Exploiting Modularity of SOS Semantics to Define Quantitative Extensions of Reaction Systems*

Linda Brodo[1][0000−0002−4455−2419], Roberto Bruni[2][0000−0002−7771−4154], Moreno Falaschi[3][0000−0002−6659−3828], Roberta Gori[2][0000−0002−7424−9576], Francesca Levi[2][0000−0002−6137−0019], and Paolo Milazzo[2][0000−0002−7309−6424]

[1] Dipartimento di Scienze Economiche e Aziendali, Università di Sassari, Via Muroni 25, Sassari, Italy, brodo@uniss.it.

[2] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, Pisa, Italy, {roberto.bruni,roberta.gori,francesca.levi,paolo.milazzo}@unipi.it.

[3] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Via Roma 56, Siena, Italy, moreno.falaschi@unisi.it.

Abstract. Reaction Systems (RSs) are a successful natural computing framework inspired by chemical reaction networks. A RS consists of a set of entities and a set of reactions. Entities can enable or inhibit each reaction, and are produced by reactions or provided by the environment. In a previous paper, we defined an original labelled transition system (LTS) semantics for RSs in the structural operational semantics (SOS) style. This approach has several advantages: (i) it provides a formal specification of the RS dynamics that enables the reuse of many formal analysis techniques and favors the implementation of tools, and (ii) it facilitates the definition of extensions of the RS framework by simply modifying some of the SOS rules in a modular way. In this paper, we demonstrate the extensibility of the framework by defining two quantitative variants of RSs: with reaction delays/durations, and with concentration levels. We provide a prototype logic programming implementation and apply our tool to a RS model of *Th* cells differentiation in the immune system.

Keywords: Bioinformatics, SOS rules, Reaction Systems, Logic programming

1 Introduction

Inspired by natural phenomena, many new computational formalisms have been introduced to model different aspects of biology. Basic chemical reactions inspired Reaction Systems (RSs), a *qualitative* modeling formalism introduced by Ehrenfeucht and Rozenberg [14,6] that is based on two opposite mechanisms: *facilitation* and *inhibition*. Facilitation means that a reaction can occur only if all its reactants are present, while inhibition means that the reaction cannot occur if any of its inhibitors is present. A *reaction* is hence a triple (R, I, P), where R, I and P are sets of entities representing *reactants*, *inhibitors* and *products*, respectively. A reaction system is represented by a set of reactions having such a form, together with a (finite) support set S containing all of the entities that can appear in reactions. The state of a Reaction System consists of a finite set of entities representing the molecular species that are present in the real system being modeled. Quantities (or concentrations) are not taken into account: the presence of an entity represents the availability of the corresponding molecule in *high concentration*.

The theory of RSs is based on three assumptions: **no permanency**, any entity vanishes unless it is sustained by a reaction; **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; **no competition**, an entity is either available for all reactions, or it is not available at all. The computation of the next state of a Reaction System is a deterministic procedure. However, the overall dynamics is influenced by the (set of) contextual entities received (non-deterministically) from the external environment at each step. Such entities join the current state of the system and participate to enabling and disabling reactions. The behaviour of a RS is hence defined as

* Research supported by University of Pisa PRA_2020_26 *Metodi Informatici Integrati per la Biomedica*, by MIUR PRIN Project 201784YSZ5 *ASPRA–Analysis of Program Analyses*, and by University of Sassari *Fondo di Ateneo per la ricerca 2020*.

a discrete time interactive process consisting of a *context sequence* (the sets of entities received at each step form the environment), a *result sequence* (the sets of entities produced at each step by reactions) and a *state sequence* (the combined sets of entities present in the system at each step). Since their introduction, RSs have shown to be a quite general computation model whose application ranges from the modeling of biological phenomena [3,9,2,4], and molecular chemistry [20] to theoretical foundations of computing [12,13].

Labelled Transition Systems (LTSs) are a powerful structure to model the behaviour of interacting processes. In the case of processes described using an algebraic language, LTSs can be conveniently defined following the Structural Operational Semantics (SOS) approach [23]. Given the signature of the language, an SOS system assigns some inference rules to each operator: the conclusion of each rule is the transition of a composite term, which is determined from those of its constituents (appearing as premises of the rule). The SOS approach has been particularly successful in the area of process algebras [18,22,15].

In [8] an LTS semantics in the SOS style for RSs has been proposed. Such a semantics is able to faithfully represent the ordinary semantics of RS, it allows more general experiments to be conducted, and it enriches the expressiveness of contexts. A similar formalization focusing on local scopes for RS entities was proposed in [21]. The SOS approach has several advantages: 1) compositionality: the behaviour of a composite system is defined in term of the behaviours of its constituents; 2) transparency: each transition label conveys information about all the activities connected to the execution step it describes; 3) the notion of contexts is better integrated in the framework; 4) different kinds of contexts (nondeterministic, recursive) are allowed, so to combine different experiments in a single LTS and to account for possibly infinite (regular) computations; 5) extensibility: the definition of enhanced RS variants can be obtained by modifying/adding language operators and SOS rules in a modular fashion; 6) SOS rules facilitate implementation in a declarative language and the use of standard techniques for defining process equivalences.

In this paper we demonstrate the extensibility of the SOS approach to RSs by extending the classical RSs with some relevant quantitative features. First, we add the possibility to express reaction delays and durations. Thanks to this feature we encode reactions with different speeds. Indeed a reaction with associated duration n will deliver its products after n steps. In more details we associate with each reaction a natural number starting from zero (the fastest reaction, the smallest delay, the product being immediately available) up to any value $n > 0$ (slower reactions, greater delay). Following this idea also a duration of 'permanency' can be specified for each reaction. A reaction r that has delay n and duration m will deliver, if applicable, its products after n steps while such products will be available for the following m steps. The second feature which we introduce is some quantitative information that tells how concentrations influence the application of a reaction. This is obtained by adding to each entity in a reaction an approximated quantitative information that will be necessary for enabling the reaction. This feature will allow us to deal with reactions that take into account different levels of concentrations. We note that we still maintain a qualitative perspective on the biological system, since the approximate quantitative information will be used to determine the set of reactions that can be applied in any step whereas competition between different enabled reactions will not be considered. We also provide a freely available prototype implementation in logic programming that allows us to compute and inspect the resulting LTS to perform computational experiments. Finally we apply our tool to the system controlling the differentiation of *Th* cells in the immune system presented in [16,5].

The structure of the paper is as follows. In Section 2 we recall the basics of RSs. In Section 3 we recall the syntax and operational semantics of our process algebra for RSs, The original contribution starts from Section 4, where we introduce the concepts of delay and duration, and define the corresponding operators. Then, in Section 5 we introduce linear functions for expressing the concentration levels of the entities which are necessary to enable a reaction. A prototype implementation in logic programming of our semantic framework is described in Section 6 on the basis of a case study about a system controlling the differentiation of *Th* cells in the immune system [16,5]. Section 7 discusses some related work and concludes the paper.

2 Reaction Systems

The theory of Reaction Systems (RSs) [6] was born in the field of Natural Computing to model the behaviour of biochemical reactions in living cells. While our contribution builds on a process algebraic presentation of RSs, we recall here the main concepts as introduced in the classical set theoretic version. In the following, 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.

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 have to be present in the current state for the reaction to take place. The presence of any of the inhibitors blocks the reaction. Products are the outcome of the reaction, to be released in the next state. We denote with $rac(S)$ the set of all reactions over S. Given $W \subseteq S$, the result of $a = (R, I, P) \in rac(S)$ on W, denoted $res_a(W)$, is given by:

$$res_a(W) \triangleq \begin{cases} P & \text{if } en_a(W) \\ \emptyset & \text{otherwise} \end{cases} \qquad en_a(W) \triangleq R \subseteq W \ \wedge \ I \cap W = \emptyset$$

where $en_a(W)$ is called the *enabling predicate*.

A Reaction System is a pair $\mathcal{A} = (S, A)$ where S is the set of entities, and $A \subseteq rac(S)$ is a finite set of reactions over S. Given $W \subseteq S$, the result of the application of reactions A to W, denoted $res_A(W)$, can be obtained by lifting function res_a to sets of reactions, i.e., $res_A(W) \triangleq \cup_{a \in A} res_a(W)$.

Since living cells are seen as open systems that react to environmental stimuli, the behaviour of a RS is formalized in terms of an *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 $i \in [0, n-1]$. The context sequence γ represents the environment, while the result sequence δ is entirely determined by γ and A. 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*. Note that 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.

Example 1. We consider a toy RS defined as $\mathcal{A} \triangleq (S, A)$ where $S \triangleq \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, and the set of reactions $A \triangleq \{a_1\}$ only contains the reaction $a_1 \triangleq (\{\mathsf{a}, \mathsf{b}\}, \{\mathsf{c}\}, \{\mathsf{b}\})$, to be written more concisely as $(\mathsf{ab}, \mathsf{c}, \mathsf{b})$. Then, we consider a $4-$*steps* interactive process $\pi \triangleq (\gamma, \delta)$, where $\gamma \triangleq \{C_0, C_1, C_2, C_3\}$, with $C_0 \triangleq \{\mathsf{a}, \mathsf{b}\}$, $C_1 \triangleq \{\mathsf{a}\}$, $C_2 \triangleq \{\mathsf{c}\}$, and $C_3 \triangleq \{\mathsf{c}\}$; and $\delta \triangleq \{D_0, D_1, D_2, D_3\}$, with $D_0 \triangleq \emptyset$, $D_1 \triangleq \{\mathsf{b}\}$, $D_2 \triangleq \{\mathsf{b}\}$, and $D_3 \triangleq \emptyset$. 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 process algebras such as CCS [18], in [8] the authors introduced an algebraic syntax for RSs and equipped it with SOS inference rules defining the behaviour of each operator. This allows us to consider a LTS semantics for RSs, where states are terms of the algebra, each transition corresponds to a step of the RS and transition labels retain some information on the entities needed to perform each step.

Definition 2 (RS processes). *Let S be a set of entities. An* RS process P *is any term defined by the following grammar:*

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

where $R, I, P \subseteq S$ are non empty sets of entities, $C, D \subseteq S$ 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 3. Note that the processes $\emptyset$ and $\mathbf{0}$ are not interchangeable: as it will become clear from the operational semantics, the process $\emptyset$ can perform just a trivial transition to itself, while the process $\mathbf{0}$ cannot perform any transition and can be used to stop the computation.

Definition 4 (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 (the shifting of γ starting at the i-th step). We write $[\![\mathcal{A}, \pi]\!]$ as a shorthand for $[\![\mathcal{A}, \pi]\!]_0$.

Example 5. Here, we give the encoding of the reaction system, $\mathcal{A} = (S, A)$, defined in Example 1. The resulting RS process is as follows:

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

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 4 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. Nondeterminstic contexts can be used to descrive several alternative experimental conditions, while recursion can be exploited to extract some regularity in the longterm behaviour of a RS. Together, they can deal with any combination of in-breadth/in-depth behavioural analysis.

Example 6. Let us further elaborate on our running example. Suppose we want to enhance the behaviour of the context by defining a process $\mathsf{K}' \triangleq \mathsf{K}_1 + \mathsf{K}_2$ that non-deterministically can behave as either $\mathsf{K}_1 \triangleq \mathsf{ab.a.c.c.}\mathbf{0}$ (as in Example 5), or $\mathsf{K}_2 \triangleq \mathsf{rec}\ X.\ \mathsf{ab.a.}X$ (which is a recursive behaviour that allows the reaction to be always enabled). Then we simply set $\mathsf{P}' \equiv [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \mathsf{K}']$.

Definition 7 (Label). *A label is a tuple $\langle W \rhd R, I, P\rangle$ with $W, R, I, P \subseteq S$. The set of transition labels is ranged over by ℓ.*

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

$$\frac{}{D \xrightarrow{\langle D \rhd \emptyset,\emptyset,\emptyset\rangle} \emptyset}\ (Ent) \qquad \frac{}{C.\mathsf{K} \xrightarrow{\langle C \rhd \emptyset,\emptyset,\emptyset\rangle} \mathsf{K}}\ (Cxt) \qquad \frac{\mathsf{K}_1 \xrightarrow{\ell} \mathsf{K}_1'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_1'}\ (Suml) \qquad \frac{\mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_2'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_2'}\ (Sumr)$$

$$\frac{}{(R,I,P) \xrightarrow{\langle \emptyset \rhd R,I,P\rangle} (R,I,P) \mid P}\ (Pro) \qquad \frac{J \subseteq I \quad Q \subseteq R \quad J \cup Q \neq \emptyset}{(R,I,P) \xrightarrow{\langle \emptyset \rhd J,Q,\emptyset\rangle} (R,I,P)}\ (Inh) \qquad \frac{\mathsf{K}[^{\mathsf{rec}\ X.\mathsf{K}}/_X] \xrightarrow{\ell} \mathsf{K}'}{\mathsf{rec}\ X.\ \mathsf{K} \xrightarrow{\ell} \mathsf{K}'}\ (Rec)$$

$$\frac{\mathsf{M}_1 \xrightarrow{\ell_1} \mathsf{M}_1' \quad \mathsf{M}_2 \xrightarrow{\ell_2} \mathsf{M}_2' \quad \ell_1 \frown \ell_2}{\mathsf{M}_1 \mid \mathsf{M}_2 \xrightarrow{\ell_1 \cup \ell_2} \mathsf{M}_1' \mid \mathsf{M}_2'}\ (Par) \qquad \frac{\mathsf{M} \xrightarrow{\langle W \rhd R,I,P\rangle} \mathsf{M}' \quad R \subseteq W}{[\mathsf{M}] \xrightarrow{\langle W \rhd R,I,P\rangle} [\mathsf{M}']}\ (Sys)$$

Fig. 1. SOS semantics of the RS processes.

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*), $\emptyset \xrightarrow{\langle \emptyset \rhd \emptyset,\emptyset,\emptyset\rangle} \emptyset$. 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 $\mathsf{rec}\ X.\ \mathsf{a.b}.X \xrightarrow{\langle \mathsf{a} \rhd \emptyset,\emptyset,\emptyset\rangle} \mathsf{b.rec}\ X.\ \mathsf{a.b}.X$ 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. The rule (*Par*) puts two processes in parallel by pooling their labels and joining all the set components of the labels. The sanity check $\ell_1 \frown \ell_2$ is required to guarantee that there is no conflict between reactants and inhibitors of the applied reactions:

$$\langle W_1 \rhd R_1, I_1, P_1\rangle \frown \langle W_2 \rhd R_2, I_2, P_2\rangle \triangleq (W_1 \cup W_2 \cup R_1 \cup R_2) \cap (I_1 \cup I_2) = \emptyset$$

In the conclusion of rule (*Par*) we write $\ell_1 \cup \ell_2$ for the componentwise union of labels:

$$\langle W_1 \rhd R_1, I_1, P_1\rangle \cup \langle W_2 \rhd R_2, I_2, P_2\rangle \triangleq \langle W_1 \cup W_2 \rhd R_1 \cup R_2, I_1 \cup I_2, P_1 \cup P_2\rangle$$

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 it is embedded in rule (*Par*).

Example 9. 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 5. The process P_0 has a unique outgoing transition, whose formal derivation is given below:

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

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

1. $\mathsf{P}_2 \xrightarrow{\langle \mathsf{bc} \rhd \mathsf{c}, \emptyset, \emptyset \rangle} \mathsf{P}_3$ shows that the presence of c has played some role in inhibiting the reaction;
2. $\mathsf{P}_2 \xrightarrow{\langle \mathsf{bc} \rhd \emptyset, \mathsf{a}, \emptyset \rangle} \mathsf{P}_3$ shows that the absence of a has played some role in inhibiting the reaction.
3. $\mathsf{P}_2 \xrightarrow{\langle \mathsf{bc} \rhd \mathsf{c}, \mathsf{a}, \emptyset \rangle} \mathsf{P}_3$ shows that the presence of c and the absence of a inhibited the reaction; this label is thus more informative than the previous two (in a sense formalised in [8]).

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 theorem from [8] shows that the rewrite steps of a RS exactly match the transitions of its corresponding RS process.

Theorem 10. *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. *$\forall i \in [0, n-1]$, $\mathsf{P}_i \xrightarrow{\langle W \rhd R, I, P \rangle} \mathsf{P}$ 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 $\mathsf{P}_i \xrightarrow{\langle W_i \rhd R, I, D_{i+1} \rangle} \mathsf{P}_{i+1}$.*

Remark 11. 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

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

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

Example 9 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 [8].

4 Delays and Durations

In Biology it is well known that reactions occur with different frequencies. For example, since enzymes catalyze reactions, many reactions are more frequent when some enzymes are present, and less frequent when such enzymes are absent. Moreover, reactions describing complex transformations may require time before releasing their products. To capture these dynamical aspects in our framework by preserving the discrete and abstract nature of RS, we propose a discretization of the *delay* between two occurrences of a reaction by using a scale of natural numbers, from 0 (smallest delay, highest frequency) up to n (increasing delay, lower frequency).

Intuitively, the notation D^n stands for making the entities D available after n time units, and we use the shorthand D for D^0. Similarly, we can associate a delay value to the product of each reaction by writing $(R, I, P)^n$ when the product of the reaction will be available after n time units, and we write (R, I, P) for $(R, I, P)^0$. The syntax for mixture processes is thus extended as below and the operational semantics is changed accordingly (see Fig. 2).

$$\mathsf{M} ::= (R, I, P)^n \mid D^n \mid \mathsf{K} \mid \mathsf{M} | \mathsf{M}$$

Rule (*Tick*) represents the passing of one time unit, while rule (*Set*) notify the availability of entities whose delay has expired. Rule (*ProS*) attaches to the product of the reaction the same delay as the one of the reaction itself, while rule (*InhS*) is used when the reaction is not enabled.

Note that the context definition is unchanged. The encoding described in Def. 4 still applies.

Example 12. Let us consider two RSs sharing the same entity set $S = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ and the same reactions $a_1 = (\mathsf{a}, \mathsf{b}, \mathsf{b})$, $a_2 = (\mathsf{b}, \mathsf{a}, \mathsf{a})$, $a_3 = (\mathsf{ac}, \mathsf{b}, \mathsf{d})$, $a_4 = (\mathsf{d}, \mathsf{a}, \mathsf{c})$, but working with different reaction speeds. For simplicity we assume only two speed levels are distinguished: 0 the fastest and 1 the slowest. The reaction system P_1 provides the following speed assignment to the reactions:

$$\frac{}{D \xrightarrow{\langle D \triangleright \emptyset,\emptyset,\emptyset\rangle} \emptyset}\ (Set) \qquad \frac{}{D^{n+1} \xrightarrow{\langle \emptyset \triangleright \emptyset,\emptyset,\emptyset\rangle} D^n}\ (Tick)$$

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

Fig. 2. SOS semantics with delays and durations.

$\{a_1^1, a_2, a_3, a_4^1\}$. The reaction system P_2 provides the following speed assignment to the reactions: $\{a_1, a_2^1, a_3^1, a_4\}$. We assume that the context process for both reaction systems is just $\mathsf{ac}.\emptyset.\mathbf{0}$:

$$\mathsf{P}_1 = [\mathsf{ac}.\emptyset.\mathbf{0}|a_1^1|a_2|a_3|a_4^1] \xrightarrow{\langle \mathsf{ac} \triangleright \mathsf{ac},\mathsf{bd},\mathsf{bd}\rangle} [\emptyset.\mathbf{0}|\mathsf{b}^1|\mathsf{d}|a_1^1|a_2|a_3|a_4^1] \xrightarrow{\langle \mathsf{d} \triangleright \mathsf{d},\mathsf{abc},\mathsf{c}\rangle} [\mathbf{0}|\mathsf{c}^1|\mathsf{b}|a_1^1|a_2|a_3|a_4^1]$$

$$\mathsf{P}_2 = [\mathsf{ac}.\emptyset.\mathbf{0}|a_1|a_2^1|a_3^1|a_4] \xrightarrow{\langle \mathsf{ac} \triangleright \mathsf{ac},\mathsf{bd},\mathsf{bd}\rangle} [\emptyset.\mathbf{0}|\mathsf{b}|\mathsf{d}^1|a_1|a_2^1|a_3^1|a_4] \xrightarrow{\langle \mathsf{b} \triangleright \mathsf{b},\mathsf{acd},\mathsf{a}\rangle} [\mathbf{0}|\mathsf{a}^1|\mathsf{d}|a_1|a_2^1|a_3^1|a_4]$$

Additionally, inspired by [7], we can also provide entities with a duration, i.e. entities that last a finite number of steps. To this aim we use the syntax $D^{[n,m]}$ to represent the availability of D for m time units starting after n time units from the current time. By assuming that each reaction only produces entities with the same duration, we can describe duration and delay also associated to reactions: $(R,I,P)^{[n,m]}$ means that all the entities in P (the products) have a delay of n but will last m steps (once they appear in the state). While we could easily define the SOS rules for the above processes, we note that durations are just syntax sugar: we can simply let

$$D^{[n,m]} \triangleq \prod_{k=n}^{n+m} D^k \qquad\qquad (R,I,P)^{[n,m]} \triangleq \prod_{k=n}^{n+m} (R,I,P)^k.$$

For example, we have $\mathsf{a}^{[2,2]} = \mathsf{a}^2|\mathsf{a}^3|\mathsf{a}^4$ and $\mathsf{a}^{[0,0]} = \mathsf{a}^0 = \mathsf{a}$.

Example 13. The cell cycle is a series of sequential events leading to cell duplication. It consists of four phases: G_1, S, G_2 and M. The first three phases (G_1, S, and G_2) are called interphase (I). In these phases, the main event which happens is the replication of DNA. In the last phase (M), called mitosis, the cell segregates the duplicated sets of chromosomes between daughter cells, and then divides. The duration of the cell cycle depends on the type of cell.

In [24] a Delay Differential Equation model of tumour growth has been proposed, that includes the immune system response and a phase-specific drug able to alter the natural course of action of the cell cycle of the tumour cells. A delay is used to model the duration of the interphase.

Inspired from [24] we define a RS model of tumour growth using delays and durations. We consider two populations of tumour cells: those in the interphase of the cell cycle ($\mathsf{T_I}$) and those in mitosis phase ($\mathsf{T_M}$). We assume that cells reside in the interphase for σ time units. Moreover, we represent the drug with entity D and assume that, once received from the environment, it takes an active form $\mathsf{D_a}$ and disappears after a delay of δ time units. The reactions of the model are the following: $\mathsf{a_1} = (\mathsf{T_I}, \mathsf{D_a}, \mathsf{T_M})^\sigma$, $\mathsf{a_2} = (\mathsf{T_M}, \emptyset, \mathsf{T_I})$, $\mathsf{a_3} = (\mathsf{D}, \emptyset, \mathsf{D_a})^{[0,\delta]}$. Let $\mathsf{A} = \mathsf{a_1}|\mathsf{a_2}|\mathsf{a_3}$.

Let us assume that the system starts from a configuration in which tumour cells are in the interphase. Hence, the RS process is $\mathsf{P} = [\mathsf{K}|\mathsf{T_I}|\mathsf{A}]$, where K is a contex process. Now, by providing different definitions for K we can emulate different drug administration strategies. For instance, let us consider $\sigma = 1$, $\delta = 1$ and these two context processes:

- $\mathsf{K_1} = \mathsf{rec}\ X.\emptyset.X$ (i.e., drug not administered)
- $\mathsf{K_2} = \mathsf{rec}\ X.\mathsf{D}.\emptyset.\emptyset.\emptyset.X$ (i.e., drug administered every 4 time units)

Now, when no drug is administered, tumour cells execute the cell cycle infinitely:

$$\mathsf{P}[{}^{\mathsf{K_1}}/_{\mathsf{K}}] = [\mathsf{K_1}|\mathsf{T_I}|\mathsf{A}] \xrightarrow{\langle \mathsf{T_I} \triangleright \mathsf{T_I},\mathsf{D_a}\mathsf{T_M}\mathsf{D},\mathsf{T_M}\rangle} [\mathsf{K_1}|\mathsf{T_M^1}|\mathsf{A}] \xrightarrow{\langle \emptyset \triangleright \emptyset,\mathsf{T_I}\mathsf{T_M}\mathsf{D},\emptyset\rangle}$$
$$[\mathsf{K_1}|\mathsf{T_M}|\mathsf{A}] \xrightarrow{\langle \mathsf{T_M} \triangleright \mathsf{T_M},\mathsf{T_I}\mathsf{D},\mathsf{T_I}\rangle} [\mathsf{K_1}|\mathsf{T_I}|\mathsf{A}] \xrightarrow{\langle \mathsf{T_I} \triangleright \mathsf{T_I},\mathsf{D_a}\mathsf{T_M}\mathsf{D},\mathsf{T_M}\rangle} \ldots$$

When the drug is administered every 4 time units, the cell cycle is interrupted after a few steps:

$$\mathsf{P}[{}^{\mathsf{K}_2}/_{\mathsf{K}}] = [\mathsf{K}_2|\mathsf{T}_\mathsf{I}|\mathsf{A}] \xrightarrow{\langle \mathsf{T}_\mathsf{I}\mathsf{D} \triangleright \mathsf{T}_\mathsf{I}\mathsf{D}, \mathsf{D}_\mathsf{a}\mathsf{T}_\mathsf{M}, \mathsf{T}_\mathsf{M}\mathsf{D}_\mathsf{a} \rangle} [\emptyset.\emptyset.\emptyset.\mathsf{K}_2|\mathsf{D}^1_\mathsf{a}|\mathsf{D}_\mathsf{a}|\mathsf{T}^1_\mathsf{M}|\mathsf{A}] \xrightarrow{\langle \mathsf{D}_\mathsf{a} \triangleright \mathsf{D}_\mathsf{a}, \mathsf{T}_\mathsf{I}\mathsf{T}_\mathsf{M}\mathsf{D}, \emptyset \rangle}$$
$$[\emptyset.\emptyset.\mathsf{K}_2|\mathsf{D}_\mathsf{a}|\mathsf{T}_\mathsf{M}|\mathsf{A}] \xrightarrow{\langle \mathsf{D}_\mathsf{a}\mathsf{T}_\mathsf{M} \triangleright \mathsf{D}_\mathsf{a}\mathsf{T}_\mathsf{M}, \mathsf{T}_\mathsf{I}\mathsf{D}, \mathsf{T}_\mathsf{I} \rangle} [\emptyset.\mathsf{K}_2|\mathsf{T}_\mathsf{I}|\mathsf{A}] \xrightarrow{\langle \mathsf{T}_\mathsf{I} \triangleright \mathsf{T}_\mathsf{I}, \mathsf{D}_\mathsf{a}\mathsf{T}_\mathsf{M}\mathsf{D}, \mathsf{T}_\mathsf{M} \rangle}$$
$$[\mathsf{K}_2|\mathsf{T}^1_\mathsf{M}|\mathsf{A}] \xrightarrow{\langle \mathsf{D} \triangleright \mathsf{D}, \mathsf{T}_\mathsf{I}\mathsf{T}_\mathsf{M}, \mathsf{D}_\mathsf{a} \rangle} [\emptyset.\emptyset.\emptyset.\mathsf{K}_2|\mathsf{D}^1_\mathsf{a}|\mathsf{D}_\mathsf{a}|\mathsf{T}_\mathsf{M}|\mathsf{A}] \xrightarrow{\langle \mathsf{T}_\mathsf{M}\mathsf{D}_\mathsf{a} \triangleright \mathsf{D}_\mathsf{a}\mathsf{T}_\mathsf{M}, \mathsf{T}_\mathsf{I}\mathsf{D}, \mathsf{T}_\mathsf{I} \rangle} [\emptyset.\emptyset.\mathsf{K}_2|\mathsf{D}_\mathsf{a}|\mathsf{T}_\mathsf{I}|\mathsf{A}]\,.$$

Alternative drug scheduling could be tested by providing alternative definitions for K.

5 Concentration Levels through Linear Functions

Quantitative modelling of chemical reaction requires taking molecule concentrations into account. An abstract representation of concentrations that is considered in many formalisms is based on *concentration levels*: rather than representing such quantities as real numbers, a finite classification is considered (e.g., *low/medium/high*) with a granularity that reflects the number of concentrations levels at which significant changes in the behaviour of the molecule are observed. In classical RSs, the modelling of concentration levels would require using different entities for the same molecule (e.g., $\mathsf{a}_\mathtt{l}$, $\mathsf{a}_\mathtt{m}$, and $\mathsf{a}_\mathtt{h}$ for low, medium and high concentration of a, respectively). This may introduce some additional complexity due to the need of guaranteeing that only one of these entities is present at any time for the state to be consistent. Moreover, consistency would be put at risk also by the fact that entities representing different levels of the same molecule (e.g., $\mathsf{a}_\mathtt{l}$ and $\mathsf{a}_\mathtt{h}$) could be provided at the same time by the context.

We now enhance RS process by adding some quantitative information associated to each entity of each reaction, so that levels are just natural numbers and the concentration levels of the products depend on the concentration levels of reactants. The idea is to associate linear expressions (such as $e = m \cdot x + n$, with $m \in \mathbb{N}$ and $n \in \mathbb{N}^+$)[1] to reactants and products of each reaction (we write $s(e)$ to state that expression e is associated to entity s). Expressions associated to reactants are used as *patterns* to match the current levels of the entities involved in the reaction. Pattern matching allows variable x (the same for all reactants) to be instantiated with a value that depends on the levels. Then, linear expressions associated to products (that can contain, again, variable x) can be evaluated to compute the concentration levels of those entities. Expressions can be associated also to reaction inhibitors in order let such entities inhibit the reaction only when their concentration level is above a given threshold. For expressions associated to inhibitors, we will require them to be *closed*, namely they cannot contain the $m \cdot x$ term and simply correspond to a positive natural number.

Also the state of the system has to take into account concentration levels. Consequently, in the definition of states we will exploit again closed expressions to obtain that each entity in the a state is associated to a natural number representing its concentration level.

Example 14. Assume that we want to write a reaction that produces c with a concentration level that corresponds to the current concentration level of a, and that requires b not to be present at a concentration level higher than 1. Such a reaction would be $r = (R, I, P)$ where $R = \mathsf{a}(x+1)$, $I = \mathsf{b}(2)$ and $P = \mathsf{c}(x+1)$. In the state $\{\mathsf{a}(3), \mathsf{b}(1)\}$. Reaction r is enabled by taking $x = 2$ (the maximum value for x that satisifes $x + 1 \leq 3$). Since $\mathsf{b}(1) < \mathsf{b}(2)$, entity c will be produced with concentration level $(x + 1) = 3$. On the contrary, in the state $\{\mathsf{a}(2), \mathsf{b}(2)\}$ the reaction a is not enabled because the concentration of the inhibitor is too high.

To formalize the above linear constraints we introduce some notations. A *closed* linear expression is just a natural number, and $e[v/x]$ represents the *substitution* of variable x with the value v in e. A *pattern* $p = \{s_1(e_1), ..., s_k(e_k)\}$ is a set of associations of linear expressions to entities. We write $p(s_i)$ for the linear expression associated with s_i in p. A pattern p is closed if $p(s_i) \in \mathbb{N}$ for any $s_i \in S$ and we write $p[v/x]$ to mean the closed pattern obtained as $p[v/x](s_i) = e_i[v/x]$ for all s_i. Given two closed patterns p, q, we write $p \leq q$ if $p(s) \leq q(s)$ for all $s \in S$.

[1] To ease the presentation, we impose $n \in \mathbb{N}^+$ on the linear epression e to guarantee that its evaluation into a positive number, even when $x = 0$. Alternative choices are possible to relax this constraint.

Then, we extend the syntax of reactions $r = (R, I, P)$ by considering I as closed pattern, and R and P as patterns such that if P is not closed, then R is not closed. A state W is a closed pattern. At each step, starting from a given state, the semantics verifies the enabled reactions using function $en()$, computes the *multiplicity* of reaction applications (the value of x obtained by matching the current state W with pattern R) by function $mul()$, and computes the resulting state by function $res()$. Given a reaction $a = (R, I, P)$ and a state W, we define:

- the function $en(a, W)$, returns 1 if the reaction is enabled, 0 otherwise

$$en(a, W) \triangleq \begin{cases} 1 & \text{if } R[0/x] \leq W \text{ and } \forall s \in S.\ I(s) > 0 \Rightarrow W(s) < I(s) \\ 0 & \text{otherwise} \end{cases}$$

- function $mul(a, W)$ returns the value v that will correctly bind x when applied to state W

$$mul(a, W) \triangleq \begin{cases} 1 & \text{if } en(a, W) = 0 \text{ or } R \text{ is a closed pattern} \\ \max\{v \in \mathbb{N} \mid R[v/x] \leq W\} & \text{otherwise} \end{cases}$$

- function $res(a, W)$ returns the product of the application of reaction a on state W

$$res(a, W) \triangleq en(a, W) \cdot P[mul(a, W)/x]$$

Example 15. Consider again the previous example, $r = (R, I, P)$ with $R = \mathsf{a}(x + 1)$, $I = \mathsf{b}(2)$ and $P = \mathsf{c}(x + 1)$ and the state $W = \{\mathsf{a}(3), \mathsf{b}(1)\}$, we compute:

- $en(r, W) = 1$, as $R[0/x] = \mathsf{a}(1) \leq \mathsf{a}(3)$ and $W(\mathsf{b}) = 1 < 2 = I(\mathsf{b})$.
- $mul(r, W) = 2$, as $\max\{x \in \mathbb{N} \mid R[v/x] = \mathsf{a}(v + 1) \leq \mathsf{a}(3)\mathsf{b}(1) = W\} = 2$.
- $res(r, W) = en(r, W) \cdot P[mul(r, W)/x] = 1 \cdot \mathsf{c}(2 + 1) = \mathsf{c}(3)$.

Once the product of each enabled reaction has been calculated, we need to compute the next state. We consider the operator that computes the maximum between two closed patterns $p \sqcup q$, defined as $(p \sqcup q)(s) = \max\{p(s), q(s)\}$. It gives the point-wise maximum value of each entity. Analogously, to combine inhibitor constraints, we will later consider the operator that computes the minimum between two closed patterns p and q, denoted by $p \sqcap q$, defined as $(p \sqcap q)(s) = \min\{p(s), q(s)\}$.

Example 16. Assume we add a new reaction $r' = (R', I', P')$ to the previous example, where $R' = \mathsf{a}(x + 2)\mathsf{b}(1)$, $I' = \emptyset$, $P' = \mathsf{c}(3x + 2)$. By applying the function $res(r', W) = en(r', W) \cdot P[mul(r', W)/x]$ we have $1 \cdot \mathsf{c}(3x + 2)[1/x] = \mathsf{c}(5)$. Therefore, in the system composed by state W and reactions r, and r', we obtain the next state $W' = \mathsf{c}(3) \sqcup \mathsf{c}(5) = \mathsf{c}(5)$.

In the SOS style, the hypotheses under which a reaction is applied or inhibited are recorded in the label and their consistency is verified by rule (Par) and (Sys). We stretch here the fact that such hypotheses consist of constraints over concentration levels. If we assume that a reaction $a = (R, I, P)$ is enabled with multiplicity v, it means that it must be $\forall s \in I.\ W(s) < I(s)$ and $\forall s \in S.\ R[v/x](s) \leq W(s)$ but $R[v + 1/x] \not\leq W$. The first two constraints can be already represented in the ordinary labels, for the last one we extend labels with a set of bounds $c = \{R_1, ..., R_n\}$ for which we shall require that $\forall i \in [1, n].\ R_i \not\leq W$, (more concisely $c \not\leq W$) and let

$$bnd(R, v) \triangleq \begin{cases} \emptyset & \text{if } R \text{ is a closed pattern} \\ \{R[v + 1/x]\} & \text{otherwise} \end{cases}$$

Correspondingly, we update the operation to combine and to compare labels as follows:

$$\langle W_1 \rhd R_1, I_1, P_1 \rangle \otimes \langle W_2 \rhd R_2, I_2, P_2 \rangle \triangleq \langle W_1 \sqcup W_2 \rhd R_1 \sqcup R_2, I_1 \sqcap I_2, P_1 \sqcup P_2 \rangle$$

$$\langle W_1 \rhd R_1, I_1, P_1 \rangle \frown \langle W_2 \rhd R_2, I_2, P_2 \rangle \triangleq \forall s \in S.\ I(s) > 0 \Rightarrow WR(s) < I(s)$$

where $I = I_1 \sqcap I_2$ and $WR = W_1 \sqcup W_2 \sqcup R_1 \sqcup R_2$.

$$\frac{v \in \mathbb{N} \quad R_v = R[v/x] \quad P_v = P[v/x] \quad c = bnd(R,v)}{(R,I,P) \xrightarrow{c,\langle \emptyset \triangleright R_v, I, P_v \rangle} (R,I,P) \mid P_v} \ (ProS) \qquad \frac{J \leq I \quad Q \leq R[0/x] \quad J \sqcup Q \neq \emptyset}{(R,I,P) \xrightarrow{\emptyset,\langle \emptyset \triangleright J, Q, \emptyset \rangle} (R,I,P)} \ (InhS)$$

$$\frac{\mathsf{M}_1 \xrightarrow{c_1,\ell_1} \mathsf{M}_1' \quad \mathsf{M}_2 \xrightarrow{c_2,\ell_2} \mathsf{M}_2' \quad \ell_1 \frown \ell_2}{\mathsf{M}_1 \mid \mathsf{M}_2 \xrightarrow{c_1 \cup c_2, \ell_1 \otimes \ell_2} \mathsf{M}_1' \mid \mathsf{M}_2'} \ (Par) \qquad \frac{\mathsf{M} \xrightarrow{c,\langle W \triangleright R, I, P \rangle} \mathsf{M}' \quad c \not\leq W \quad R \leq W}{[\mathsf{M}] \xrightarrow{c,\langle W \triangleright R, I, P \rangle} [\mathsf{M}']} \ (Sys)$$

Fig. 3. SOS semantics for linear functions.

6 Application

In [8] we presented a preliminary implementation of RSs in a logic programming language (Prolog), mainly intended for rapid prototyping. Following the formal definition of RS with delays/durations and with concentration levels given in the previous sections, we describe how such extensions have been integrated in the prototype, available for download[2]. We remark that the modular nature of the SOS formalization simplified significantly the adaptation of the tool.

A RS is represented as a list of reactions. In the case of delays/durations, a reaction is a quintuple [R, I, P, N, M], while in the case of concentration levels it is a triple [R, I, P]. There, R, I and P represent reactants, inhibitors and products, respectively, N is the delay and M the duration. After coding the RS, a query is performed by calling either the predicate `computation(InitialState,StateSequence)`, for the case of delays/durations, or the predicate `computeFiniteComputation(InitialState,MaxStepNum,StateSequence)`, for the case of concentration levels. The result is a sequence of states starting from `InitialState` and performing at most `MaxStepNum` steps in the case of concentration levels, while in the delays/durations case the interpreter will give the user the choice to perform also a possibly infinite computation. A computation can also stop when the state gets empty. The predicate `reactionSet/1` defines the list of reactions for the case of delays/durations, while `reactionsQ/1` defines the list of reactions for the case of concentration levels and they have to be redefined for each example to be studied.

6.1 Case study: controlling the differentiation in Th-cell

The immune system is composed by various cell types, including antigen cells and B and T lymphocytes. Among the latter, T cells can be further sub-classified into T helper 1 (*Th1*) or T helper 2 (*Th2*) cells, originating from a common precursor *Th0*. A complex gene network regulates the differentiation of *Th0* cells. Studying the molecular mechanisms of this differentiation process is relevant since enhanced *Th1* and *Th2* responses may cause autoimmune and allergic diseases, respectively.

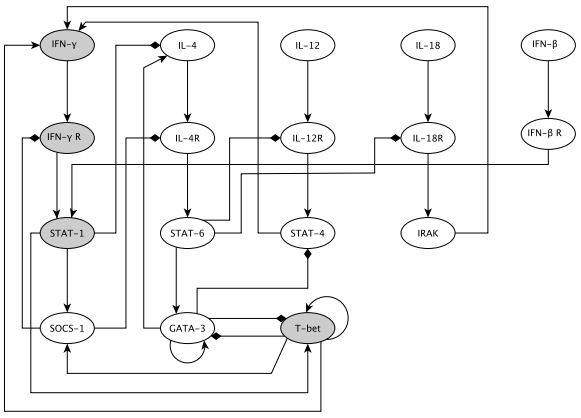

Fig. 4. Graphical representation of the Boolean network

In [16] a Boolean network model of such a regulatory process has been conceived from the large amount of molecular data available in the literature. The network includes 17 nodes regulating the differentiation of the *Th0* precursor [19,1]. The Boolean network is depicted in Fig. 4. Details about the Boolean update functions are in Fig. 5.

In [5] the authors translated the Boolean network into a closed RS (a RS without environment) that used different entities to model different levels for the gray nodes in Fig 4. The RS can reproduce the dynamics of the update functions in Fig. 5. However, this translation was not very

[2] https://www3.diism.unisi.it/~falaschi/ReactionSystemsQuantities

natural because it to introduce the artificial concept of *valid state* to avoid entites representing different levels of the same node to be present at the same time.

Here, we rephrase such a translation using the new linear functions framework presented in Section 5. We express different concentrations levels with concentration values $\{1, 2\}$, where 1 stands for medium and 2 for high. For example, the state $\{STAT\text{-}1(1),\ T\text{-}bet\ (2)\}$ states that we have a medium concentration of $STAT$-1 and a high concentration of *T-bet*.

$$
\begin{aligned}
IFN\text{-}\gamma\text{-}m(t+1) &= (STAT\text{-}4(t) \wedge \neg IRAK(t) \wedge \neg T\text{-}bet\text{-}h(t)) \vee (T\text{-}bet\text{-}m(t) \wedge (\neg STAT\text{-}4(t) \vee \neg IRAK(t))) \\
IFN\text{-}\gamma\text{-}h(t+1) &= (STAT\text{-}4(t) \wedge IRAK(t)) \vee T\text{-}bet\text{-}h(t) \\
IL\text{-}4(t+1) &= GATA\text{-}3(t) \wedge \neg STAT\text{-}1\text{-}m(t) \wedge \neg STAT\text{-}1\text{-}h(t) \\
IFN\text{-}\gamma R\text{-}m(t+1) &= IFN\text{-}\gamma(t)\text{-}m \vee (IFN\text{-}\gamma\text{-}h(t) \wedge SOCS\text{-}1(t)) \\
IFN\text{-}\gamma R\text{-}h(t+1) &= IFN\text{-}\gamma\text{-}h(t) \wedge \neg SOCS\text{-}1(t) \\
IL\text{-}4R(t+1) &= IL\text{-}4(t) \wedge \neg SOCS\text{-}1(t) \\
IL\text{-}12R(t+1) &= IL\text{-}12(t) \wedge \neg STAT\text{-}6(t) \\
IL\text{-}18R(t+1) &= IL\text{-}18(t) \wedge \neg STAT\text{-}6(t) \\
IFN\text{-}\beta R(t+1) &= IFN\text{-}\beta(t) \\
STAT\text{-}1\text{-}m(t+1) &= (IFN\text{-}\beta R(t) \wedge \neg IFN\text{-}\gamma R\text{-}h(t)) \vee IFN\text{-}\gamma R\text{-}m(t) \\
STAT\text{-}1\text{-}h(t+1) &= IFN\text{-}\gamma R\text{-}h(t) \\
GATA\text{-}3(t+1) &= (STAT\text{-}6(t) \vee GATA\text{-}3(t)) \wedge \neg(T\text{-}bet\text{-}h(t) \vee T\text{-}bet\text{-}m(t)) \\
SOCS\text{-}1(t+1) &= T\text{-}bet\text{-}m(t) \vee T\text{-}bet\text{-}h(t) \vee STAT\text{-}1\text{-}m(t) \vee STAT\text{-}1\text{-}h(t) \\
IRAK(t+1) &= IL\text{-}18R(t) \\
STAT\text{-}4(t+1) &= IL\text{-}12R(t) \wedge \neg GATA\text{-}3(t) \\
STAT\text{-}6(t+1) &= IL\text{-}4R(t) \\
T\text{-}bet\text{-}m(t+1) &= (STAT\text{-}1\text{-}m(t) \vee T\text{-}bet\text{-}m(t)) \wedge \neg(STAT\text{-}1\text{-}h(t) \vee T\text{-}bet\text{-}h(t) \vee GATA\text{-}3(t)) \\
T\text{-}bet\text{-}h(t+1) &= (STAT\text{-}1\text{-}h(t) \vee T\text{-}bet\text{-}h(t)) \wedge \neg GATA\text{-}3(t)
\end{aligned}
$$

Fig. 5. Boolean functions modelling the differentiation of Th cells

We advocate that using the linear functions framework has many advantages compared to the modelling approach adopted in [5]. For example, in [5] reaction ({$GATA$-3},{$STAT$-1-h, $STAT$-1-m },{IL-4}) was used to describe the production of IL-4, which is inhibited by $STAT$-1 in high or medium concentration. This required to include two inhibitors in the rule (distinguished by the final -h and -m, respectively). Instead, in the our new framework based on linear functions the same event can be modeled by the following simpler reaction: {$GATA$-3(1)},{$STAT$-1(1)},{IL-4(1)}). It is worth noticing that such reaction is enabled in any state containing $GATA$-3, but that does not containing $STAT$-1 at any level, as desired. Similarly, in [5] the production of $SOCS$-1 when *T-bet* at any level is present was modelled by reactions ({*T-bet*-h },{},{$SOCS$-1}) and ({*T-bet*-m},{},{$SOCS$-1}). In the linear functions framework the production of $SOCS$-1 can be expressed by the single reaction ({*T-bet*(1)},{},{$SOCS$-1(1)}). Even more interestingly, in [5] two reactions ({IFN-γR-m},{},{$STAT$-1-m}), ({IFN-γR-h},{},{$STAT$-1-h}) were introduced to express the fact that IFN-γR at some level produces $STAT$-1 at the same level. In the linear functions framework one reaction suffices: ({ IFN-γR (x+1)},{},{$STAT$-1(x+1)}).

As a result we obtain 26 reactions, available online[3], that model the system described in Fig. 5. Our prototype implementation allows us to compute the LTS. We performed two in silico experiments that show some paths leading to *Th1* differentation (note that the presence of T-bet is the marker of the differentiation of the cell into *Th1* form). In the first one, the evolution is driven by the up-regulation of IFN-γ that is expressed at the maximal level at the initial state: {IFN-γ(2)}. After 6 steps we reach the stable state {IFN-γ(2), IFN-γR(2), $STAT$-1(2), *T-bet*(2)} that shows that the differentiation towards *Th1* cell is successfully accomplished. The second experiment is driven by the initial expression of both IL-12 and IL-18. The initial state in this case is {IL-12, IL-18}, and, after 9 steps, the system reaches the stable state. Of course, all intermediate states can be inspected, and this allows us to observe that the second experiment reaches soon a high level of IFN-γ, and then the execution continues as in the first experiment.

[3] https://www3.diism.unisi.it/~falaschi/reactionsConcentrationLevels.txt

7 Conclusions and related work

The model of RSs is qualitative as there is no direct representation of the number of molecules involved in biochemical reactions as well as of rate parameters influencing the frequency of reactions. In [17] the authors introduce an extension with discrete concentrations allowing for quantitative modelling. They demonstrate that although RSs with discrete concentrations are semantically equivalent to the original qualitative RSs, they provide much more succinct representations in terms of the number of molecules being used. They then define the problem of reachability for RSs with discrete concentrations, and provide its suitable encoding in satisfiability modulo theory, together with a verification method (bounded model checking) for reachability properties. Experimental results show that verifying RSs with discrete concentrations instead of the corresponding basic RS is more efficient. A crucial feature of a RS is that (unless introduced from outside the system) an entity from the current state will belong also to the next state only if it is in the product set of an enabled reaction. In other words, an entity vanishes unless it is sustained by a reaction. In [7] it is introduced an extension where such a property is mitigated, indeed they provide each entity x with a duration $d(x)$, which guarantees that x will last through at least $d(x)$ consecutive states. The authors demonstrate that duration/decay is a result of an interaction with a "structured environment", and they also investigate fundamental properties of state sequences of reaction systems with duration". Each of these enhancements of the RS framework requires complex changes in the syntax and semantics of the original framework and they cannot easily be combined together. Our semantic framework for RSs is more flexible, since it allows us to define extensions by simply playing with the defined SOS rules. We have shown this possibility by defining extensions with reaction delays and durations in Section 4, and with concentration levels in Section 5. Also adapting our prototype tool form RS execution was made easier by the SOS formalization. It is worth noting that these and other extensions can be combined and integrated in our framework by following the same approach.

As future work we plan to exploit our framework to deepen the study of quantitative extensions of RSs. In particular, we will continue the investigation of the extensions we introduced in this paper also by evaluating their applicability to case studies of biochemical pathways and gene regulation networks. This will be done without violating the discrete and abstract nature of RSs. Moreover, the availability of a formal semantics will allow us to study and apply formal analysis techniques aimed at assessing dynamical properties of the modelled biological systems. Finally, we plan to investigate the applicability of abstract interpretation techniques [11,10] to study properties of classes of reaction systems by exploiting under- and over-approximations of current states, which is particularly convenient when quantitative information is present in the system.

References

1. Agnello, D., Lankford, C.S.R., Bream, J., Morinobu, A., Gadina, M., O'Shea, J.J., Frucht, D.M.: Cytokines and transcription factors that regulate t helper cell differentiation: New players and new insights. Journal of Clinical Immunology **23**(3), 147–161 (2003). https://doi.org/https://doi.org/10.1023/A:1023381027062
2. Azimi, S.: Steady states of constrained reaction systems. Theor. Comput. Sci. **701**(C), 20–26 (2017). https://doi.org/https://doi.org/10.1016/j.tcs.2017.03.047
3. 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
4. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Investigating dynamic causalities in reaction systems. Theor. Comput. Sci. **623**, 114–145 (2016). https://doi.org/https://doi.org/10.1016/j.tcs.2015.11.041
5. Barbuti, R., Gori, R., Milazzo, P.: Encoding boolean networks into reaction systems for investigating causal dependencies in gene regulation. Theor. Comput. Sci. (2021). https://doi.org/https://doi.org/10.1016/j.tcs.2020.07.031
6. 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/https://doi.org/10.1142/S0129054111008842
7. Brijder, R., Ehrenfeucht, A., Rozenberg, G.: Reaction systems with duration. In: Computation, Cooperation, and Life: Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday. pp. 191–202. Springer Berlin Heidelberg (2011). https://doi.org/10.1007/978-3-642-20000-7_16

8. Brodo, L., Bruni, R., Falaschi, M.: A logical and graphical framework for reaction systems. Theor. Comput. Sci. **875**, 1–27 (2021). https://doi.org/https://doi.org/10.1016/j.tcs.2021.03.024
9. 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/https://doi.org/10.1016/j.tcs.2012.04.003
10. Cousot, P.: Principles of Abstract Interpretation. MIT Press (2021)
11. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proc. ACM POPL'77. pp. 238–252. ACM (1977). https://doi.org/10.1145/512950.512973
12. Ehrenfeucht, A., Main, M.G., Rozenberg, G.: Combinatorics of life and death for reaction systems. Int. J. Found. Comput. Sci. **21**(3), 345–356 (2010). https://doi.org/10.1142/S0129054110007295
13. Ehrenfeucht, A., Main, M.G., Rozenberg, G.: Functions defined by reaction systems. Int. J. Found. Comput. Sci. **22**(1), 167–178 (2011). https://doi.org/10.1142/S0129054111007927
14. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundamenta informaticae **75**(1-4), 263–280 (2007)
15. Hillston, J.: A compositional approach to performance modelling. Ph.D. thesis, University of Edinburgh, UK (1994)
16. Mendoza, L.: A network model for the control of the differentiation process in th cells. Biosystems **84**(2), 101 – 114 (2006). https://doi.org/10.1016/j.biosystems.2005.10.004
17. Meski, A., Koutny, M., Penczek, W.: Towards quantitative verification of reaction systems. In: Amos, M., CONDON, A. (eds.) Unconventional Computation and Natural Computation. pp. 142–154. Springer International Publishing, Cham (2016). https://doi.org/https://doi.org/10.1007/978-3-319-41312-9_12
18. Milner, R.: A Calculus of Communicating Systems. Lecture Notes in Computer Science 92, Springer (1980). https://doi.org/10.1007/3-540-10235-3
19. Murphy, K.M., Reiner, S.L.: Decision making in the immune system: The lineage decisions of helper t cells. Nature Reviews Immunology **2**, 933–944 (2002). https://doi.org/https://doi.org/10.1038/nri954
20. 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
21. Pardini, G., Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tini, S.: Compositional semantics and behavioural equivalences for reaction systems with restriction. Theor. Comput. Sci. **551**, 1–21 (2014). https://doi.org/10.1016/j.tcs.2014.04.010
22. Plotkin, G.D.: An operational semantics for CSP. In: Bjørner, D. (ed.) Proceedings of the IFIP Working Conf. on Formal Description of Programming Concepts- II, Garmisch-Partenkirchen. pp. 199–226. North-Holland (1982)
23. 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
24. Villasana, M., Radunskaya, A.: A delay differential equation model for tumor growth. Journal of mathematical biology **47**(3), 270–294 (2003). https://doi.org/10.1007/s00285-003-0211-0