distinction is never exploited in the SOS rules.

Lemma 1. *If* $\mathsf{M} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{M}'$ *then* $\mathsf{M}' \equiv \mathsf{M}''|P$ *for some* M''.

Lemma 2. *If* $\prod_{a\in A} a \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{M}$ *then* $W = \emptyset$ *and* $\mathsf{M} \equiv \prod_{a\in A} a \mid P$.

Lemma 3. *If* $\mathsf{M} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{M}'$ *then* $(W \cup R) \cap I = \emptyset$.

Lemma 4. *If* $\mathsf{P} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{P}'$ *then* $R \subseteq W$ *and* $W \cap I = \emptyset$.

The main theorem shows that the rewrite steps of a RS exactly match the transitions of its corresponding RS process.

Theorem 1. *Let* $\mathcal{A} = (S, A)$ *be a RS, and* $\pi = (\gamma, \delta)$ *an* n*-step interactive process in* $\mathcal{A}$ *with* $\gamma = \{C_i\}_{i\in[0,n]}$, $\delta = \{D_i\}_{i\in[0,n]}$, *and let* $W_i \triangleq C_i \cup D_i$ *and* $\mathsf{P}_i \triangleq [\![\mathcal{A}, \pi]\!]_i$ *for any* $i \in [0, n]$. *Then:*

1. $\forall i \in [0, n-1]$, $\mathsf{P}_i \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{P}$ *implies* $W = W_i$, $P = D_{i+1}$ *and* $\mathsf{P} \equiv \mathsf{P}_{i+1}$;
2. $\forall i \in [0, n-1]$, *there exists* $R, I \subseteq S$ *such that* $\mathsf{P}_i \xrightarrow{\langle W_i \triangleright R,I,D_{i+1}\rangle} \mathsf{P}_{i+1}$.

Remark 2. Note that the process $\mathsf{P}_n = [\![\mathcal{A}, \pi]\!]_n = [\prod_{a\in A} a \mid D_n \mid C_n.\mathbf{0}]$ has one more transition available (the $(n+1)$-th step from P_0), even if the standard theory of RSs stops the computation after n steps. We thus have additional steps

$$\mathsf{P}_n \xrightarrow{\langle W_n \triangleright R_n, I_n, res_A(W_n)\rangle} \left[\prod_{a\in A} a \mid res_A(W_n) \mid \mathbf{0}\right]$$

for suitable $R_n, I_n \subseteq S$. The target process contains $\mathbf{0}$ and therefore is deadlock.

Example 4 shows that we can have redundant transitions because of rule (*Inh*). However, they can be easily detected and eliminated by considering a notion of dominance. To this aim we introduce an order relation $\sqsubseteq$ over pairs of set of entities defined as follows:

$$(R', I') \sqsubseteq (R, I) \quad \text{if} \quad R' \subseteq R \wedge I' \subseteq I.$$

Definition 10 (Dominance). *A transition* $\mathsf{P} \xrightarrow{\langle W \triangleright R',I',P\rangle} \mathsf{P}'$ *is* dominated *if there exists another transition* $\mathsf{P} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{P}'$ *such that* $(R', I') \sqsubset (R, I)$.

Note that in the definition of dominance we require the dominated transition to have the same source and target processes as the dominant transition, and that their labels carry also the same sets W and P.

Finally, we can immediately derive an LTS, whose transitions are written using double arrows, where only dominant transitions are considered. The LTS is defined by the additional SOS rule (*Dom*) below:

$$\frac{\mathsf{P} \xrightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{P}' \quad (R, I) = \max_{\sqsubseteq}\{(R', I') \mid \mathsf{P} \xrightarrow{\langle W \triangleright R',I',P\rangle} \mathsf{P}'\}}{\mathsf{P} \xRightarrow{\langle W \triangleright R,I,P\rangle} \mathsf{P}'} \ (Dom)$$

In other words, a transition $\mathsf{P} \stackrel{\langle W \triangleright R,I,P\rangle}{\Longrightarrow} \mathsf{P}'$ guarantees that any instance of the rule (*Inh*) is applied in a way that maximizes the sets J and Q (given the overall available entities W).

Example 5. Looking back at Example 4, both transitions $\mathsf{P}_2 \xrightarrow{\langle \mathsf{bc} \triangleright \mathsf{c},\emptyset,\emptyset\rangle} \mathsf{P}_3$ and $\mathsf{P}_2 \xrightarrow{\langle \mathsf{bc} \triangleright \emptyset,\mathsf{a},\emptyset\rangle} \mathsf{P}_3$ are dominated by $\mathsf{P}_2 \xrightarrow{\langle \mathsf{bc} \triangleright \mathsf{c},\mathsf{a},\emptyset\rangle} \mathsf{P}_3$. Therefore, the process $\mathsf{P}_2 = [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}]$ has a unique (double-arrow) transition $\mathsf{P}_2 \stackrel{\langle \mathsf{bc} \triangleright \mathsf{c},\mathsf{a},\emptyset\rangle}{\Longrightarrow} \mathsf{P}_3$.

4 Bio-simulation

Bisimulation equivalences [28] play a central role in process algebras. They can be defined in terms of coinductive games, of fixpoint theory and of logics. The bisimulation game is played by an attacker and a defender: the former wants to disprove the equivalence between two processes p and q, the latter that p and q are equivalent. The game is turn based: at each turn the attacker picks one process, e.g., p, and one transition $p \xrightarrow{\lambda} p'$ and the defender must reply by picking one transition $q \xrightarrow{\lambda} q'$ of the other process with exactly the same label λ; then the game continues challenging the equivalence between p' and q'. The game ends when the attacker has no transition available, and the defender wins, or when defender cannot match the move of the attacker, and the attacker wins. The defender also wins if the game doesn't end. Then p and q are not equivalent iff the attacker has a winning strategy. There are many variants of the bisimulation for process algebras, for example the barbed bisimulation [23] 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 [18].

In the case of biological systems, the classical notion of bisimulation can be too concrete. In fact, in a biological soup, a high number of interactions occur every time instant, and generally, biologists are only interested to analyse a small subset of them and to focus on a subset of entities. In the case of RS processes, the labels that we used for the LTS consider too many details and convey too much information: they record the entire information about all the reactions that have been applied in one transition, the entities that acted as reactants, as inhibitors or as products, or that were available in the state. All this information stored in the label is necessary to compose a transition in a modular way. Depending on the application, only a suitable abstraction over the label can be of interest. For this reason, following the approach introduced in Brodo et al. [10], we propose an alternative notion of bisimulation, called *bio-simulation*, that compares two biological systems by restricting the observation to only a limited set of events that are of particular interest. With respect to the work in Brodo et al. [10], here the labels are easier to manage and simpler to parse.

In a way, at each step of the bisimulation game, we want to query our labels about some partial information. To this goal, we define an assertion language to express detailed and partial queries about what happened in a single transition.

Example 6. 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 presence of the entity a been exploited by some reaction?
2. Have the entities a and b been produced by some reaction?
3. Have the entities a or c been provided by the state?
4. Has the reaction $(\mathsf{ab}, \mathsf{c}, \mathsf{b})$ been applied or not?

As detailed before, in the following we assume that the context can be non-deterministic, otherwise it makes little sense to rely on bisimulation to observe the branching structure of system dynamics.

The bio-simulation approach works as follows: first we introduce an assertion language to abstract away some information from the labels; then we define a bisimilarity equivalence that is parametric to a given assertion, called bio-similarity; finally we give a logical characterisation of bio-similarity, called bio-logical equivalence, by tailoring the classical HML to the given assertion.

4.1 Assertion language

An assertion is a formula that predicates on the labels of our LTS. The assertion language that we propose is very basic, but can be extended if necessary.

Definition 11 (Assertion Language). *Given a set of entities S, assertions F on S are built from the following syntax, where $E \subseteq S$ and $Pos \in \{\mathcal{W}, \mathcal{R}, \mathcal{I}, \mathcal{P}\}$:*

$$\mathsf{F} ::= E \subseteq Pos \;\mid\; ? \in Pos \;\mid\; \mathsf{F} \vee \mathsf{F} \;\mid\; \mathsf{F} \wedge \mathsf{F} \;\mid\; \mathsf{F} \mathbin{\hat{}} \mathsf{F} \;\mid \neg \mathsf{F}$$

Roughly, Pos distinguishes different positions in the labels: $\mathcal{W}$ stands for entities provided by current state, $\mathcal{R}$ stands for reactants, $\mathcal{I}$ stands for inhibitors, and $\mathcal{P}$ stands for products. An assertion F is either the membership of a subset of entities E in a given position Pos, $E \subseteq Pos$, the test of Pos for non-emptyness, $? \in Pos$, the disjunction of two assertions $\mathsf{F}_1 \vee \mathsf{F}_2$, their conjunction $\mathsf{F}_1 \wedge \mathsf{F}_2$, their exclusive or $\mathsf{F}_1 \mathbin{\hat{}} \mathsf{F}_2$, or the negation of an assertion $\neg\mathsf{F}$.

Definition 12 (Satisfaction of Assertion). *Let $\upsilon = \langle W \rhd R, I, P \rangle$ 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 and only if the following hold:*

$$\begin{array}{ll}
\upsilon \models E \subseteq Pos & \textit{iff}\ \ E \subseteq \mathsf{select}(\upsilon, Pos) \\
\upsilon \models ? \in Pos & \textit{iff}\ \ \mathsf{select}(\upsilon, Pos) \neq \emptyset \\
\upsilon \models \mathsf{F}_1 \wedge \mathsf{F}_2 & \textit{iff}\ \ \upsilon \models \mathsf{F}_1 \wedge \upsilon \models \mathsf{F}_2 \\
\upsilon \models \mathsf{F}_1 \vee \mathsf{F}_2 & \textit{iff}\ \ \upsilon \models \mathsf{F}_1 \vee \upsilon \models \mathsf{F}_2 \\
\upsilon \models \mathsf{F}_1 \mathbin{\hat{}} \mathsf{F}_2 & \textit{iff}\ \ (\upsilon \models \mathsf{F}_1 \wedge \upsilon \models \neg\mathsf{F}_2) \vee (\upsilon \models \neg\mathsf{F}_1 \wedge \upsilon \models \mathsf{F}_2) \\
\upsilon \models \neg\mathsf{F} & \textit{iff}\ \ \upsilon \not\models \mathsf{F}
\end{array}$$

$$\textit{where} \qquad \mathsf{select}(\langle W \rhd R, I, P\rangle, Pos) \triangleq \begin{cases} W & \textit{if } Pos = \mathcal{W} \\ R & \textit{if } Pos = \mathcal{R} \\ I & \textit{if } Pos = \mathcal{I} \\ P & \textit{if } Pos = \mathcal{P} \end{cases}$$

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 both do not.

Example 7. Some assertions matching the queries listed in Example 6 are:

1. $\mathsf{F}_1 \triangleq \mathsf{a} \subseteq \mathcal{R}$
2. $\mathsf{F}_2 \triangleq \mathsf{ab} \subseteq \mathcal{P}$
3. $\mathsf{F}_3 \triangleq \mathsf{a} \subseteq \mathcal{W} \vee \mathsf{c} \subseteq \mathcal{W}$
4. $\mathsf{F}_4 \triangleq \mathsf{ab} \subseteq \mathcal{R} \wedge \mathsf{c} \subseteq \mathcal{I}$ checks if the reaction has been applied, while $\mathsf{F}_5 \triangleq \mathsf{a} \subseteq \mathcal{I} \vee \mathsf{b} \subseteq \mathcal{I} \vee \mathsf{c} \subseteq \mathcal{R}$ the opposite case. Alternatively, we can set $\mathsf{F}_5 \triangleq \neg\mathsf{F}_4$.

If we take the label $\upsilon = \langle \mathsf{ab} \rhd \mathsf{ab}, \mathsf{c}, \mathsf{b}\rangle$ it is immediate to check that

$$\upsilon \models \mathsf{F}_1 \qquad \upsilon \not\models \mathsf{F}_2 \qquad \upsilon \models \mathsf{F}_3 \qquad \upsilon \models \mathsf{F}_4 \qquad \upsilon \not\models \mathsf{F}_5$$

With respect to the assertion language proposed in our previous paper [10], the one in Definition 11 applies to a much simpler LTS, designed for Reaction Systems, while the previous one had to consider the more general kinds of labels of the multi-party process algebra cCNA, which include, e.g., the name of each reaction together with the reagents and inhibitors that provide the reason why a reaction has been applied or not. Here, reactions are anonymous, but their enabling can still be inferred from transitions labels. The main interest of this proposal is that it is directly applied to the LTS tailored for RSs.

4.2 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 reaction 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 reaction rules when different contexts are provided.

Definition 13 (Bio-similarity $\sim_{\mathsf{F}}$ [10]). *Given an assertion* F*, a* bio-simulation $\mathbf{R}_{\mathsf{F}}$ *that respects* F *is a binary relation over RS processes s.t., if* $\mathsf{P}\ \mathbf{R}_{\mathsf{F}}\ \mathsf{Q}$ *then:*

- $\forall \upsilon, \mathsf{P}'$ *s.t.* $\mathsf{P} \stackrel{\upsilon}{\Rightarrow} \mathsf{P}'$*, then* $\exists w, \mathsf{Q}'$ *s.t.* $\mathsf{Q} \stackrel{w}{\Rightarrow} \mathsf{Q}'$ *with* $\upsilon \equiv_{\mathsf{F}} w$ *and* $\mathsf{P}'\ \mathbf{R}_{\mathsf{F}}\ \mathsf{Q}'$*.*
- $\forall w, \mathsf{Q}'$ *s.t.* $\mathsf{Q} \stackrel{w}{\Rightarrow} \mathsf{Q}'$*, then* $\exists \upsilon, \mathsf{P}'$ *s.t.* $\mathsf{P} \stackrel{\upsilon}{\Rightarrow} \mathsf{P}'$ *with* $\upsilon \equiv_{\mathsf{F}} w$ *and* $\mathsf{P}'\ \mathbf{R}_{\mathsf{F}}\ \mathsf{Q}'$*.*

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* $\mathsf{P} \sim_{\mathsf{F}} \mathsf{Q}$*.*

Remark 3. 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 $\mathsf{P} \stackrel{\upsilon}{\Rightarrow} \mathsf{P}'$ such that $\upsilon \models \mathsf{F}$ into $\mathsf{P} \stackrel{\mathsf{F}}{\Rightarrow} \mathsf{P}'$ and each transition $\mathsf{P} \stackrel{\upsilon}{\Rightarrow} \mathsf{P}'$ such that $\upsilon \not\models \mathsf{F}$ into $\mathsf{P} \stackrel{\neg\mathsf{F}}{\Longrightarrow} \mathsf{P}'$.

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.

Example 8. Let us consider some variants of our working example. The behavior of $\mathsf{P}_0 \triangleq [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \mathsf{ab.a.ac.}\mathbf{0}]$ is deterministic, and its unique trace of labels is:

$$\mathsf{P}_0 \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} \mathsf{P}_1 \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} \mathsf{P}_2 \stackrel{\langle \mathsf{abc} \triangleright \mathsf{c},\emptyset,\emptyset\rangle}{\Longrightarrow} [(\mathsf{ab}, \mathsf{c}, \mathsf{b})|\mathbf{0}]$$

Instead, the behavior of $\mathsf{P}'_0 \triangleq [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid (\mathsf{ab.a.ac.}\mathbf{0} + \mathsf{ab.a.a.}\mathbf{0})]$ is non deterministic. Now there are two possible traces of labels: the first trace is equal to the above one, and the other one follows:

$$\begin{array}{l} \mathsf{P}'_0 \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} \mathsf{P}_1 \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} \mathsf{P}_2 \stackrel{\langle \mathsf{abc} \triangleright \mathsf{c},\emptyset,\emptyset\rangle}{\Longrightarrow} [(\mathsf{ab}, \mathsf{c}, \mathsf{b})|\mathbf{0}] \\ \quad\quad \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} \mathsf{P}'_1 \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} \mathsf{P}'_2 \stackrel{\langle \mathsf{ab} \triangleright \mathsf{ab},\mathsf{c},\mathsf{b}\rangle}{\Longrightarrow} [(\mathsf{ab}, \mathsf{c}, \mathsf{b})|\mathsf{b}|\mathbf{0}] \end{array}$$

Now, it is easy to check that the two processes $\mathsf{P}_0, \mathsf{P}'_0$ are not bio-similar w.r.t. the assertion $\mathsf{F}_1 \triangleq \mathsf{c} \in \mathcal{E}$, requiring that in the state configuration entity c is present, and are bio-similar w.r.t. the assertion $\mathsf{F}_2 \triangleq (\mathsf{a} \in \mathcal{R}) \frown (\mathsf{c} \in \mathcal{R})$, requiring that either c or a are used as reactants. Hence, we write $\mathsf{P}_0 \not\sim_{\mathsf{F}_1} \mathsf{P}'_0$ and $\mathsf{P}_0 \sim_{\mathsf{F}_2} \mathsf{P}'_0$.

Now, we introduce a slightly modified version of the Hennessy-Milner Logic [19], 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 to the assertion F of interest:

Definition 14 (BioHML [10]). *Let F be an assertion, then the set of bioHML formulas G that respects F are built by the following syntax, where $\chi \in \{\mathsf{F}, \neg\mathsf{F}\}$:*

$$\mathsf{G}, \mathsf{H} ::= \mathtt{t} \mid \mathtt{f} \mid \mathsf{G} \wedge \mathsf{G} \mid \mathsf{G} \vee \mathsf{G} \mid \langle\chi\rangle\mathsf{G} \mid [\chi]\mathsf{G}$$

Remark 4. An alternative way to look at bioHML formulas is as ordinary HML formulas over the set of labels $\{\mathsf{F}, \neg\mathsf{F}\}$.

The semantics of a bioHML formula is the set of processes that satisfy it.

Definition 15 (Semantics of BioHML). *Let $\mathbb{P}$ denote the set of all RS processes over S. For a BioHML formula G, we define $[\![\mathsf{G}]\!] \subseteq \mathbb{P}$ inductively on G:*

$$[\![\mathtt{t}]\!] \triangleq \mathbb{P} \qquad [\![\mathtt{f}]\!] \triangleq \emptyset \qquad [\![\mathsf{G} \wedge \mathsf{H}]\!] \triangleq [\![\mathsf{G}]\!] \cap [\![\mathsf{H}]\!] \qquad [\![\mathsf{G} \vee \mathsf{H}]\!] \triangleq [\![\mathsf{G}]\!] \cup [\![\mathsf{H}]\!]$$

$$\begin{array}{l} [\![\langle\chi\rangle\mathsf{G}]\!] \triangleq \{\mathsf{P} \in \mathbb{P} : \exists \upsilon, \mathsf{P}'.\ \mathsf{P} \stackrel{\upsilon}{\Rightarrow} \mathsf{P}' \textit{ with } \upsilon \models \chi \textit{ and } \mathsf{P}' \in [\![\mathsf{G}]\!]\} \\ [\![[\chi]\mathsf{G}]\!] \triangleq \{\mathsf{P} \in \mathbb{P} : \forall \upsilon, \mathsf{P}'.\ \mathsf{P} \stackrel{\upsilon}{\Rightarrow} \mathsf{P}' \textit{ implies } \upsilon \models \chi \textit{ and } \mathsf{P}' \in [\![\mathsf{G}]\!]\} \end{array}$$

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

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.

We let $\mathcal{L}_{\mathsf{F}}$ be the set of all bioHML formulas that respects F.

Definition 16 (Bio-logical equivalence). *We say that* P, Q *are* bio-logically *equivalent w.r.t.* F*, written* $\mathsf{P} \equiv_{\mathcal{L}_{\mathsf{F}}} \mathsf{Q}$*, when* P *and* Q *satisfy the exactly the same bioHML formulas in* $\mathcal{L}_{\mathsf{F}}$*, i.e. when for any* $\mathsf{G} \in \mathcal{L}_{\mathsf{F}}$ *we have* $\mathsf{P} \models \mathsf{G} \ \Leftrightarrow \mathsf{Q} \models \mathsf{G}$.

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 2 (Correspondence [10]). $\sim_{\mathsf{F}} = \equiv_{\mathcal{L}_{\mathsf{F}}}$

Example 9. We continue by considering our running example in Example 8. There already is the evidence that the two processes $\mathsf{P}_0 \triangleq [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \mathsf{ab.a.ac.}\mathbf{0}]$ and $\mathsf{P}'_0 \triangleq [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid (\mathsf{ab.a.ac.}\mathbf{0} + \mathsf{ab.a.a.}\mathbf{0})]$ are not bio-similar w.r.t. the assertion $\mathsf{F}_1 \triangleq \mathsf{c} \in \mathcal{W}$. Here, we give a bioHML formula that distinguishes P_0 and P'_0:

$$\mathsf{G} \triangleq \langle \neg \mathsf{F}_1 \rangle [\neg \mathsf{F}_1] \langle \neg \mathsf{F}_1 \rangle \mathtt{t}.$$

In fact, G is not satisfied by P_0, written $\mathsf{P}_0 \not\models \mathsf{G}$, because, along the unique possible path, the labels of the first two transitions satisfy $\neg \mathsf{F}_1$ but P_2 cannot perform any transition whose label satisfies $\neg \mathsf{F}_1$.

Differently, $\mathsf{P}'_0 \models \mathsf{G}$. In fact, P'_0 can move to P'_1 with a transition whose label satisfies $\neg \mathsf{F}_1$, then P'_1 has a unique transition to P'_2 whose label satisfies $\neg \mathsf{F}_1$ and finally the target state P'_2 can perform a transition whose label satisfies $\neg \mathsf{F}_1$.

5 Implementation

In Falaschi and Palma [17] we have presented some preliminary work on how to implement RS formalism in a logic programming language (Prolog). Our implementation did not aim to be highly performing. We aimed to obtain a rapid prototyping tool for implementing extensions of Reaction Systems. Our initial prototype allowed to perform finite computations on RSs, in the form of interactive processes. Here we have extended the implementation by including the more general notion of contexts, the labels and keeping track of them building corresponding LTSs. Then we have added the predicates for formulating expressions of our assertion language that acts on the transition labels. On the basis of this assertion language we have implemented a slightly modified version of the Hennessy-Milner logic to make it parametric on the specific assertion specified by the user. Our interpreter is available for download[5].

For performance reasons and in conformance with the double-arrow transition system, our implementation uses the (*InH*) rule in a *deterministic* way by maximising the sets of present inhibitors and lacking reagents in the current

[5] `https://www3.diism.unisi.it/~falaschi/AssertionsForReactionSystems`

computation. This improves the efficiency of the tool. We have run and checked the examples in this paper, by using our interpreter. As explained in the online instructions, the tool can be customised by instantiating a few predicates providing the Reaction System specification and a BioHML formula to be verified.

6 Two extensions

Here we present two extensions: a numeric extension that takes into account the number of times an entity is used as a reactant in a single transition; an extension that introduces an operator for letting two RSs be *connected*.

Reactant occurrences.
The first idea is to introduce some naive measure for the number of entities that are needed by the reactions. Now, we assume that the number associated to entities in the sets R (reactants) and P (products) are the stoichiometric numbers, as specified in the corresponding biochemical equation. This amounts to use multisets instead of sets (for R and P) within the labels. The set I (of inhibitors) remains a simple set. At the level of notation, we write a multiset as a formal sum $\bigoplus_{\mathsf{a}\in S} n_{\mathsf{a}}\mathsf{a}$, where $n_{\mathsf{a}} \in \mathbb{N}$ is the number of occurrences of a. For simplicity, we write just a instead of $1\mathsf{a}$ and we omit any term of the form $0\mathsf{a}$. For example, the multiset $2\mathsf{a} \oplus \mathsf{b}$ has two instances of a and one of b. Overloading the notation we use $\cup$ as multiset union, i.e.

$$(\bigoplus_{\mathsf{a}\in S} n_{\mathsf{a}}\mathsf{a}) \cup (\bigoplus_{\mathsf{a}\in S} m_{\mathsf{a}}\mathsf{a}) = \bigoplus_{\mathsf{a}\in S}(n_{\mathsf{a}} + m_{\mathsf{a}})\mathsf{a}$$

If $R = \bigoplus_{\mathsf{a}\in S} n_{\mathsf{a}}\mathsf{a}$ we let $R(a) = n_{\mathsf{a}}$.

Similarly, we want to use multisets also for the contexts, but in this case we want the possibility to parameterize the context w.r.t. the number of entities it provides. To this purpose, fixed a finite set $X = \{x_1, ..., x_n\}$ of variables, we introduce some linear expressions of the form $e = \sum_{i=1}^{n} k_i x_i + h$ with coefficients $k_i, h \in \mathbb{N}$, such that a context C associates to each entity a a linear expression e_{a} and not just a number. Thus we write a context C as a formal sum $C = \bigoplus_{\mathsf{a}\in S} e_{\mathsf{a}}\mathsf{a}$. A multiset is just a particular case of the above expression where all variable coefficients are 0. For example, we can let $C = (x + y)\mathsf{a} \oplus (x + 1)\mathsf{b}$. The union of contexts is then defined as follows

$$\bigoplus_{\mathsf{a}\in S} e^1_{\mathsf{a}}\mathsf{a} \cup \bigoplus_{\mathsf{a}\in S} e^2_{\mathsf{a}}\mathsf{a} = \bigoplus_{\mathsf{a}\in S}(e^1_{\mathsf{a}} + e^2_{\mathsf{a}})\mathsf{a}$$

We assume that variables in X can only range over positive values, so that if $e_{\mathsf{a}} \neq 0$ then a is present in $\bigoplus_{\mathsf{a}\in S} e_{\mathsf{a}}\mathsf{a}$.

In the SOS rules we need to use the requirements $(W \cup R) \cap I = \emptyset$ and $R \subseteq W$. They are intended to be satisfied at the qualitative level, not necessarily at the quantitative one. Correspondingly, the disjointness condition $(W \cup R) \cap I = \emptyset$ is satisfied when $\forall \mathsf{a} \in I.\ (W \cup R)(\mathsf{a}) = 0$, and the inclusion condition $R \subseteq W$ is satisfied when $\forall \mathsf{a} \in S.\ R(\mathsf{a}) \neq 0 \Rightarrow W(\mathsf{a}) \neq 0$. Our new transition labels differ

from the ones in Figure 1 just because R, P, and W are now multisets. We keep the same SOS rules as before.

The advantage is that to each transition $\mathsf{P} \xrightarrow{\langle W \rhd R, I, P \rangle} \mathsf{P}'$ we can now assign a system of linear inequalities: $\forall \mathsf{a} \in S.\ R(\mathsf{a}) \leq W(\mathsf{a})$, where $R(\mathsf{a}) \in \mathbb{N}$ and $W(\mathsf{a})$ is an expression. The aim is to estimate, with no computational effort, the *relative quantities* of biological material which should be provided to the system to reach a desired configuration. This could be helpful during the setting phase of an *in vitro* experiment to avoid over-use of biological material, given its high cost. Please note that the qualitative nature of RS is unchanged, we only add some extra information that we elaborate by manipulating transition labels, only. Here we give an intuition with a short example.

Example 10. Let us consider the chemical reactions in Azimi et al. [2], Table 3, in particular reactions (i) and (vii); we will use their formalization in the syntax of RS, by keeping the stoichiometric numbers:

$$a_1 \triangleq (\{(\mathsf{hsf}, 3)\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\}) \qquad a_2 \triangleq (\{\mathsf{hsp}, \mathsf{hsf}_3\}, \{\mathsf{d_I}\}, \{\mathsf{hsp{:}hsf}, (\mathsf{hsf}, 2)\})$$

Reaction a_1 requires three copies of the entity hsf, while a_2 produces two copies of hsf. We assume that the context initially provides the set $C \triangleq x\mathsf{hsf} \oplus \mathsf{hsp} \oplus \mathsf{hsf}_3$ and then it provides the empty set, i.e. it is defined as $\mathsf{K} \triangleq C.\emptyset.\mathbf{0}$. The resulting system can only execute two transitions: in the first transition both reactions a_1 and a_2 are applied, in the second transition only reaction a_1 is applied:

$$[\mathsf{K}|a_1|a_2] \xrightarrow{\langle C \rhd R, I, P \rangle} [P|\emptyset.\mathbf{0}|a_1|a_2] \xrightarrow{\langle P \rhd R', I', P' \rangle} [P'|\mathbf{0}|a_1|a_2]$$

$$\begin{array}{lllll} \text{where} & R = 3\mathsf{hsf} \oplus \mathsf{hsp} \oplus \mathsf{hsf}_3 & I = \{\mathsf{d_I}\} & P = \mathsf{hsf}_3 \oplus \mathsf{hsp{:}hsf} \oplus 2\mathsf{hsf} \\ & R' = 3\mathsf{hsf} & I' = \{\mathsf{hsp}, \mathsf{d_I}\} & P' = \mathsf{hsf}_3 \end{array}$$

Now, from the first transition we extract the requirement $R(\mathsf{hsf}) = 3 \leq C(\mathsf{hsf}) = x$, while from the second transition we get $R'(\mathsf{hsf}) = 3 \leq P(\mathsf{hsf}) = 2$. If we would wanted a quantitative estimate of need of entity hsf, this comparison would reveal that the production of hsf is not sufficient to trigger the second reaction.

The connector operator. In Bodei et al. [9] and Brodo et al. [10] we have presented the encoding of RS into the `link`-calculus and discussed how to connect two encoded RS such that some of the entities produced by one RS are provided to the second one, similarly to what has been done in Bottoni et al. [7]. Thus we introduce a "connector", written as $\mathsf{P}_1 \overset{L}{\Rightarrow} \mathsf{P}_2$: when the RS process P_1 produces entities in the set L, these entities are available, at the next step, as reactants to the continuations of both RS processes. When $L = \emptyset$, there cannot be any exchange of entities and P_1 and P_2 run in parallel. We denote this by $\mathsf{P}_1 \parallel \mathsf{P}_2$.

7 Conclusion and future work

We have presented an SOS semantics for the Reaction Systems that generates a labelled transition system. We have revised RSs as processes, formulating a set of

$$\frac{\mathsf{P}_1 \xrightarrow{\langle W_1 \triangleright R_1, I_1, P_1 \rangle} \mathsf{P} \quad \mathsf{P}_2 \xrightarrow{\langle W_2 \triangleright R_2, I_2, P_2 \rangle} [\mathsf{M}]}{\mathsf{P}_1 \overset{L}{\Rightarrow} \mathsf{P}_2 \xrightarrow{\langle W_1 \cup W_2 \triangleright R_1 \cup R_2, I_1 \cup I_2, P_1 \cup P_2 \rangle} \mathsf{P} \overset{L}{\Rightarrow} [\mathsf{M}|(L \cap P_1)]} \ (Lnk)$$

Fig. 2: SOS semantics rule for the connector operator

ad-hoc inference rules. Our flexible framework allows one to add new operators in a natural way. The transition labels add expressivity at the computation allowing for additional analysis, as we did in Section 4. In Section 5 we have described a preliminary interpreter in logic programming which implements the verification of BioHML formulas on computations of RS processes with nondeterministic contexts in our framework. As future work we plan to define SOS semantics for other synchronous rewrite-rule systems to define a uniform computational framework. We plan to improve our implementation including its functionalities, interface, scalability and usability, for the analysis of large case studies. We want to study the relation to analysis techniques [16,15,13].

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

References

1. Azimi, S.: Steady states of constrained reaction systems. Theor. Comput. Sci. **701**(C), 20–26 (2017). https://doi.org/10.1016/j.tcs.2017.03.047
2. Azimi, S., Iancu, B., Petre, I.: Reaction system models for the heat shock response. Fundam. Informaticae **131**(3-4), 299–312 (2014). https://doi.org/10.3233/FI-2014-1016
3. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Investigating dynamic causalities in reaction systems. Theor. Comput. Sci. **623**, 114–145 (2016). https://doi.org/10.1016/j.tcs.2015.11.041
4. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Troina, A.: Bisimulations in calculi modelling membranes. Form. Asp. Comput. **20**(4), 351–377 (2008). https://doi.org/10.1007/s00165-008-0071-x
5. Bodei, C., Brodo, L., Bruni, R.: A formal approach to open multiparty interactions. Theor. Comput. Sci. **763**, 38–65 (2019). https://doi.org/10.1016/j.tcs.2019.01.033
6. Bodei, C., Brodo, L., Bruni, R.: The link-calculus for open multiparty interactions. Inf. Comput. **275** (2020). https://doi.org/10.1016/j.ic.2020.104587
7. Bottoni, P., Labella, A., Rozenberg, G.: Networks of reaction systems. Int. J. Found. Comput. Sci. **31**, 53–71 (2020). https://doi.org/10.1142/S0129054120400043
8. Brijder, R., Ehrenfeucht, A., Main, M., Rozenberg, G.: A tour of reaction systems. Int. J. Found. Comput. Sci. **22**(07), 1499–1517 (2011). https://doi.org/10.1142/S0129054111008842
9. Brodo, L., Bruni, R., Falaschi, M.: Enhancing reaction systems: A process algebraic approach. In: The Art of Modelling Computational Systems. LNCS, vol. 11760, pp. 68–85. Springer Berlin (2019). https://doi.org/10.1007/978-3-030-31175-9_5
10. Brodo, L., Bruni, R., Falaschi, M.: A process algebraic approach to reaction systems. Theor. Comput. Sci. (2020). https://doi.org/10.1016/j.tcs.2020.09.001, in Press

11. Cardelli, L., Gordon, A.D.: Mobile ambients. Theor. Comput. Sci. **240**(1), 177–213 (2000). https://doi.org/10.1016/S0304-3975(99)00231-5
12. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Forward and backward bisimulations for chemical reaction networks. In: CONCUR'15. vol. 42, pp. 226–239. Schloss Dagstuhl Publ. (2015). https://doi.org/10.4230/LIPIcs.CONCUR.2015.226
13. Chiarugi, D., Falaschi, M., Olarte, C., Palamidessi, C.: Compositional modelling of signalling pathways in timed concurrent constraint programming. In: BCB'10. pp. 414–417. ACM (2010). https://doi.org/10.1145/1854776.1854843
14. Corolli, L., Maj, C., Marinia, F., Besozzi, D., Mauri, G.: An excursion in reaction systems: From computer science to biology. Theor. Comput. Sci. **454**, 95–108 (2012). https://doi.org/10.1016/j.tcs.2012.04.003
15. Falaschi, M., Olarte, C., Palamidessi, C.: Abstract interpretation of temporal concurrent constraint programs. Theory and Practice of Logic Programming **15**(3), 312–357 (2015). https://doi.org/10.1017/S1471068413000641
16. Falaschi, M., Olarte, C., Palamidessi, C.: A framework for abstract interpretation of timed concurrent constraint programs. In: PPDP'09. pp. 207–218. ACM (2009). https://doi.org/10.1145/1599410.1599436
17. Falaschi, M., Palma, G.: A Logic Programming Approach to Reaction Systems. In: DIP'20. OASIcs, vol. 86, pp. 6:1–6:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/OASIcs.Gabbrielli.6
18. Gordon, A., Cardelli, L.: Equational properties of mobile ambients. Math. Struct. Comput. Sci. **13**(3), 371–408 (2003). https://doi.org/10.1017/S0960129502003742
19. Hennessy, M., Milner, R.: On observing nondeterminism and concurrency. In: ICALP'80. LNCS, vol. 85, pp. 299–309. Springer (1980). https://doi.org/10.1007/3-540-10003-2_79
20. Hillston, J.: A compositional approach to performance modelling. Ph.D. thesis, University of Edinburgh, UK (1994), `http://hdl.handle.net/1842/15027`
21. Kleijn, J., Koutny, M., Mikulski, Ł., Rozenberg, G.: Reaction systems, transition systems, and equivalences. In: Adventures Between Lower Bounds and Higher Altitudes. LNCS, vol. 11011, pp. 63–84. Springer (2018). https://doi.org/10.1007/978-3-319-98355-4_5
22. Milner, R.: A Calculus of Communicating Systems. Lecture Notes in Computer Science 92, Springer (1980). https://doi.org/10.1007/3-540-10235-3
23. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: ICALP'92. LNCS, vol. 623, pp. 685–695. Springer (1992). https://doi.org/10.1007/3-540-55719-9_114
24. Okubo, F., Yokomori, T.: The computational capability of chemical reaction automata. Natural Computing **15**(2), 215–224 (2016). https://doi.org/10.1007/s11047-015-9504-7
25. Plotkin, G.D.: A structural approach to operational semantics. Tech. Rep. DAIMI FN-19, Computer Science Department, Aarhus University (1981)
26. Plotkin, G.D.: An operational semantics for CSP. In: IFIP Working Conf. on Formal Description of Programming Concepts - II. pp. 199–226. North-Holland (1982)
27. Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebraic Methods Program. **60-61**, 17–139 (2004). https://doi.org/10.1016/j.jlap.2004.05.001
28. Sangiorgi, D.: Introduction to Bisimulation and Coinduction. Cambridge University Press, USA (2011). https://doi.org/10.1017/CBO9780511777110