From biochemistry to stochastic processes

Cosimo Laneve[1], Sylvain Pradalier[2], and Gianluigi Zavattaro[1]

[1] Dipartimento di Scienze dell'Informazione, Università di Bologna
laneve@cs.unibo.it, zavattar@cs.unibo.it
[2] Ecole Polytechnique, Paris
sylvain.pradalier@lix.polytechnique.fr

Abstract. The nanoκ calculus is a formalism for modelling biochemical systems following a *reactive-oriented* approach. We study the implementation of nanoκ into the Stochastic Pi Machine that complies with the *stochastic behaviors* of solutions. Our implementation allows us to use nanoκ as a front-end for a process-oriented simulator, thus being intelligible to biochemists, and to reuse theories and tools already developed for process calculi.

1 Introduction

Several stochastic formalisms emerged in the last few years as models for the representation of biological systems (see e.g. [3, 10, 1, 5, 7, 6] just to mention a few). These formalisms usually follow either a *reactive-oriented* (as [3, 1, 10] in the list above) or a *process-oriented* approach (as [5, 7, 6]). 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.

Process-oriented descriptions depart from ordinary biochemical models because they define the sequences of actions once and for all with ad-hoc syntaxes, use discrete data, and become entangled when specify fine-grain distributed controls. As a consequence, such descriptions are not intelligible to biochemists. On the other hand, process-oriented calculi retain several simulators and tools, which make them attractive for experiments *in silico*.

In this paper we bridge the gap between the two approaches by implementing the nanoκ calculus [8], a reactive-oriented formalism, into the *Stochastic Pi Machine* [5], SPIM calculus in the following, a simulator for the stochastic π-calculus [16, 5]. In nanoκ molecules may react by means of three types of reactions - creations, destructions, and exchanges - and retain a stochastic semantics. For example

$$\rho_1 \qquad A(a), B(b) \overset{\lambda_1}{\twoheadrightarrow} A(a^x), B(b^x)$$

models a creation of a bond between the site a of the molecule A and the site b of B. The real number λ_1 is the rate of the reaction. The reaction

$$\rho_2 \qquad A(a^x), B(b^x) \overset{\lambda_2}{\twoheadrightarrow} A(a), B(b)$$

defines a destruction that is opposite to the above creation. The rate λ_2 may be different from λ_1, thus yielding *equilibria* in accordance with them. The reaction

$$\rho_3 \qquad A(a^x),\, C(c) \overset{\lambda_3}{\twoheadrightarrow} A(a),\, C(^x)$$

is an exchange rule defining a bond flipping from the site a of `A` to c of `C`. Despite the simplicity of these reactions, it is hard to implement them into the SPIM calculus because of the stochastic semantics. Let us examine the problems.

An implementation of nanoκ into SPIM should project the behaviour of each molecule out of the data base 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

$$\begin{aligned}\widehat{\mathtt{A}}(z) \; &= \mathtt{behaviour\text{-}of\ A\ in}\ \rho_1 \\ &\quad + \mathtt{behaviour\text{-}of\ A\ in}\ \rho_2 \\ &\quad + \mathtt{behaviour\text{-}of\ A\ in}\ \rho_3\end{aligned}$$

That is, a molecule is implemented by a parametric process denition, where the parameters define the values of fields and sites (fields in nanoκ are intended to model the state of the molecule, such as its shape or its hydrogen groups or phosphor groups). Next, the "`behaviour-of A in` ρ_1" might be defined as

$$[z = \varepsilon]\; \overline{\rho_1}\,(x).\widehat{\mathtt{A}}(x)$$

where

- $[z = \varepsilon]$ means that such a behaviour may be triggered provided the site a is unbound (has value ε);
- in this case the channel ρ_1 is used to output a fresh name (modelling the bond). We expect that the behaviour of `B` will perform a corresponding input when the site b is unbound as well. We also expect that the rate of the channel ρ_1 has been declared to be λ_1;
- then $\widehat{\mathtt{A}}$ will continue as the process $\widehat{\mathtt{A}}(x)$.

This analogy reaction-names as channels fails for modelling destructions. For example, if the "`behaviour-of A in` ρ_2" is defined as

$$[z = \neg\varepsilon]\; \overline{\rho_2}\,().\widehat{\mathtt{A}}(\varepsilon)$$

then an $\widehat{\mathtt{A}}$ might interact with the wrong $\widehat{\mathtt{B}}$ in the implementation of $A(a^x), B(b^x), A(a^y), B(b^y)$. This means that we should profit of 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

$$[z = \neg\varepsilon]\; \overline{z}\,().\widehat{\mathtt{A}}(\varepsilon)$$

where we are assuming that $\widehat{\mathtt{B}}$ will input on z and that the rate of z is λ_2. But this solution is also inadequate because we might have a further destruction on the site a of `A`:

$$\rho_4 \qquad A(a^x),\, C(c^x) \overset{\lambda_4}{\twoheadrightarrow} A(a),\, C(c)$$

with $\lambda_4 \neq \lambda_2$. Because of this inequality, it is not possible to use the channel x anymore. One should rather use a channel for every reaction addressing the bond. In the above example:

$$\begin{aligned}\widehat{\mathtt{A}}(z_1, z_2) \;&= [z_1 = \varepsilon]\ \overline{\rho_1}\,(x : \lambda_2).\ \widehat{\mathtt{A}}(x, z_2)\\ &\quad + [z_1 = \neg\varepsilon]\ \overline{z_1}\,().\ \widehat{\mathtt{A}}(\varepsilon, z_2)\\ &\quad + [z_2 = \neg\varepsilon]\ \overline{z_2}\,().\ \widehat{\mathtt{A}}(z_1, \varepsilon)\\ &\quad + \cdots\end{aligned}$$

where $\overline{\rho_1}\,(x : \lambda_2)$ creates a channel x with rate λ_2. However this solution is defective, too. Consider the two rules

$$\begin{array}{ll}\rho_5 & A(a^x), C(c) \overset{\lambda_5}{\rightharpoonup} A(a), C(^x)\\ \rho_6 & C(c^x), B(b^x) \overset{\lambda_6}{\rightharpoonup} C(c), B(b)\end{array}$$

with $\lambda_6 \neq \lambda_2$. If the creation of x amounted to the creation of the channel in the first argument of $\widehat{\mathtt{A}}$ then one is in trouble when $\widehat{\mathtt{B}}$ is going to be defined. This because $\mathtt{B}$ may destroy the bond x on b by reacting either with $\mathtt{A}$ (with rate λ_2) or with $\mathtt{C}$ (with rate λ_6). Therefore the same channel cannot be used in the behaviours of the two destructions ρ_2 and ρ_6 in $\widehat{\mathtt{B}}$. This means that the behaviour of ρ_1 must create (at least) two channels that will be used by the behaviours of ρ_2 and ρ_6, respectively. It is worth to remark that a communication between $\widehat{\mathtt{A}}$ and $\widehat{\mathtt{B}}$ may create a channel that will be used in a communication between two other processes.

To be accurate, 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. (In the above examples, addressing was performed by destructions, but a bond may be addressed by creations and exchanges, as well.) Actually, for simplicity, our solution over-approximates the "precise" solution, by representing bonds with tuples whose length is the number of reactions in the database – a *gang*, in our terminology.

There is another subtle issue about encodings of stochastic calculi. Our encoding $[\![\cdot]\!]$ of nanoκ into SPIM is such that $\mathsf{S} \overset{\lambda}{\longmapsto}_{\mathtt{nano}\kappa} \mathsf{T}$ if and only if $[\![\mathsf{S}]\!] \overset{\lambda}{\longmapsto}_{spim} [\![\mathsf{T}]\!]$ (we are subscribing the reductions for reader's convenience). In particular S and $[\![\mathsf{S}]\!]$ are *strongly stochastic bisimilar* [2]. This is different from usual implementations that almost never preserve the granularity of transitions. This strong relationship can be hardly weakened because stochastic rates of transitions correspond to an exponential law controlling the waiting before a transition can be fired (the *sojour time*). Since the sum of exponentials is not an exponential, it will be difficult to assert the correctness of an encoding that maps one transition to several.

Related works. In [17], it has been shown that systems of molecular interactions with explicit bonds might be represented and simulated using the stochastic π-calculus. Our encoding corrobates this result since the SPIM calculus is a subset of the stochastic π-calculus. We remark that the example provided in [17] and, we believe, the descriptions done in this approach can be rewritten in SPIM calculus and even in sub-calculus of its, since our encodings doesn't use its full power.

In [4], 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 [4] is a subset of the SPIM calculus and because the nanoκ calculus extends the language of chemical reactions of [4] with explicit bonds between molecules and with internal states. However, our results are weaker than those in [4], since we only assert the correctness of the encoding with respect to the stochastic semantics.

Encodings from the full κ calculus to nanoκ calculus, or to π-calculus are presented in [11] and [9]. Yet, they only preserve non-stochastic semantics. Indeed, encodings preserving the stochastic semantics do not exist, due to the negative results of [14]).

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. Then in part 3, we present the encoding. It is down in two steps, in 3.1 we modify nanoκ in order to introduce tuples of names – the *gangs* – and in 3.2 we complete the definition of the encoding. The correctness of each step with respect to the basic stochastic semantics is asserted by theorems 1 and 2. In part 4 we present the collective stochastic semantics and present the theorem 3 which states the correctness of the encoding with respect to the collective stochastic semantics.

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 [8] and [5].

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 *bonds*(S) the set of the bonds appearing in the solution S.

Definition 1. *A* solution *is a term defined by the grammar*

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

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

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

The `nano`κ calculus semantics is defined by means of reaction rules. 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 the functions are all partial and $\mathrm{dom}(u) \cap \mathrm{dom}(u') = \emptyset$ and $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\sigma') = \emptyset$.

Definition 2. *Reactions of* `nano`κ *calculus are either* creations, destructions, *or* exchanges *that are labelled by* rates, *which are positive real numbers or* ∞. *Creations have format*

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

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*

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

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:*

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

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:

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

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 explicitely. For instance $A(\widetilde{x}) \triangleq \underset{i\in\{1\}}{\Sigma}[\widetilde{x}_i=\widetilde{x}_i]\alpha.P$ 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 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 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, even if it is intensional. It follows that this correcteness also holds for the collective semantics, since it is derived in the same way from the basic transition relation (see theorem 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 $\mathtt{nano}\kappa$ reactions may be addressed by:

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

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

Definition 3. *The basic transition relation of* $\mathtt{nano}\kappa$*, written either* $\xrightarrow{\rho,\imath}_{\ell,\ell'}$ *or* $\xrightarrow{\mu,\imath}_{\ell}$*, is the least relation that satisfies the following rules:*

– *(init) let* $\rho = A[u](\sigma)\,\mathtt{,}\,B[v](\phi) \overset{\lambda}{\rightarrow} A[u'](\sigma')\,\mathtt{,}\,\mathsf{S}$*. Then both* $A[u+w](\sigma \circ \imath + \nu) \xrightarrow{\rho_L,\imath}_1 A[u'+w](\sigma' \circ \imath + \nu)$ *and* $B[v+w](\phi \circ \imath + \nu) \xrightarrow{\rho_R,\imath}_1 \mathsf{T}$*, where* T *is either* $B[v'+w](\phi' \circ \imath + \nu)\,\mathtt{,}\,\imath(\mathsf{S})$ *or* $\imath(\mathsf{S})$*, according to the shape of the right hand side, where* $\imath$ *is an injective renaming and where* $\mathrm{ran}(i) \cap \mathrm{ran}(\nu) = \emptyset$*;*

– *(lifts) if* $\mathsf{S} \xrightarrow{\mu,\imath}_{\ell} \mathsf{S}'$ *and* $(\mathit{bonds}(\mathsf{S}') \setminus \mathit{bonds}(\mathsf{S})) \cap \mathit{bonds}(\mathsf{T}) = \emptyset$, *then both* $\mathsf{S},\mathsf{T} \xrightarrow{\mu,\imath}_{\ell} \mathsf{S}',\mathsf{T}$ *and* $\mathsf{T},\mathsf{S} \xrightarrow{\mu,\imath}_{\ell'+\ell} \mathsf{T},\mathsf{S}'$, *where* T *has* ℓ' *molecules;*
– *(communications) if* $\mathsf{S} \xrightarrow{\mu,\imath}_{\ell} \mathsf{S}'$ *and* $\mathsf{T} \xrightarrow{\overline{\mu},\imath}_{\ell'} \mathsf{T}'$ *and* $\imath$ *is an order-preserving injection that map bonds into the least ones not used in* S,T *then* $\mathsf{S},\mathsf{T} \xrightarrow{\rho}_{\ell,\ell''+\ell'} \mathsf{S}',\mathsf{T}'$, *where* ρ *is the rule of* μ *and* S *has* ℓ'' *molecule.*

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 $\ell'' + \ell'$, 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 $\rho : H(1), H(1) \overset{\lambda}{\rightarrowtriangle} H(1^u), H(1^u)$ be the hydrogen gas reaction. Then the following three transitions are possible

$$\begin{aligned}
&3H(1) \xrightarrow{\rho}_{1,2} 2H(1^x), H(1)\\
&3H(1) \xrightarrow{\rho}_{1,3} H(1^x), H(1), H(1^x)\\
&3H(1) \xrightarrow{\rho}_{2,3} H(1), 2H(1^x)
\end{aligned}$$

The basic transition relation is labelled by finite injective renamings. To clarify this point, consider the creation $\varrho = Na(1^x+2), Na(1^x+2) \overset{10}{\rightarrowtriangle} Na(1^x+2^y), Na(1^x+2^y)$ (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

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

following a structured operational semantics approach [15]. Namely, we focus on the single reactants and lift the transitions to "**,**"-contextes. 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 $Na[ion^0](1^v+2) \not\xrightarrow{\varrho_R,\imath}_1$.

Our final remarks regard the rule *(communications)*. There are possibly infinitely many transitions $\mathsf{S} \xrightarrow{\rho,\imath}_{\ell} \mathsf{T}$ 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 $\xrightarrow{\mu}_{\ell,\ell'}$, which models the evolution of a solution, is finitely branching, while the auxiliary relation $\xrightarrow{\mu}_{\ell}$ is not finitely branching. It is also worth to notice that there is no rule lifting a transition $\xrightarrow{\mu}_{\ell,\ell'}$ to a context "$\texttt{,}$": we use the associativity of $\texttt{,}$ to partition a solution S into $\mathsf{S}'\texttt{,}\mathsf{S}''$ 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 4. *The basic transition relation of the* SPIM *calculus, written either* $\xrightarrow{\alpha}_{\ell.i,\ell'.j}$ *or* $\xrightarrow{\tau_\lambda}_{\ell.i,\ell'.j}$ *or* $\xrightarrow{\alpha}_{\ell.i}$*, 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* $P \xrightarrow{\alpha}_{\ell.i} P'$ *and* $\mathtt{bn}(\alpha) \cap \mathtt{fn}(Q) = \emptyset$ *and* $\ell' =$ *length*(Q)*, then both* $P \mid Q \xrightarrow{\alpha}_{\ell.i} P' \mid Q$ *and* $Q \mid P \xrightarrow{\alpha}_{\ell'+\ell.i} Q \mid P'$*;*
- *(comunications) let* $\ell'' =$ *length*(P)*,* λ *be the rate of* x*, and* $Q \xrightarrow{x\,(\widetilde{u})}_{\ell'.i'} Q'$*. If* $P \xrightarrow{\overline{x}\,\widetilde{v}}_{\ell.i} P'$ *then* $(\widetilde{z} : \widetilde{\lambda'})(P \mid Q) \xrightarrow{\tau_\lambda}_{\ell.i,\ell'+\ell''.i'} [(\widetilde{z} : \widetilde{\lambda'})(P \mid Q\{\widetilde{v}/\widetilde{u}\}]_{GC}$*; if* $P \xrightarrow{\overline{x}\,(\widetilde{v}:\widetilde{\lambda''})}_{\ell.i} P'$ *then* $(\widetilde{z} : \widetilde{\lambda'})(P \mid Q) \xrightarrow{\tau_\lambda}_{\ell.i,\ell'+\ell''.i'} [(\widetilde{z} : \widetilde{\lambda'} + \widetilde{v} : \widetilde{\lambda''})(P' \mid Q'\{\widetilde{v}/\widetilde{u}\})]_{\mathtt{GC}}$ *where* $\widetilde{v}$ *are the least names not occurring in* $P \mid Q$*. Simmetrically when* P *performs an input and* Q *performs an output.*

As for $\mathtt{nano}\kappa$, there is always at most one $(\widetilde{z} : \widetilde{\lambda'})P'$ such that $(\widetilde{x} : \widetilde{\lambda})P \xrightarrow{\tau_{\lambda''}}_{\ell,i,\ell',i'} (\widetilde{z} : \widetilde{\lambda'})P'$ 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 $\mathtt{nano}\kappa$ calculus into the SPIM calculus

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

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 5. *Let $\mathcal{R} = \{\rho_i : \mathsf{L}_i \overset{\lambda_i}{\rightarrowtriangle} \mathsf{R}_i \mid i \in 1..n\}$ be a set of* `nano`κ *reaction rules and let $\jmath$ be an 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 $\{\rho_i : [\![\mathsf{L}_i]\!]_\jmath \overset{\lambda_i}{\rightarrowtriangle} [\![\mathsf{R}_i]\!]_\jmath \mid i \in 1..n\}$.

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 $\rho_R, [\![\imath]\!]_\jmath$, according to μ is $\rho_L, \imath$ or $\rho_R, \imath$. The correctness of the encoding of Definition 5 is stated in the following theorem.

Theorem 1. *1. if $\mathsf{S} \xrightarrow{\mu}_\ell \mathsf{T}$ then $[\![\mathsf{S}]\!]_\jmath \xrightarrow{[\![\mu]\!]_\jmath}_\ell [\![\mathsf{T}]\!]_\jmath$ (similarly for $\mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$);*

2. if $[\![\mathsf{S}]\!]_\jmath \xrightarrow{\mu}_\ell \mathsf{T}$ 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 $[\![\mathsf{S}]\!]_\jmath \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$).

(The proof is omitted because it is a standard induction on the proof trees of the transitions.)

3.2 From gangs to the SPIM calculus: agents as molecules

The second step of our translation encodes the `nano`κ 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 the same length of $\{\![\sigma]\!\}_1$ such that the i-th element is ε if the i-th element of $\{\![\sigma]\!\}_1$ is ε and it is $\neg\varepsilon$ otherwise.

Then we continue with a sequel of notational 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 $(M_{i,j})_{i \in \mathrm{dom}(\sigma), j \leq n}$ where $M_{i,j} = [x_{n*(i-1)+j} = \epsilon]$ if the j-th element of the tuple $\sigma(i)$ is ε, and $M_{i,j} = [x_{n*(i-1)+j} = \neg\epsilon]$ 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 $A[u](\sigma)\,\text{,}\,B[v](\phi) \overset{\lambda}{\rightarrow} A[u'](\sigma')\,\text{,}\,\mathsf{S} \in \mathcal{R}$ then both $A[u](\sigma) \overset{\lambda}{\rightarrow} A[u'](\sigma') \in_{\mathrm{L}} \mathcal{R}$ and $B[v](\phi) \overset{\lambda}{\rightarrow} \mathsf{S} \in_{\mathrm{R}} \mathcal{R}$;
- 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 $(x_{i \times n} : \lambda_{i \times n}, x_{i \times n+1} : \lambda_{i \times n+1}, \cdots, x_{i \times n+n-1} : \lambda_{i \times n+n-1})$ correspond 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 `nano`κ with gangs to SPIM.

Definition 6. *Let $\mathcal{R}$ be a set of n reactions in* `nano`κ*. The* SPIM *agent corresponding to the species A is:*

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

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 $\alpha_{\rho,L} = \overline{proj(\widetilde{y}, A, a, i)}\,(\widetilde{u})$ 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 $\mathcal{R}$.

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 $\{[\ell]\}_\rho$ 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 6.

Theorem 2. *1. If $\mathsf{S} \xrightarrow{\mu}_\ell \mathsf{T}$ then $\{[\mathsf{S}]\} \xrightarrow{\{[\mu]\}}_{\{[\ell]\}_\rho} \{[\mathsf{T}]\}$ (similarly for $\mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$);*
2. 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: $\{[\mathsf{S}']\} = \mathsf{T}$, $\mathsf{S} \xrightarrow{\mu'}_\ell \mathsf{S}'$, and $\mu = \{[\mu']\}$ (similarly for $\{[\mathsf{S}]\} \xrightarrow{\rho}_{\ell.i,\ell'.i'} \mathsf{T}$).

The proof is similar to that of Theorem 1.

4 The stochastic collective semantics

The basic transition relation we considered takes 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.

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 7. *The structural equivalence of the $\mathtt{nano}\kappa$ calculus is the least equivalence satisfying the following rules (solutions are already quotiented by associativity of “,”):*

1. $\mathsf{S}\,,\mathsf{T} \equiv \mathsf{T}\,,\mathsf{S}$;

2. $\mathsf{S} \equiv \mathsf{T}$ *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 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

– $\mathit{next}(F) = \{((\rho, \partial), G) \mid F \xrightarrow{\rho}_{\partial} G\}$;
– $\mathit{next}_{\infty}(F) = \{((\rho, \partial), G) \mid F \xrightarrow{\rho}_{\partial} G \text{ and } \mathit{rate}(\rho) = \infty\}$
– F has finite rates if and only if $\mathit{next}_{\infty}(F) = \emptyset$
– let $\mathcal{F}$ be a set of pairs (X, G) (the second element is a term, the first one is left unspecified), $[\mathcal{F}]_G$ is the subset of $\mathcal{F}$ of those pairs (X, G') such that $G' \equiv G$;
– $\mathit{can}(\mathcal{F})$ 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 8 (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* $\mathit{rate}(\rho) = \infty$ *then* $F \overset{\infty}{\longmapsto} \mathit{can}([\mathit{next}_{\infty}(F)]_G)$*;*
– *if* $F \xrightarrow{\rho}_{\partial} G$ *and* F *has finite rates then* $F \overset{\lambda}{\longmapsto} \mathit{can}([\mathit{next}(F)]_G)$*, where*

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

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

Theorem 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 [2] where the bisimulation relation is a bijection.

5 Future works

Our current interests are mainly about simulators and analysis tools for SPIM calculus. In facts, this contribution allows us to simulate nanoκ systems. However, the same encoding makes also possible to model-check nanoκ formalizations in the PRISM platform [12], since it supports verifications of probabilistic and stochastic extensions of π-calculus [13]. More precisely, it should be possible to wire our encoding from the nanoκ calculus to SPIM – a subset of stochastic π-calculus – with the implementation in [13]. There are two questions to bother with. Firstly, our encoding uses polyadic communications, which is still not considered in [13]. 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 [13] 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 nanoκ 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. 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.
2. 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.
3. L. Calzone, F. Fages, and S. Soliman. Biocham: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.
4. L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190–215, 2008.
5. 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.
6. L. Cardelli and G. Zavattaro. On the computational power of biochemistry. In *Proc. of Algebraic Biology 2008*, volume to appear of *LNCS*, 2008.
7. F. Ciocchetta and J. Hillston. Bio-pepa: An extension of the process algebra pepa for biochemical networks. *Electronic Notes in Theoretical Computer Science*, 194(3):103–117, 2008.
8. 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.
9. Pierre-Louis Curien, Vincent Danos, Jean Krivine, and Min Zhang. Computational self-assembly. *Theor. Comput. Sci.*, 404(1-2):61–75, 2008.
10. 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.
11. V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

12. Luca de Alfaro, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Roberto Segala. Symbolic model checking of probabilistic processes using mtbdds and the kronecker representation. In *TACAS*, pages 395–410, 2000.
13. 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.
14. C. Laneve and A.Vitale. Expressivness in the κ-family. In *Proc. of MFPS*, ENTCS, 2008.
15. G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.
16. Corrado Priami. Stochastic pi-calculus. *Computer Journal*, 38(7):578–589, 1995.
17. 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.