Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 253 (2009) 167–185
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

From Biochemistry to Stochastic Processes

Cosimo Laneve [1]

Dipartimento di Scienze dell'Informazione, Università di Bologna

Sylvain Pradalier [2]

Ecole Polytechnique, Paris

Gianluigi Zavattaro [3]

Dipartimento di Scienze dell'Informazione, Università di Bologna

Abstract

The nanoκ calculus is a formalism that models biochemical systems by defining its set of reactions. We study the implementation of nanoκ into the Stochastic Pi Machine where biochemical systems are defined by regarding molecules as processes, and deriving the overall behaviour by means of communication rules. Our implementation complies with the *stochastic behaviors* of systems, thus allowing one to use nanoκ as an intelligible front-end for a process-oriented simulator. This study also permits to reuse, in nanoκ, the theories and tools already developed for process calculi.

Keywords: Stochastic process calculi, systems biology, encoding from reactive to process-oriented formalisms.

1 Introduction

Several stochastic formalisms emerged in the last few years as models for the representation of biological systems (see e.g. [5,12,2,7,9,8,4] just to mention a few). These formalisms usually follow either a *reactive-oriented* (as [5,2,12,9,4] in the list above) or a *process-oriented* approach (as [7,14,8]). According to the former approach – inspired by traditional chemical kinetics – a system is specified as a set of reactions; according to the latter – inspired by process calculi – a system is specified by defining each molecule as a process, and deriving the overall behaviour by means of communication rules.

[1] Email:laneve@cs.unibo.it
[2] Email:sylvain.pradalier@lix.polytechnique.fr
[3] Email:zavattar@cs.unibo.it

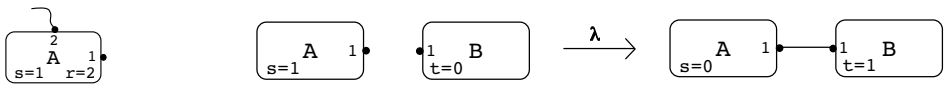
1571-0661 © 2009 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2009.10.011

Process-oriented descriptions depart from ordinary biochemical models because they define the sequences of actions once and for all and use syntaxes usually devoted to computer science. Moreover the modelling of a molecule is a term of size proportional to the number of interactions addressing the molecule. As a consequence, such descriptions are less intelligible to biochemists than reaction-oriented approaches, whose syntax is closer to biochemistry and whose complexity is spread over the reactions. On the other hand, process-oriented calculi retain several simulators and tools, which make them attractive for experiments *in silico* (see for instance [16,7,14]).

In this paper we bridge the gap between the two approaches by implementing the `nano`κ calculus [10], a reactive-oriented formalism, into the *Stochastic Pi Machine* [7], SPIM calculus in the following, a simulator for the stochastic π-calculus [19,7]. In `nano`κ calculus a molecule is a term $A[s^1 + r^2](1 + 2^x)$ with fields s and r and sites 1 and 2. The fields define the internal state of the molecule – they model its shape or its hydrogen groups or phosphate groups: in the above case, the fields s and r are set to 1 and 2, respectively; the sites are the binding capabilities of the molecule: in the above case, A is bound to another molecule with site 2, the bond is called x, and is unbound on site 1. Note that only sites can be bound and that only fields can store a value. The `nano`κ calculus retains a graphical representation – the above molecule is rendered in Figure 1(a). The dynamics of a `nano`κ calculus system is defined by reactions that describe how two reactants may evolve. For example, the reaction

$$\rho_1 \qquad A[s^1](1), B[t^0](1) \xrightarrow{\lambda} A[s^0](1^x), B[t^1](1^x)$$

illustrated in Figure 1(b), specifies that every molecule A with an internal state s



(a) (b)

Fig. 1. Molecules and reactions in `nano`κ calculus

equal 1 and a free site 1 may react with every B with internal state t equal 0 and free site 1. The result is a complex where A and B are connected by a bond, called x, and the two internal states have swapped values. This change in the internal state might represent for instance an exchange of electronical charge. The label λ of the reaction represents its rate. It is worth to notice that this reaction applies to the molecule A in Figure 1(a), as well as to every other A with a different value of r and/or with an unbound site 2.

In `nano`κ molecules may react by means of three types of reactions – creations, destructions, and exchanges – and retain a stochastic semantics. The above reaction ρ_1 is an example of creations. The reaction

$$\rho_2 \qquad A[s^1](1^x), B[t^0](1^x) \xrightarrow{\lambda'} A[s^0](1), B[t^1](1)$$

defines a destruction that is opposite to the above creation. The rate λ' may be different from λ, thus yielding *equilibria* in accordance with them. The reaction

$$\rho_3 \qquad A(2^z), C(1) \stackrel{\lambda''}{\rightarrowtriangle} A(2), C(1^z)$$

is an exchange rule defining a bond flipping from the site 2 of A to 1 of C. Bond flipping is of a peculiar interest to model nano-machines where links between components are dynamically rearranged [10]. Despite the simplicity of these reactions, their implementation in SPIM calculus is complex if the stochastic semantics must be preserved. Let us discuss the problems through a number of examples.

An implementation of nanoκ into SPIM should project the behaviour of each molecule out of the set of reactions and collect them into a process definition. For example, the SPIM process $\widehat{\mathtt{A}}$ of the molecule A in the above examples is

$$\begin{aligned}\widehat{\mathtt{A}}(s,t,a_1,a_2) \;=\; & \texttt{behaviour-of A in } \rho_1 \\ & + \texttt{behaviour-of A in } \rho_2 \\ & + \texttt{behaviour-of A in } \rho_3\end{aligned}$$

That is, a molecule is implemented by a parametric process definition, where the parameters s, t, a_1, a_2 define the values of fields and sites. The "`behaviour-of A in` ρ_1" might be defined as

$$[a_1 = \varepsilon, s = 1]\, \overline{\rho_1}\,(x).\widehat{\mathtt{A}}(0,t,x,a_2)$$

where

- $[a_1 = \varepsilon, s = 1]$ means that such a behaviour may be triggered provided the site a_1 is unbound (has value ε) and the field s is equal 1;
- in this case the channel ρ_1 is used to output a fresh name (modelling the bond). We expect that $\widehat{\mathtt{B}}$ will perform a corresponding input when the field r is 1 and the site 1 is unbound. We also expect that the rate of the channel ρ_1 has been declared to be λ;
- then $\widehat{\mathtt{A}}$ will continue as the process $\widehat{\mathtt{A}}(0,t,x,a_2)$.

However the analogy *reaction-names as channels* cannot be pushed forward to destruction. In facts, if the "`behaviour-of A in` ρ_2" was modelled as

$$[a_1 = \neg\varepsilon, s = 0]\, \overline{\rho_2}\,().\widehat{\mathtt{A}}(1,t,\varepsilon,a_2)$$

then $\widehat{\mathtt{A}}$ might interact with the wrong $\widehat{\mathtt{B}}$ in the SPIM implementation ($a_1 = \neg\epsilon$ means that a_1 is bound). This difficulty may be circumvented by using the names encoding bonds – that are shared exactly by the two connected molecules – in order to send the disconnection signal. So the "`behaviour-of A in` ρ_2" becomes

$$[a_1 = \neg\varepsilon, s = 0]\, \overline{a_1}\,().\widehat{\mathtt{A}}(1,t,\varepsilon,a_2)$$

and we are assuming that $\widehat{\mathtt{B}}$ will input on a_1 and that the rate of this name is λ'.

Unfortunately, this solution is also defective. Consider the rule

$$\rho_4 \qquad A(1^x + 2^z), B(1^x + 2) \stackrel{\lambda'''}{\rightarrow} A(1^x + 2), B(1^x + 2^z)$$

with $\lambda''' \neq \lambda'$. In this case, a bond may flip from A to B (with an exchange rule) *provided they are connected* through the site 1. A simple solution is to implement this rule with exactly one SPIM interaction by using the channel x for exchanging the bond z. But the mismatch between the rates λ' and λ''' makes this solution (stochastically) unfeasible. However, this last difficulty may be overcome by introducing, at creation time, a tuple of channels, one for every use. So in case of a destruction, a channel of the tuple is used (and the whole tuple is destroyed), in case of exchange, another one is used, which retains a different rate. Said otherwise, a bond in nanoκ calculus corresponds to a tuple of channels in SPIM with possibly different rates.

In facts, the above one is the implementation we describe in this contribution. In an accurate solution, a bond in nanoκ should be represented in SPIM by a tuple whose length is the number of reactions that address that bond directly or indirectly through sequences of exchanges. Actually, for simplicity, our solution over-approximates the "precise" solution, by representing bonds with tuples whose length is the size of the set of reactions – the *gangs*, in our terminology.

The encoding of nanoκ into SPIM defined in this paper, let it be $[\![\cdot]\!]$, is such that $\mathsf{S} \stackrel{\lambda}{\longmapsto}_{nanok} \mathsf{T}$ if and only if $[\![\mathsf{S}]\!] \stackrel{\lambda}{\longmapsto}_{spim} [\![\mathsf{T}]\!]$ (the arrow is subscribed to ease the reading). It follows that S and $[\![\mathsf{S}]\!]$ are *strongly stochastic bisimilar* [3]. This is different from usual implementations that almost never preserve the granularity of transitions. Of course, this strong relationship may be weakened by defining suitable protocols, in the style of [13]. However such a solution might hardly preserve the stochastic semantics. In facts, the stochastic semantics defines an exponential law controlling the waiting time before a transition can be fired (the *sojourn time*). In turn, the match of one reaction with a sequence of transitions amounts to matching an exponential distribution with a sum of exponential distributions, which is not possible.

Related works.

In [20], it has been shown that systems of molecular interactions with explicit bonds might be represented and simulated using the stochastic π-calculus. Our encoding corroborates this result since the SPIM calculus is a subset of the stochastic π-calculus. We remark that the example provided in [20] and, we believe, the descriptions done in this approach, can easily be rewritten in SPIM calculus and even in a sub-calculus of it, since our encodings doesn't use its full power.

In [6], Cardelli has encoded chemical systems into process algebra and back preserving both the stochastic and the ODE semantics. Our encoding extends these encodings because the CGF process algebra used in [6] is a subset of the SPIM calculus and because the nanoκ calculus extends the language of chemical reactions of [6] with explicit bonds between molecules and with internal states. However,

our results are weaker than those in [6], since we only assert the correctness of the encoding with respect to the stochastic semantics.

Another stochastic process calculus that has been used also for the modeling of biochemical systems is PEPA [14]. For instance, in [4], PEPA has been exploited to examine the influence of the RAF Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase (ERK) signalling pathway. Also in that paper, as in the present one, a reagent-centric view and a pathway-centric (process-centric in our terminology) view are studied. Our analysis of the two approaches is different for two main reasons. First of all, in the PEPA-based approach one process is used to represent the concentration of one species while we follow the Cardelli's approach considering one process for each molecule. In fact, we have found this approach appropriate for a compositional model of discrete state systems (in which we count the number of molecules instead of considering their concentrations). The second difference is that in [4] only finitely many different species are considered, thus the translation form the reagent-centric to the pathway-centric views can be obtained using an intermediate matrix representation that quantifies the impact of each reaction on each reagent in a manner analogous to the stoichiometry matrix of the chemical reactions. We cannot exploit this approach as we do not impose any bound to the number of different complexes that can be produced in a nanoκ calculus system. More recently, the PEPA approach has been also extended with a reaction-centric model called Bio-PEPA [9]. Also in this case, one process is used to represent the concentration of one species.

Encodings from the full κ calculus to nanoκ calculus, or to π-calculus are presented in [13] and [11]. Yet, they only preserve non-stochastic semantics. In facts, encodings preserving the stochastic semantics do not exist, due to the negative results of [17].

Structure of the article.

The rest of the article is organized as follows. First we recall the syntax of nanoκ and SPIM, and present their basic stochastic semantics. In part 3, we first introduce the gangs and then complete the definition of the encoding. Its correctness is asserted by theorem 3.2 and 3.4. In part 4 we present the collective stochastic semantics and state the correctness of the encoding with respect to the collective stochastic semantics in theorem 4.3.

2 The stochastic calculi

We shortly present the two stochastic calculi we analyze in this paper: a subcalculus of nanoκ calculus, where reactants share at most one bond, and a subcalculus of SPIM. Examples and additional details can be found in [10] and [7].

2.1 The nanoκ *calculus*

Terms, called *solutions*, are sequences of *molecules*. Each molecule belongs to a *species* and retain an *internal state*, which is determined by a tuple of *fields*, and

an *interface*, which is a tuple of *sites* that may be bound to other sites. Formally, a molecule will be written $A[u](\sigma)$, where

- A is the species. The molecules of a species retain the same set of fields and the same set of sites that are finitely many; fields and sites will be addressed by numbers 0, 1, 2, $\cdots$;
- u – called the *evaluation* – is a total map from fields of A to *finite sets* (the internal states of molecules are always finitely many);
- σ – called the *interface* – is a total map from sites of A to either *bonds*, which are names of a *totally ordered countable set* ranged over by x, y, z, $\cdots$, or ε, a special value indicating that the site is not bound.

For example, $A[1 \mapsto 0; 2 \mapsto 1](1 \mapsto \varepsilon; 2 \mapsto x; 3 \mapsto \varepsilon)$ is a molecule with two fields 1 and 2 and three sites 1, 2, and 3. The fields 1 and 2 have values 0 and 1, respectively; the site 2 is the only one that is bound and the bond is x. In order to ease the reading, we write this molecule as $A[1^0 + 2^1](1 + 2^x + 3)$ (the value ε is always omitted). Let $\emptyset$ be the empty map. We write $A(\sigma)$ instead of $A[\emptyset](\sigma)$, $A[u]$ instead of $A[u](\emptyset)$, and simply A instead of $A[\emptyset](\emptyset)$. We denote by $\mathrm{ran}(\sigma)$ the range of an interface σ deprived of ε and by $\mathit{bonds}(\mathsf{S})$ the set of the bonds appearing in the solution S.

Definition 2.1 A *solution* is a term defined by the grammar

$$\mathsf{S} \quad ::= A[u](\sigma) \;\mid\; \mathsf{S},\mathsf{S}$$

The operator "$,$" is assumed to be associative, so $(\mathsf{S},\mathsf{T}),\mathsf{R}$ is equal to $\mathsf{S},(\mathsf{T},\mathsf{R})$ (therefore parentheses are always omitted).

Bonds always occur at most twice in solutions. A solution is *proper* if every bond therein occurs exactly twice.

The nanoκ calculus semantics is defined by means of reaction rules. A few preliminary definitions are in order:

- we write $\sigma \leq \sigma'$ if $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$ and, for every i, if $\sigma(i) \neq \varepsilon$ then $\sigma(i) = \sigma'(i)$ (the two interfaces may differ on sites mapped to the empty value ε by σ: σ' may map such sites to bonds);
- a *pre-solution* is a sequence of terms $A[u](\sigma)$ where u and σ are partial functions (with an abuse of notation, we denote partial and total functions in the same way);
- a pre-solution is *proper* when every bond therein occurs exactly twice.

In the following when we write $u+u'$ and $\sigma+\sigma'$ we assume that $\mathrm{dom}(u) \cap \mathrm{dom}(u') = \emptyset$ and $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\sigma') = \emptyset$.

Definition 2.2 Reactions of nanoκ calculus are either *creations*, *destructions*, or *exchanges* that are labelled by *rates*, which are positive real numbers or ∞. Cre-

ations have format

$$A[u](\sigma)\,,B[v](\rho) \stackrel{\lambda}{\rightarrow} A[u'](\sigma')\,,B[v'](\rho')\,,C_1[w_1](\eta_1)\,,\cdots,C_n[w_n](\eta_n)$$

where $\sigma \leq \sigma'$, $\rho \leq \rho'$, $\mathrm{dom}(u) = \mathrm{dom}(u')$, $\mathrm{dom}(v) = \mathrm{dom}(v')$, and w_i and η_i are total. Destructions have formats

$$A[u](\sigma)\,,B[v](\rho) \stackrel{\lambda}{\rightarrow} A[u'](\sigma')\,,B[v'](\rho')$$
$$A[u](\sigma)\,,B[v](\rho) \stackrel{\lambda}{\rightarrow} A[u'](\sigma')$$

where $\sigma \geq \sigma'$, $\mathrm{dom}(u) = \mathrm{dom}(u')$, and, in the first case, $\rho \geq \rho'$, $\mathrm{dom}(v) = \mathrm{dom}(v')$ and, in the second case, ρ has to be total. Exchanges have one of the formats:

$$A[u](\sigma)\,,B[v](\rho) \stackrel{\lambda}{\rightarrow} A[u'](\sigma)\,,B[v'](\rho)$$
$$A[u](a^x+\sigma)\,,B[v](b+\rho) \stackrel{\lambda}{\rightarrow} A[u'](a+\sigma)\,,B[v'](b^x+\rho)$$

where the pre-solutions $A[u](\sigma)\,,B[v](\rho)$ and $A[u](a+\sigma)\,,B[v](b+\rho)$ are proper and $\mathrm{dom}(u) = \mathrm{dom}(u')$ and $\mathrm{dom}(v) = \mathrm{dom}(v')$.

In the rest of the paper we assume that reactants share at most one bond, i.e. $\mathrm{ran}(\sigma) \cap \mathrm{ran}(\rho)$ is either an empty set or a singleton.

Creations produce new bonds between two unbound sites and/or synthesize new molecules. Destructions behave in the other way around. Exchanges either leave the interfaces unchanged or move one bond from a reactant to the other (bond-flipping exchange).

It is worthwhile to remark that reactions do not address every field and site of the reactants (evaluations and interfaces are partial). The intended meaning is that two molecules reacts if they are *instances* of the left-hand side of a reaction. We will formalize this notion later on in the section.

2.2 The SPIM *calculus*

The SPIM calculus uses two sets of identifiers: *names*, which is totally ordered and ranged over by x, y, u, $\cdots$, *agents*, ranged over by A, B, $\cdots$. Names have a rate that is a positive real number or ∞. This rate may be explicitly declared in the process or globally defined (for free names). The following syntactic categories are used in SPIM calculus:

$$\begin{array}{rcll} M & ::= & [u=v] \;\mid\; M\,M & \text{matches} \\ \alpha & ::= & x\,(\widetilde{u}) \;\mid\; \overline{x}\,\widetilde{u} \;\mid\; \overline{x}(\widetilde{u}:\widetilde{\lambda}) & \text{actions} \\ P & ::= & \mathbf{0} \;\mid\; A(\widetilde{u})|P & \text{terms} \end{array}$$

Matches are sequences of equalities between values. Actions are either *input* $x\,(\widetilde{u})$ on x of a tuple $\widetilde{u}$, or *output* $\overline{x}\,\widetilde{u}$ on x of a tuple $\widetilde{u}$, or *bound output* $\overline{x}\,(\widetilde{u}:\widetilde{\lambda})$ on

x of a tuple $\widetilde{u}$ with rates $\widetilde{\lambda}$. Terms can be the inert $\mathbf{0}$ or a parallel composition of agent invocations. The parallel operator | is assumed to be associative. Agent declarations have the form:

$$A(\widetilde{x}) \triangleq \sum_{i \in I} M_i \alpha_i . P_i$$

Notation.

Whenever a match has the form $[u = u]$, or a sum has only one branch we omit to write them explicitly. For instance $A(\widetilde{x}) \triangleq \sum_{i \in \{1\}} [\widetilde{x}_i = \widetilde{x}_i]\alpha.P$ is written $A(\widetilde{x}) \triangleq \alpha.P$.

A process is a term $(\widetilde{x} : \widetilde{\lambda})\ P$ – the set of agent definitions is kept implicit – where $\widetilde{\lambda}$ are rates. The term $\widetilde{x} : \widetilde{\lambda}$ has to be considered a set with the constraint that every two different elements have different names. Processes are ranged over by P, Q, $\cdots$.

Scope restrictions bind names, that is in $(x : \lambda)\ P$ the x free in P is bound by $x : \lambda$. Likewise, input $x\,(\widetilde{u}).P$ and bound output $\overline{x}\,(\widetilde{u} : \widetilde{\lambda}).P$ bind $\widetilde{u}$ with scope P. The agent definition $A(\widetilde{u}) \triangleq \sum_{i \in I} M_i \alpha_i . P_i$ binds $\widetilde{u}$ with scope to the right hand side of the definition. Names that are not bound are called *free* and we write $\mathtt{fn}(T)$ for the set of such names in T.

We assume that all terms meet the following well formed properties:

- in $(\widetilde{x} : \widetilde{\lambda})P$, $\widetilde{x} \subseteq \mathtt{fn}(P)$ (there is no garbage);
- bound names in agent definitions never clash with free names (this allows us to avoid alpha-conversions).

The reductions of SPIM calculus are communications on a channel. Since they are fixed, they will not play any relevant role in the following Definition 2.4. (They will be embodied in the *(init)* item of the definition.) Therefore we omit the formal definition here.

2.3 Basic transition relations

Reactions only define the (biochemical) changes of the reactants. These descriptions are used to infer transitions of solutions consisting of several possible reactants. Such transition relations are given in two steps: a first one, called *basic transition relation*, that records the position of the reactants in the whole solution; a second one, called *collective transition relation*, that computes the rate of a transition by summing the rate of the basic transitions that produce the same solution (regardless the position of the molecules/agents). Below we define the basic transition relation for $\mathtt{nano}\kappa$ and SPIM calculi. (The two definitions are very close, this is why they have been collected in this subsection.) Our result of correctness of the encoding of $\mathtt{nano}\kappa$ in SPIM regards the basic transition relation It follows that this correctness also holds for the collective semantics, since it is derived in the same way from the basic transition relation (see theorem 4.3 in Section 4).

The definition of the basic transition relation of the $\mathtt{nano}\kappa$ calculus requires few

notations. Let μ range over ρ_L and ρ_R and let $\overline{\rho_L} = \rho_R$ and $\overline{\rho_R} = \rho_L$ (notice that $\overline{\overline{\mu}} = \mu$). The nano$\kappa$ reactions may be addressed by:

$$A[u](\sigma)\,,B[v](\rho) \stackrel{\lambda}{\rightarrow} A[u'](\sigma')\,,\mathsf{S}$$

where S may also be empty (denoted by $\_$). The special term $\_$ is considered a unit for the "," operator (the solutions $\_,\mathsf{S}$, $\mathsf{S},\_$ and S are equal). With an abuse of notation we lift a renaming $\imath$ to a solution by applying it pointwise.

Definition 2.3 The basic transition relation of nanoκ, written either $\stackrel{\rho,\imath}{\longrightarrow}_{l,l'}$ or $\stackrel{\mu,\imath}{\longrightarrow}_{l}$, is the least relation that satisfies the following rules:

- (init) let $\rho = \ A[u](\sigma)\,,B[v](\phi) \stackrel{\lambda}{\rightarrow} A[u'](\sigma')\,,\mathsf{S}$. Then both $A[u+w](\sigma\circ\imath+\nu) \stackrel{\rho_L,\imath}{\longrightarrow}_1 A[u'+w](\sigma'\circ\imath+\nu)$ and $B[v+w](\phi\circ\imath+\nu) \stackrel{\rho_R,\imath}{\longrightarrow}_1 \mathsf{T}$, where T is either $B[v'+w](\phi'\circ\imath+\nu)\,,\imath(\mathsf{S})$ or $\imath(\mathsf{S})$, according to the shape of the right hand side, and $\imath$ is an injective renaming with $\mathrm{ran}(\imath)\cap\mathrm{ran}(\nu)=\emptyset$;
- (lifts) if $\mathsf{S} \stackrel{\mu,\imath}{\longrightarrow}_l \mathsf{S}'$ and $(\mathit{bonds}(\mathsf{S}')\setminus\mathit{bonds}(\mathsf{S}))\cap\mathit{bonds}(\mathsf{T})=\emptyset$, then both $\mathsf{S}\,,\mathsf{T} \stackrel{\mu,\imath}{\longrightarrow}_l \mathsf{S}'\,,\mathsf{T}$ and $\mathsf{T}\,,\mathsf{S} \stackrel{\mu,\imath}{\longrightarrow}_{l'+l} \mathsf{T}\,,\mathsf{S}'$, where T has l' molecules;
- (communications) if $\mathsf{S} \stackrel{\mu,\imath}{\longrightarrow}_l \mathsf{S}'$ and $\mathsf{T} \stackrel{\overline{\mu},\imath}{\longrightarrow}_{l'} \mathsf{T}'$ and $\imath$ is an order-preserving injection that map bonds into the least ones not used in S,T then $\mathsf{S},\mathsf{T} \stackrel{\rho}{\longrightarrow}_{l,l''+l'} \mathsf{S}',\mathsf{T}'$, where ρ is the rule of μ and S has l'' molecules.

The indexes of the basic transition relation identify the position of the reactants since solutions are sequences of molecules. In the case *(init)*, the position is always 1 because the solution consists of one molecule. In the case of *(lifts)*, the index is increased by the number of the molecules on the left, if any. The last case models a reaction: the solution is split into two parts S and T containing the reactants at positions l and l', respectively. In the composite solution S,T, the reactants are at position l and $l''+l'$, where l'' is the number of molecules of S. For example let $k\mathsf{M}$ be $\underbrace{\mathsf{M},\cdots,\mathsf{M}}_{k\ times}$ and let $\rho : H(1),H(1) \stackrel{\lambda}{\rightarrow} H(1^u),H(1^u)$ be the hydrogen gas reaction. Then the following three transitions are possible

$$3H(1) \stackrel{\rho}{\longrightarrow}_{1,2} 2H(1^x),H(1)$$
$$3H(1) \stackrel{\rho}{\longrightarrow}_{1,3} H(1^x),H(1),H(1^x)$$
$$3H(1) \stackrel{\rho}{\longrightarrow}_{2,3} H(1),2H(1^x)$$

The basic transition relation is labelled by finite injective renamings. To clarify this point, consider the creation $\varrho = \ Na(1^x+2),Na(1^x+2) \stackrel{10}{\rightarrow} Na(1^x+2^y),Na(1^x+2^y)$ (a bond is created between two sodium molecules provided they are already bound). Then take the solution $Na[ion^0](1^z+2),Na[ion^0](1^v+2),Na[ion^1](1^z+2)$,

$Na[ion^0](1^v+2)$. We derive the expected transition

$$Na[ion^0](1^z+2), Na[ion^0](1^v+2), Na[ion^1](1^z+2), Na[ion^0](1^v+2)$$
$$\xrightarrow{\varrho}_{1,3} Na[ion^0](1^z+2^w), Na[ion^0](1^v+2), Na[ion^1](1^z+2^w),\ Na[ion^0](1^v+2)$$

following a structured operational semantics approach [18]. Namely, we focus on the single reactants and lift the transitions to ","-contexts. This is correct inasmuch as one records the instantiation of bonds in the left-hand sides of reactions with the actual names of the molecules: the two reactants must instantiate bonds in the same way. This is the reason why the first two molecules of the above solution cannot react with ϱ. More precisely, $Na[ion^0](1^z+2) \xrightarrow{\varrho_{L,\imath}}_1 Na[ion^0](1^z+2^w)$, where $\imath = [x \mapsto z, y \mapsto w]$, and $Na[ion^0](1^v+2) \xrightarrow{\varrho_{R,\imath}}\!\!\!\!\!\!/\ \ _1$.

Our final remarks regard the rule *(communications)*. There are possibly infinitely many transitions $\mathsf{S} \xrightarrow{\rho,\imath}_l \mathsf{T}$ because there are infinitely many renamings $\imath$ which satisfy the conditions of the (*init*) rule. However this nondeterminism is removed when the reaction occurs because the created bonds have to be the least names not occurring in S, and because the renaming has to be order-preserving. Said otherwise, the relation $\xrightarrow{\mu}_{l,l'}$, which models the evolution of a solution, is finitely branching, while the auxiliary relation $\xrightarrow{\mu}_l$ is not finitely branching. It is also worth to notice that there is no rule lifting a transition $\xrightarrow{\mu}_{l,l'}$ to a context ",": we use the associativity of , to partition a solution S into S',S'' such that the reactants are in S' and S''.

The basic transition relation of the SPIM calculus requires few definitions:

- M *is true* if M is a sequence of $[x = x]$;
- $length(A_1(\widetilde{u_1}) \mid \cdots \mid A_n(\widetilde{u_n}))$ returns n;
- $\widetilde{x} : \widetilde{\lambda} + \widetilde{y} : \widetilde{\lambda'}$ is the sequence $z_1 : \lambda_1, \cdots, z_n : \lambda_n$ where $z_1, \cdots, z_n$ are pairwise different names, $\{z_1, \cdots, z_n\} = \widetilde{x} \cup \widetilde{y}$, and $z_i : \lambda_i$ if either $z_i : \lambda_i \in \widetilde{y} : \widetilde{\lambda'}$ or $z_i \notin \widetilde{y}$ and $z_i : \lambda_i \in \widetilde{x} : \widetilde{\lambda}$.
- $[(\widetilde{x} : \widetilde{\lambda})P]_{\mathtt{GC}} = (\widetilde{z} : \widetilde{\lambda'})P$ such that $y : \lambda''$ is in $\widetilde{z} : \widetilde{\lambda'}$ if $y \in \mathtt{fn}(P)$ and $y : \lambda''$ is in $\widetilde{x} : \widetilde{\lambda}$.
- with an abuse of notation we lift a renaming $\imath$ to a tuple of names or to a process by applying it pointwise.

Let $\mathtt{bn}(\alpha)$ be $\widetilde{u}$ if μ is either $x\,(\widetilde{u})$ or $\overline{x}\,(\widetilde{u} : \widetilde{\lambda})$; it is $\emptyset$ if $\alpha = \overline{x}\,\widetilde{u}$.

Definition 2.4 The basic transition relation of the SPIM calculus, written either $\xrightarrow{\alpha}_{l.i,l'.j}$ or $\xrightarrow{\tau_\lambda}_{l.i,l'.j}$ or $\xrightarrow{\alpha}_{l.i}$, is the least one satisfying the following rules:

- (init) let $A(\widetilde{u}) = \sum_{i\in I} M_i\alpha_i.P_i$ and let $M_j\{\widetilde{v}/\widetilde{u}\}$ be true. If $\alpha_j\{\widetilde{v}/\widetilde{u}\} = \overline{x}\,\widetilde{w}$ then $A(\widetilde{v}) \xrightarrow{\overline{x}\,\widetilde{w}}_{1.j} P_j$; if $\alpha_j\{\widetilde{v}/\widetilde{u}\} = \overline{x}\,(\widetilde{w} : \widetilde{\lambda})$ then $A(\widetilde{v}) \xrightarrow{\overline{x}\,(\imath(\widetilde{w}):\widetilde{\lambda})}_{1.j} \imath(P_j)$; if $\alpha_j\{\widetilde{v}/\widetilde{u}\} = x\,(\widetilde{w})$ then $A(\widetilde{v}) \xrightarrow{x\,(\imath(\widetilde{w}))}_{1.j} \imath(P_j)$, where $\imath$ is an injective order-preserving renaming;
- (lifts) if $P \xrightarrow{\alpha}_{l.i} P'$ and $\mathtt{bn}(\alpha) \cap \mathtt{fn}(Q) = \emptyset$ and $l' = length(Q)$, then both

$P \mid Q \xrightarrow{\alpha}_{l.i} P' \mid Q$ and $Q \mid P \xrightarrow{\alpha}_{l'+l.i} Q \mid P'$;

- (communications) let $l'' = \mathit{length}(P)$, λ be the rate of x, and $Q \xrightarrow{x\,(\widetilde{u})}_{l'.i'} Q'$. If $P \xrightarrow{\overline{x}\,\widetilde{v}}_{l.i} P'$ then $(\widetilde{z} : \widetilde{\lambda'})(P \mid Q) \xrightarrow{\tau_\lambda}_{l.i,l'+l''.i'} [(\widetilde{z} : \widetilde{\lambda'})(P \mid Q\{\widetilde{v}/\widetilde{u}\})]_{GC}$; if $P \xrightarrow{\overline{x}\,(\widetilde{v}:\widetilde{\lambda''})}_{l.i} P'$ then $(\widetilde{z} : \widetilde{\lambda'})(P \mid Q) \xrightarrow{\tau_\lambda}_{l.i,l'+l''.i'} [(\widetilde{z} : \widetilde{\lambda'} + \widetilde{v} : \widetilde{\lambda''})(P' \mid Q'\{\widetilde{v}/\widetilde{u}\})]_{\mathtt{GC}}$ where $\widetilde{v}$ are the least names not occurring in $P \mid Q$. Symmetrically when P performs an input and Q performs an output.

As for nanoκ, there is always at most one $(\widetilde{z} : \widetilde{\lambda'})P'$ such that $(\widetilde{x} : \widetilde{\lambda})P \xrightarrow{\tau_{\lambda''}}_{l,i,l',i'} (\widetilde{z} : \widetilde{\lambda'})P'$ because alpha-conversion is never considered in the basic transition relation, because created names are the least possible ones and because the renamings are order-preserving.

3 Encoding the nanoκ calculus into the SPIM calculus

The definition of the encoding of nanoκ calculus into SPIM calculus is presented in two steps. The first one defines an internal translation of nanoκ calculus that expands every bond into tuples of bonds. The bonds in the tuple are an over-approximation of the reactions that use the bond. We call these tuples of newly generated names *gangs*. The second step defines a translation from nanoκ (with gangs) to the SPIM calculus.

An example illustrating the encoding is postponed to the appendix.

3.1 Gangs: a dedicated name for every reaction

In the following we use tuples that will be ordered as follows: $(x_1, \ldots, x_m) \leq (y_1, \ldots, y_m)$ if and only if, for every i, $x_i \leq y_i$. Let ε^m be a tuple of m elements ε.

Definition 3.1 Let $\mathcal{R} = \{\rho_i : \mathsf{L}_i \overset{\lambda_i}{\rightarrow} \mathsf{R}_i \mid i \in 1..n\}$ be a set of nanoκ reaction rules and let $\jmath$ be a bijective function that maps ε to ε^n and bonds to n-tuples of bonds such that if $x \leq y$ then $\jmath(x) \leq \jmath(y)$ (such a $\jmath$ exists because the set of names is countable).

The solution $[\![\mathsf{S}]\!]_\jmath$ is S where every z being either a bond or ε is replaced by $\jmath(z)$. The set of reactions $[\![\mathcal{R}]\!]_\jmath$ is $\{\rho_i : [\![\mathsf{L}_i]\!]_\jmath \overset{\lambda_i}{\rightarrow} [\![\mathsf{R}_i]\!]_\jmath \mid i \in 1..n\}$.

Namely $[\![\mathcal{R}]\!]_\jmath$ and $[\![\mathsf{S}]\!]_\jmath$ are such that

- interfaces map sites to tuples of bonds of length n – a *gang*;
- two distinct tuples do not contain the same name;
- tuples preserve the order of bonds in $\mathcal{R}$ and S.

We let $[\![\imath]\!]_\jmath = \jmath \circ \imath \circ \jmath^{-1}$ and $[\![\mu]\!]_\jmath$ be either $\rho_L, [\![\imath]\!]_\jmath$ or $\rho_R, [\![\imath]\!]_\jmath$, depending on wether μ is $\rho_L, \imath$ or $\rho_R, \imath$. The correctness of the encoding of Definition 3.1 is stated in the following theorem.

Theorem 3.2 (i) *if* $\mathsf{S} \xrightarrow{\mu}_l \mathsf{T}$ *then* $[\![\mathsf{S}]\!]_\jmath \xrightarrow{[\![\mu]\!]_\jmath}_l [\![\mathsf{T}]\!]_\jmath$ *(similarly for* $\mathsf{S} \xrightarrow{\rho}_{l,l'} \mathsf{T}$*);*

(ii) *if* $[\![\mathsf{S}]\!]_\jmath \xrightarrow{\mu}_l \mathsf{T}$ *then there exists* S' *and* μ' *such that:* $[\![\mathsf{S}']\!]_\jmath = \mathsf{T}$, $\mathsf{S} \xrightarrow{\mu'}_l \mathsf{S}'$, *and* $\mu = [\![\mu']\!]_\jmath$ *(similarly for* $[\![\mathsf{S}]\!]_\jmath \xrightarrow{\rho}_{l,l'} \mathsf{T}$*).*

Proof. We first consider the transitions $\xrightarrow{\mu}_l$ and then we lift the results to $\xrightarrow{\rho}_{l,l'}$.

We consider $\mathsf{S} \xrightarrow{\mu}_l \mathsf{T}$. We prove that the first item holds by induction on $\mathit{length}(\mathsf{S})$. If $l = 1$, the transition has been obtained by the (*init*) rule of the nanoκ calculus, and we remark that $[\![\mathsf{S}]\!]$ and $[\![\imath]\!]$ fit the requirements of the (*init*) rule of the SPIM calculus thus $[\![\mathsf{S}]\!] \xrightarrow{[\![\mu]\!]}_l [\![\mathsf{T}]\!]$. If $l > 1$, the transition has been obtained by the (*lift*) rule of the nanoκ calculus and $\mathsf{S} = \mathsf{S}_1, \mathsf{S}_2$ with either $\mathsf{S}_1 \xrightarrow{\mu}_l \mathsf{T}_1$ (and $\mathsf{T} = \mathsf{T}_1, \mathsf{S}_2$) or $\mathsf{S}_2 \xrightarrow{\mu}_{l-\mathit{length}(\mathsf{S}_1)} \mathsf{T}_2$ (and $\mathsf{T} = \mathsf{S}_1, \mathsf{T}_2$). By induction hypothesis we have that either $[\![\mathsf{S}_1]\!] \xrightarrow{[\![\mu]\!]}_l [\![\mathsf{T}_1]\!]$ or $[\![\mathsf{S}_2]\!] \xrightarrow{[\![\mu]\!]}_{l-\mathit{length}(\mathsf{S}_1)} [\![\mathsf{T}_2]\!]$. By application of the (*lift*) rule of the SPIM calculus we have that $[\![\mathsf{S}]\!] \xrightarrow{[\![\mu]\!]}_l [\![\mathsf{T}]\!]$.

The second item for the case $[\![\mathsf{S}]\!] \xrightarrow{\mu}_l \mathsf{T}$ is proved similarly.

We now consider the first item for the case $\mathsf{S} \xrightarrow{\rho}_{l,l'} \mathsf{T}$. This transition is obtained by application of the (*communications*) rule, thus $\mathsf{S} = \mathsf{S}_1, \mathsf{S}_2$ where S_1 and S_2 performs two complementary transitions of the form $\xrightarrow{\mu}_l$. As we have already proved that the theorem holds for such transitions, we have that also $[\![\mathsf{S}_1]\!]$ and $[\![\mathsf{S}_2]\!]$ can perform complementary transitions. By application of the (*communications*) rule it is easy to see that also $[\![\mathsf{S}]\!] \xrightarrow{[\![\rho]\!]}_{l,l'} [\![\mathsf{T}]\!]$.

The second item for the case $[\![\mathsf{S}]\!] \xrightarrow{\rho}_{l,l'} \mathsf{T}$ is proved similarly. □

3.2 From gangs to the SPIM *calculus: agents as molecules*

The second step of our translation encodes the nanoκ calculus with gangs of bonds into processes of SPIM. As discussed in the Introduction, we encode a species A by a parametric agent definition $\widehat{\mathtt{A}}(\widetilde{x}) = P$, whose parameters $\widetilde{x}$ represent the possible values of fields and sites of the molecules of that species. The body P is a choice with a branch for every reaction involving the species A. A molecule $A[u](\sigma)$ is an invocation $\widehat{\mathtt{A}}(\{\!|u, \sigma|\!\})$.

We begin by defining $\{\!|u, \sigma|\!\}$. Let ε and $\neg\varepsilon$ be two distinguished channels. Then $\{\!|u, \sigma|\!\}$ is equal to $\{\!|u|\!\}_0, \{\!|\sigma|\!\}_1, \{\!|\sigma|\!\}_2$, where

- $\{\!|u|\!\}_0$ yields the tuple of the values of the fields in u;
- $\{\!|\sigma|\!\}_1$ yields the concatenation of the gangs in the range of σ;
- $\{\!|\sigma|\!\}_2$ yields a tuple of length equal to the size of $\mathit{dom}(\sigma)$, whose i-th element is ϵ if all the element of the tuple $\sigma(i)$ are ε and $\neg\epsilon$ if not.

Then we continue with a sequence of definitions. We assume given a set of n reactions $\mathcal{R}$.

- $[x_1, \cdots, x_m]_u$ is the sequence of matches $[x_i = u(i)]_{i \in \mathrm{dom}(u)}$ (u is a partial map);
- $[x_1, \cdots, x_m]_\sigma$ is the sequence of matches $(M_i)_{i \in \mathrm{dom}(\sigma)}$ where $M_i = [x_i = \epsilon]$ if all the element of the tuple $\sigma(i)$ are ε, and $M_i = [x_i = \neg\epsilon]$ if not;

- $set(\widetilde{x}, u)$ is the tuple where the i-th element is $u(i)$ whenever $i \in \mathrm{dom}(u)$, it is the i-th element of $\widetilde{x}$, otherwise;
- $set_1(\widetilde{x}, \sigma)$ is the tuple where the element $n*(i-1)+j$ is the j-th element of the tuple $\sigma(i)$, when $i \in \mathrm{dom}(\sigma)$, and $x_{n*(i-1)+j}$ otherwise;
- $set_2(\widetilde{x}, \sigma)$ is the tuple where the element $n*(i-1)+j$ is ε if $\sigma(i) = \varepsilon^n$, it is $\neg\varepsilon$ if $i \in \mathrm{dom}(\sigma)$ and $\sigma(i) \neq \varepsilon^n$, and it is $x_{n*(i-1)+j}$ otherwise;
- $proj(\widetilde{x}, a)$ is the tuple $(x_{n*(a-1)+i})_{i \leq n}$ and $proj(\widetilde{x}, a, i)$ is $x_{n*(a-1)+i}$;
- if $A[u](\sigma), B[v](\phi) \stackrel{\lambda}{\rightarrow} A[u'](\sigma'), \mathsf{S} \in \mathcal{R}$ then both $A[u](\sigma) \stackrel{\lambda}{\rightarrow} A[u'](\sigma') \in_{\mathrm{L}} \mathcal{R}$ and $B[v](\phi) \stackrel{\lambda}{\rightarrow} \mathsf{S} \in_{\mathrm{R}} \mathcal{R}$;
- If ρ is a creation, $\mathsf{CR}(\rho, \mathcal{R})$ is a sequence $(x_1 : \lambda_1, \cdots, x_m : \lambda_m)$ where every subsequence $(x_{i\times n} : \lambda_{i\times n}, x_{i\times n+1} : \lambda_{i\times n+1}, \cdots, x_{i\times n+n-1} : \lambda_{i\times n+n-1})$ corresponds to the i-th bond created by ρ and $\lambda_{i\times n}, \cdots, \lambda_{i\times n+n-1}$ are the rates of the reactions in $\mathcal{R}$.

Every preliminary notation is in place for the definition of the encoding from nanoκ with gangs to SPIM.

Definition 3.3 Let $\mathcal{R}$ be a set of n reactions in nanoκ. The SPIM agent corresponding to the species A is:

$$\begin{array}{rcl} \widehat{\mathtt{A}}(\widetilde{x}, \widetilde{y}, \widetilde{z}) & = & \sum_{\rho : A[u](\sigma) \stackrel{\lambda}{\rightarrow} A[u'](\sigma') \in_{\mathrm{L}} \mathcal{R}} [\widetilde{x}]_u \, [\widetilde{z}]_\sigma \, \alpha_{\rho,L} \, . \, P_{\rho,L} \\ & & + \sum_{\rho : A[u](\sigma) \stackrel{\lambda}{\rightarrow} \mathsf{S} \in_{\mathrm{R}} \mathcal{R}} [\widetilde{x}]_u \, [\widetilde{z}]_\sigma \, \alpha_{\rho,R} \, . \, P_{\rho,R} \end{array}$$

where the length of $\widetilde{x}$ is the number of fields of A, and the lengths of $\widetilde{y}$ and $\widetilde{z}$ are the number of sites of A times n. In addition:

- if ρ is a creation with an empty set of bonds in the left-hand side then $\alpha_{\rho,L} = \overline{\rho}\,(\widetilde{u : \lambda})$ and $\alpha_{\rho,R} = \rho\,(\widetilde{u})$ and $(\widetilde{u : \lambda}) = \mathsf{CR}(\rho, \mathcal{R})$;
- if ρ is a creation with a bond x in the left-hand side then $\alpha_{\rho,L} = \overline{proj(\widetilde{y}, a, i)}\,(\widetilde{u : \lambda})$ and $\alpha_{\rho,R} = proj(\widetilde{y}, a, i)\,(\widetilde{u})$, where a is the site of A bound by x, i is the index of ρ in $\mathcal{R}$ and $(\widetilde{u : \lambda}) = \mathsf{CR}(\rho, \mathcal{R})$;
- if ρ is a destruction with a bond x in the left-hand side then $\alpha_{\rho,L} = \overline{proj(\widetilde{y}, a, i)}\,(\,)$ and $\alpha_{\rho,R} = proj(\widetilde{y}, a, i)\,(\,)$, where a is the site of A bound by x and i is the index of ρ in $\mathcal{R}$;
- if ρ is an exchange with an empty set of bonds in the left-hand side or with a bond occurring once and in A then $\alpha_{\rho,L} = \overline{\rho}\,\widetilde{u}$ and $\alpha_{\rho,R} = \rho\,(\widetilde{u})$, where $\widetilde{u}$ is either empty, if there is no bond in the left-hand side, or $proj(\widetilde{y}, A, a)$ if the site with the bond is a;
- if ρ is an exchange with a bond x shared by the reactants then one defines $\alpha_{\rho,L} = \overline{proj(\widetilde{y}, A, a, i)}\,(\widetilde{u})$ and $\alpha_{\rho,R} = proj(\widetilde{y}, A, a, i)\,(\widetilde{u})$, where a is the site of A bound by x, i is the index of ρ in $\mathcal{R}$ and $\widetilde{u}$ is either empty, if there is no bond in the left-hand side apart x, or $proj(\widetilde{y}, A, a')$ if A has a further bond on the site a'.

As regards continuations, $P_{\rho,L} = \widehat{\mathtt{A}}(set(\widetilde{x},u'), set_1(\widetilde{y},\sigma'), set_2(\widetilde{z},\sigma'))$ and $P_{\rho,R}$ is either $\mathbf{0}$, if $\mathsf{S} = \_$, or $\widehat{\mathtt{A}}(set(\widetilde{x},u'), set_1(\widetilde{y},\sigma'), set_2(\widetilde{z},\sigma'))\,,\widehat{\mathtt{C}_1}(\{\!|u_1|\!\}_1, \{\!|\phi_1|\!\}_2, \{\!|\phi_1|\!\}_3)\,,$ $\cdots\,,\widehat{\mathtt{C}_\mathtt{h}}(\{\!|u_h|\!\}_1, \{\!|\phi_h|\!\}_2, \{\!|\phi_h|\!\}_3)$ if $\mathsf{S} = A[u'](\sigma')\,, C_1[v_1](\phi_1)\,, \cdots\,, C_n[v_n](\phi_n)$.

The encoding of a $\mathtt{nano}\kappa$ calculus solution with gangs is:

$$\{\!|A_1[u_1](\sigma_1)\,, \cdots\,, A_m[u_m](\sigma_m)|\!\} \triangleq (\delta_\mathsf{S})(\widehat{\mathtt{A}_1}\{\!|u_1,\sigma_1|\!\}\,, \cdots\,, \widehat{\mathtt{A}_\mathtt{m}}\{\!|u_m,\sigma_m|\!\})$$

where δ_S is the minimal set that contains

- $(\rho : \lambda)$, if ρ has no bond between reactants and has rate λ,
- $(x_{n\times(i-1)} : \lambda_1, \cdots, x_{n\times(i-1)+n-1} : \lambda_n)$, if there is an agent invocation $\widehat{\mathtt{A}}\{\!|u_1,\sigma_1|\!\}$ and $\{\!|\sigma_1|\!\}_2 = (\cdots, x_{n\times(i-1)}, \cdots x_{n\times(i-1)+n-1}, \cdots)$, with $x_i \neq \varepsilon$ and $\lambda_1, \cdots, \lambda_n$ being the rates of the reactions in $\mathcal{R}$.

To illustrate our encoding we chose a toy-modelling of the transcription of a gene. There are four species:

- *Gn* models a gene. It has one field *tr* and two sites *pr* and *rnap*; *tr* is 1 when the gene is being transcripted by the RNA polymerase and 0 if not; *pr* and *rnap* are used to link to *Pr* and *RNAp*, respectively;
- *Pr* models the various promoter-sequences of the gene. It has one field *act* and two sites *rnap* and *gn*. The activation of the promoters by the transcription-factors is represented by switching *act* from 0 to 1. The sites *rnap* and *gn* are used to link *RNAp* and *Gn*, respectively;
- *RNAp* models the RNA polymerase. It has one field *act* that is set to 1 when the molecule is activated by the complexations with the promoters and the transcriptins factors, it is set to 0 otherwise. It has a site *link* that may be bound either to *Pr* or to *Gn*, according to the stage of the transcription;
- The species *mRNA* models the RNA messenger corresponding to the gene. It has neither fields nor sites.

There are three reactions. The creation ρ_1 models the binding of the RNA polymerase to the promoters (between sites *rnap* of *Pr* and *link* of *RNAp*) and their activation (update of the fields *act* of *Pr* and *RNAp*). The exchange ρ_2 models the movement of the RNA-polymerase to the gene and the beginning of the transcription itself (update of the field *tr* of *Gn* to 1). The destruction ρ_3 models the termination of the transcription.

$$\rho_1 :\ Pr[act^0](rnap)\ ,\ RNAp[act^0](link) \stackrel{\lambda_1}{\rightarrow} Pr[act^1](rnap^x)\ ,\ RNAp[act^1](link^x)$$

$$\rho_2 :\ Pr[act^1](rnap^y + gn^x)\ ,\ Gn[tr^0](pr^x + rnap) \stackrel{\lambda_2}{\rightarrow} Pr[act^0](rnap + gn^x)\ ,\ Gn[tr^1](pr^x + rnap^y)$$

$$\rho_3 :\ RNAp[act^1](link^x)\ ,\ Gn[tr^1](rnap^x) \stackrel{\lambda_3}{\rightarrow} RNAp[act^0](link)\ ,\ Gn[tr^0](rnap)\ ,\ mRNA$$

The encoding of this nanoκ systems yields the following four recursive definitions in SPIM. We notice that, in the encoding of Gn, the parameters $p\rho_1, p\rho_2, p\rho_3, ?p$ correspond to the gang of the site pr (the three former are yielded by $\{\![.]\!\}_1$ and the latter by $\{\![.]\!\}_2$), similarly the parameters $r\rho_1, r\rho_2, r\rho_3, ?r$ correspond to the gang of the site $rnap$.

$$\begin{array}{l}\widehat{Gn}(tr, p\rho_1, p\rho_2, p\rho_3, ?p, r\rho_1, r\rho_2, r\rho_3, ?r) \triangleq \\ \quad [tr = 0, ?p = \neg\epsilon, ?r = \epsilon]\, p\rho_2\, (r_1, r_2, r_3)\, .\, \widehat{Gn}(1, p\rho_1, p\rho_2, p\rho_3, ?p, r_1, r_2, r_3, \neg\epsilon) \\ \quad + [tr = 1, ?r = \epsilon]\, r\rho_3\, ()\, .\, \widehat{Gn}(1, p\rho_1, p\rho_2, p\rho_3, ?p, \epsilon, \epsilon, \epsilon, \epsilon)\end{array}$$

In the encoding of Pr, the parameters $r\rho_1, r\rho_2, r\rho_3, ?r$ correspond to the gang of the site $rnap$ and the parameters $g\rho_1, g\rho_2, g\rho_3, ?g$ correspond to the gang of the site gn.

$$\begin{array}{l}\widehat{Pr}(act, r\rho_1, r\rho_2, r\rho_3, ?r, g\rho_1, g\rho_2, g\rho_3, ?g) \triangleq \\ \quad [act = 0, ?r = \epsilon]\, \overline{\rho_1}\, (r_1 : \lambda_1, r_2 : \lambda_1, r_3 : \lambda_3)\, .\, \widehat{Pr}(1, r_1, r_2, r_3, \neg\epsilon, g\rho_1, g\rho_2, g\rho_3, ?g) \\ \quad + [act = 1, ?r = \neg\epsilon, ?g = \neg\epsilon]\, \overline{g\rho_2}\, (r\rho_1, r\rho_2, r\rho_3)\, .\, \widehat{Pr}(0, \epsilon, \epsilon, \epsilon, \epsilon, g\rho_1, g\rho_2, g\rho_3, ?g)\end{array}$$

In the encoding of the species $RNAp$, the parameters $l\rho_1, l\rho_2, l\rho_3, ?l$ correspond to the gang of the site $link$.

$$\begin{array}{l}\widehat{RNAp}(act, l\rho_1, l\rho_2, l\rho_3, ?l) \triangleq \\ \quad [act = 0, ?l = \epsilon]\, \rho_1\, (r_1, r_2, r_3))\, .\, \widehat{RNAp}(1, r_1, r_2, r_3, \neg\epsilon) \\ \quad + [act = 1, ?l = \neg\epsilon]\, \overline{l\rho_3}\ .\, (\widehat{RNAp}(0, \epsilon, \epsilon, \epsilon, \epsilon) \mid \widehat{mRNA})\end{array}$$

Since the molecule $mRNA$ is not involved in any reaction, the corresponding process is:

$$\widehat{mRNA}() \triangleq 0$$

Finally, the encoding of the solution $Pr[act^0](rnap{+}gn^x)$, $Gn[tr^0](prev^x{+}rnap)$,

$RNAp[act](link)$ is the term

$$(\rho_1 : \lambda_1, pr_1 : \lambda_1, pr_2 : \lambda_2, pr_3 : \lambda_3)(\ \widehat{Pr}(0, \epsilon, \epsilon, \epsilon, \epsilon, pr_1, pr_2, pr_3, \neg\epsilon) \mid \widehat{Gn}(0, pr_1, pr_2, pr_3, \neg\epsilon, \epsilon, \epsilon, \epsilon, \epsilon) \mid \widehat{RNAp}(act, \epsilon, \epsilon, \epsilon, \neg\epsilon)\ )$$

Remarkably, our encoding retains some interesting compositional properties. First of all, consider two solutions S and S' and their encodings $(\delta_{\mathsf{S}})P$ and $(\delta_{\mathsf{S}'})P'$, respectively. The encoding of the parallel composition $\mathsf{S}|\mathsf{S}'$ is $(\delta_{\mathsf{S}} \cup \delta_{\mathsf{S}'})P|P'$ where $(\delta_{\mathsf{S}} \cup \delta_{\mathsf{S}'})$ is the union of the name declarations δ_{S} and $\delta_{\mathsf{S}'}$ assuming that local names are disjoints (this assumption can be guaranteed exploiting α-conversion). Furthermore, if we add a new reaction rule to a system that we have already encoded, there are very few changes to be made. First, we need to expand the length of the gangs by one. After, we have to add one line to the definitions corresponding to the two species occurring in the left hand side of the new reaction, in order to describe this additional behaviour for the molecules belonging to these two species.

The next theorem states the correctness of $\{\!|.|\!\}$. If A is the l-th molecule in S and if ρ corresponds to the i-th branch of the choice in $\widehat{\mathtt{A}}$, we let $\{\!|l|\!\}_\rho$ be the pair (l, i). We also let $\{\!|\rho_L, \imath|\!\}$ and $\{\!|\rho_R, \imath|\!\}$ to be respectively $\alpha_{\rho,L}$ and $\alpha_{\rho,R}$ as defined in Definition 3.3.

Theorem 3.4 (i) *If* $\mathsf{S} \xrightarrow{\mu}_l \mathsf{T}$ *then* $\{\!|\mathsf{S}|\!\} \xrightarrow{\{\!|\mu|\!\}}_{\{\!|l|\!\}_\rho} \{\!|\mathsf{T}|\!\}$ *(similarly for* $\mathsf{S} \xrightarrow{\rho}_{l,l'} \mathsf{T}$*);*

(ii) *if* $\{\!|\mathsf{S}|\!\} \xrightarrow{\mu}_{l.i} \mathsf{T}$ *(resp.* $\{\!|\mathsf{S}|\!\} \xrightarrow{\rho}_{l.i,l'.j} \mathsf{T}$*) then there exist* S' *and* μ' *such that:* $\{\!|\mathsf{S}'|\!\} = \mathsf{T}$, $\mathsf{S} \xrightarrow{\mu'}_l \mathsf{S}'$, *and* $\mu = \{\!|\mu'|\!\}$ *(similarly for* $\{\!|\mathsf{S}|\!\} \xrightarrow{\rho}_{l.i,l'.i'} \mathsf{T}$*).*

Proof. The theorem is proven similarly to theorem 3.2 (this because we have called the three items of the definition 2.4 of the basic transition relation of the SPIM calculus with the same names (*init*), (*lifts*), and (*communications*) of the corresponding definition 2.3 for the nanoκ calculus). □

4 The stochastic collective semantics

The basic transition relation we considered keeps track of all the possible transitions that the molecules in a solution can perform. However, some of these transitions are somehow "equivalent" because, for instance, they have the same source and the targets are indistinguishable. This is the case when the solution contains several copies of a molecule and the reaction is an homeodimerization, i.e. two identical molecules get bound.

The following collective semantics merges "equivalent" transitions into one transition with an associated rate obtained as the sum of the rates of the merged transitions. It uses the structural equivalence to formalize the indistinguishability of solutions.

Definition 4.1 The structural equivalence of the nanoκ calculus, noted $\equiv$, is the least equivalence satisfying the following rules (solutions are already identified by associativity of "$,$"):

(i) $\mathsf{S},\mathsf{T} \equiv \mathsf{T},\mathsf{S}$;

(ii) $\mathsf{S} \equiv \mathsf{T}$ if there exists an injective renaming $\imath$ on bonds such that $\mathsf{S} = \imath(\mathsf{T})$.

The structural equivalence of the SPIM calculus, that, with an abuse of notation, we also note $\equiv$, is the least equivalence satisfying the following rules:

- $P|Q \equiv Q|P$
- if P is α-equivalent to Q then $P \equiv Q$

In order to give a unique definition of the collective semantics, we introduce a few notations. The letters F, G are used to range over solutions or processes; we assume that transitions of the basic transition systems have shape $\xrightarrow{\rho}_{\partial}$, where ∂ is a pair (we are considering evolutions of closed systems). Let also

- $next(F) = \{((\rho,\partial), G) \mid F \xrightarrow{\rho}_{\partial} G\}$;
- $next_\infty(F) = \{((\rho,\partial), G) \mid F \xrightarrow{\rho}_{\partial} G \text{ and } rate(\rho) = \infty\}$
- F has finite rates if and only if $next_\infty(F) = \emptyset$
- given a set $\mathcal{F}$ of pairs (X, T), where T is a term, let $[\mathcal{F}]_G$, where G is a fixed term, be $\{(X, G') \mid (X, G') \in \mathcal{F} \text{ and } G' \equiv G\}$;
- $can(\mathcal{F})$ is defined over sets of pairs (X, G) (the second element is a term, the first one is left unspecified), such that the terms occurring as second element of the pairs are all structurally equivalent. It returns a term G' such that there is X with $(X, G') \in \mathcal{S}$.

Definition 4.2 [Stochastic collective transition relation] The stochastic transition relation $\longmapsto$ induced by a basic transition relation $\xrightarrow{\rho}_{\partial}$ (∂ is a pair of indexes) and structural equivalence $\equiv$ on a language is the least relation satisfying the following rules:

- if $F \xrightarrow{\rho}_{\partial} G$ and $rate(\rho) = \infty$ then $F \overset{\infty}{\longmapsto} can([next_\infty(F)]_G)$;
- if $F \xrightarrow{\rho}_{\partial} G$ and F has finite rates then $F \overset{\lambda}{\longmapsto} can([next(F)]_G)$, where

$$\lambda \;=\; \sum_{((\rho,\partial),G') \in [next(F)]_G} rate(\rho)$$

The correctness result of the collective transition relation is stated below.

Theorem 4.3 $\mathsf{S} \overset{\lambda}{\longmapsto} \mathsf{T}$ *in* nanoκ *if and only if there exists* P *such that* $\{\![[\![\mathsf{S}]\!]_\jmath]\!\} \overset{\lambda}{\longmapsto} P$ *and* $P \equiv \{\![[\![\mathsf{T}]\!]_\jmath]\!\}$ *in* SPIM.

Our correctness notion corresponds to the subcase of the strong stochastic bisimulation [3] where the bisimulation relation is a bijection.

5 Future works

Our current interests are mainly about simulators and analysis tools for SPIM calculus. In fact, this contribution allows us to simulate `nano`κ systems. However, the same encoding also makes it possible to model-check `nano`κ formalizations in the PRISM platform [1], since it supports verifications of probabilistic and stochastic extensions of π-calculus [15]. More precisely, it should be possible to wire our encoding from the `nano`κ calculus to SPIM – a subset of stochastic π-calculus – with the implementation in [15]. This requires that reactions do not create molecules, otherwise the state space would be infinite, and PRISM cannot handle such systems.

There are two questions to bother with. Firstly, our encoding uses polyadic communications, which is still not considered in [15]. However this should be one of the next extensions of this work. The second issue is more problematic. A relevant constraint for the efficiency of the encoding in [15] is the absence of name creations within agent definition. This is not the case for our encoding, because agents may perform bounded outputs. Yet, in `nano`κ subsystems where the creation of new molecules is finite, the number of names used at every stage of the computation is finite. So, a clever algorithm might compute this number statically (an over-approximation is $k \times h$, where k is the maximal number of molecules and h is the maximal length of the arguments of an agent) and use a garbage-collection mechanism to recycle names. This should allow the static allocation of variables in the PRISM language to handle all the private names.

References

[1] L.de Alfaro, M.Z.Kwiatkowska, G.Norman, D.Parker, and R.Segala. Symbolic model checking of probabilistic processes using mtbdds and the kronecker representation. In *TACAS*, pages 395–410, 2000.

[2] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. A calculus of looping sequences for modelling microbiological systems. *Fundamenta Informaticae*, 72(1-3):21–35, 2006.

[3] M. Bernardo. A survey of markovian behavioral equivalences. In *Proc. of International School on Formal Methods for the Design of Computer, Communication, and Software Systems 2007*, volume 4486 of *LNCS*, pages 180–219, 2007.

[4] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of rkip on the erk signalling pathway using the stochastic process algebra pepa. In *Transactions on Computational Systems Biology VII*, volume 4230 of *Lecture Notes in Computer Science*, pages 1–23, 2006.

[5] L. Calzone, F. Fages, and S. Soliman. Biocham: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[6] L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190–215, 2008.

[7] L. Cardelli and A. Phillips. A corret abstract machine for the stochastic pi-calculus. In *Proc. of Workshop on Concurrent Models in Molecular Biology*, 2004.

[8] L. Cardelli and G. Zavattaro. On the computational power of biochemistry. In *Proc. of Algebraic Biology 2008*, volume to appear of *LNCS*, 2008.

[9] F. Ciocchetta and J. Hillston. Bio-pepa: a framework for modelling and analysis of biological systems. *Theoretical Computer Science*, to appear.

[10] A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, and G. Zavattaro. Modelization and simulation of nano devices in the nano-k calculus. In *Proc. of Computational Methods in Systems Biology 2007*, volume 4695 of *LNCS*, pages 168–183, 2007.

[11] Pierre-Louis Curien, Vincent Danos, Jean Krivine, and Min Zhang. Computational self-assembly. *Theoretical Computer Science*, 404(1-2):61–75, 2008.

[12] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. In *Proc. of Asian Symposium on Programming Languages and Systems 2007*, volume 4807 of *LNCS*, pages 139–157, 2007.

[13] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[14] Stephen Gilmore and Jane Hillston. The pepa workbench: a tool to support a process algebra-based approach to performance modelling. In *Proceedings of the 7th international conference on Computer performance evaluation : modelling techniques and tools*, pages 353–368, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[15] D.Parker G.Norman, C.Palamidessi and P.Wu. Model-checking probabilistic and stochastic extensions of the pi-calculus. In *IEEE Transactions on Software engineering*, 2007.

[16] J.Heath, M.Z.Kwiatkowska, G.Norman, D.Parker, and O.Tymchyshyn. Probabilistic model checking of complex biological pathways. In *CMSB*, pages 32–47, 2006.

[17] C. Laneve and A.Vitale. Expressivness in the κ-family. In *Proc. of MFPS*, ENTCS, 2008.

[18] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[19] Corrado Priami. Stochastic pi-calculus. *Computer Journal*, 38(7):578–589, 1995.

[20] Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.