A Generic Abstract Machine for Stochastic Process Calculi

Loïc Paulevé
IRCCyN, UMR CNRS 6597
École Centrale de Nantes
France
loic.pauleve@irccyn.ec-nantes.fr

Simon Youssef
Department für Physik
Lehrstuhl Rädler
Ludwig-Maximilians-Universität
München, Germany
youssef@physik.lmu.de

Matthew R. Lakin
Microsoft Research
7 JJ Thomson Avenue
Cambridge, UK
v-mlakin@microsoft.com

Andrew Phillips
Microsoft Research
7 JJ Thomson Avenue
Cambridge, UK
andrew.phillips@microsoft.com

ABSTRACT

This paper presents a generic abstract machine for simulating a broad range of process calculi with an arbitrary reaction-based simulation algorithm. The abstract machine is instantiated to a particular calculus by defining two functions: one for transforming a process of the calculus to a set of species, and another for computing the set of possible reactions between species. Unlike existing simulation algorithms for chemical reactions, the abstract machine can simulate process calculi that generate potentially unbounded numbers of species and reactions. This is achieved by means of a just-in-time compiler, which dynamically updates the set of possible reactions and chooses the next reaction in an iterative cycle. As a proof of concept, the generic abstract machine is instantiated for the stochastic pi-calculus, and the instantiation is implemented as part of the SPiM stochastic simulator. The structure of the abstract machine facilitates a significant optimisation by allowing channel restrictions to be stored as species complexes. We also present a novel algorithm for simulating chemical reactions with general distributions, based on the Next Reaction Method of Gibson and Bruck. We use our generic framework to simulate a stochastic pi-calculus model of plasmid co-transfection, where plasmids can form aggregates of arbitrary size and where rates of mRNA degradation are non-exponential. The example illustrates the flexibility of our framework, which allows an appropriate high-level language to be paired with the required simulation algorithm, based on the biological system under consideration.

Categories and Subject Descriptors

I.6.1 [**Simulation and Modeling**]: Simulation Theory;

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Stochastic processes*; F.4.3 [**Theory of Computation**]: Mathematical logic and formal languages—*Formal Languages*

General Terms

Algorithms

Keywords

generic abstract machine, non-Markovian simulation, stochastic pi-calculus, bioambient calculus, implementation

1. INTRODUCTION

Biological systems typically involve large numbers of components with complex, highly parallel interactions and intrinsic stochasticity. Numerous programming languages have been developed for modelling such systems, many of which are based on process calculi. Examples include variants of the stochastic pi-calculus [18, 20, 9, 13], Bio-PEPA [2], BlenX [4], the Kappa-calculus [3], LBS [11], variants of the Bioambient calculus [19, 12] and DSD [14], to name but a few. Most of these calculi are expressive enough to generate potentially unbounded numbers of species and reactions. As a result, they cannot rely on standard reaction-based simulation algorithms or tools [8, 6], and generally require a custom simulation algorithm. To help address this issue, we propose a generic abstract machine that can be instantiated to a range of process calculi and a range of reaction-based simulation algorithms. The abstract machine can handle process calculi with potentially unbounded numbers of species and reactions.

Although the idea of integrating different modelling and simulation methods within a common framework is not a new one [5], our approach is the first attempt to formally define a generic framework for simulating a broad range of process calculi with an arbitrary reaction-based simulation algorithm. Having a clear separation between the simulation algorithm and the language specification allows us not only to rapidly instantiate the machine to different process calculi, but also to add new features, such as non-Markovian

simulation, which can then be used by all instantiated process calculi. Markovian reactions predominate in stochastic models of biological systems, largely due to the so-called *memoryless property* of the Markovian distribution, which assumes that the duration of the next reaction occurring in the system does not depend on the history of the system. Thanks to this property, very efficient simulation algorithms have been created, such as the Gillespie algorithm [7] along with its many optimisations [6, 8, 22]. However, Markovian reactions are quite restrictive in terms of modelling flexibility. For example, the variance and mean of the exponential law are tied: the mean of a random variable following an exponential distribution of rate r is $1/r$, while its variance is $1/r^2$. Providing tools for simulating non-Markovian reactions would enable greater accuracy in analysing the effects of stochasticity within models.

We apply our generic abstract machine to the simulation of a variant of stochastic pi-calculus [13] with general rate distributions. Based on the Next Reaction Method [6], we present a novel algorithm for simulating non-Markovian reactions, and we show how the generic abstract machine can be used to achieve correct simulation. We also extend the stochastic pi-calculus with a basic complexation primitive using bound output, and extend the abstract machine so that it stores complexes as a single species, allowing efficient simulation of complexes. As a biological application of our approach, we propose a stochastic pi-calculus model of plasmid co-transfection to simulate gene transfer. In this model, plasmids can form complexes of arbitrary size and the degradation of mRNA produced by the translocated plasmids is efficiently simulated using an Erlang distribution.

The paper is structured as follows. The generic abstract machine is defined in Sec. 2, and the simulation of non-Markovian species is tackled in Sec. 3.2. The machine is instantiated to the stochastic pi-calculus with general distributions in Sec. 4, and the instantiated machine is applied to the non-Markovian simulation of a stochastic pi-calculus model of plasmid co-transfection in Sec. 5. Appendix B shows an instantiation of the machine to the Bioambient calculus.

2. GENERIC ABSTRACT MACHINE

This section presents a generic abstract machine for simulating a broad range of process calculi with an arbitrary reaction-based simulation algorithm (Definition 1). A machine term T is a triple (t, S, R), where t is the current time, S contains the set of possible species and R contains the set of possible reactions. S maps each species I to its population, while R maps each reaction O to its activity A, which is used to compute the next reaction. The syntax of species I is specific to the choice of process calculus. Each reaction is represented by a tuple (J, F, f, J'), where $J = \{I_1, \ldots, I_N\}$ denotes the reactant species, $J' = \{I'_1, \ldots, I'_M\}$ denotes the product species, F denotes the probability distribution of the reaction and f is a function which allows the calculus to modify the machine term (as in Appendix B). Thus, reactions are assumed to be of the form $I_1 + \cdots + I_N \xrightarrow{F} I'_1 + \cdots + I'_M$. Once the next reaction has been selected, it is executed by removing the reactants J from the machine term, adding the products J' and updating the current time of the machine (7).

To instantiate the abstract machine with a given process

$$
\begin{aligned}
T &::= & (t, S, R) & \quad \text{Term} \\
R &::= & \{O_1 \mapsto A_1, \ldots, O_N \mapsto A_N\} & \quad \text{Reactions} \\
S &::= & \{I_1 \mapsto i_1, \ldots, I_N \mapsto i_N\} & \quad \text{Populations} \\
J &::= & \{I_1, \ldots, I_N\} & \quad \text{Species set} \\
O &::= & (J, F, f, J') & \quad \text{Reaction}
\end{aligned}
$$

$$
\begin{aligned}
P \oplus (t, S, R) &\triangleq species(P) \oplus (t, S, R) && (1) \\
\{I_1, \ldots, I_N\} \oplus (t, S, R) &\triangleq I_1 \oplus \ldots \oplus I_N \oplus (t, S, R) && (2) \\
(t, S, R) \ominus \{I_1, \ldots, I_N\} &\triangleq (t, S, R) \ominus I_1 \ominus \ldots \ominus I_N && (3)
\end{aligned}
$$

$$
\begin{aligned}
I \oplus (t, S, R) &\triangleq (t, S', R \cup R') \text{ if } S(I) = i && (4) \\
&\quad \textbf{and } S' = S\{I \mapsto i + 1\} \\
&\quad \textbf{and } R' = updates(I, (t, S', R)) \\
I \oplus (t, S, R) &\triangleq (t, S', R \cup R') \text{ if } I \notin dom(S) && (5) \\
&\quad \textbf{and } L = reactions(I, dom(S)) \\
&\quad \textbf{and } S' = S\{I \mapsto 1\} \\
&\quad \textbf{and } R' = init(L, (t, S', R)) \\
(t, S, R) \ominus I &\triangleq (t, S', R \cup R') \text{ if } S(I) = i && (6) \\
&\quad \textbf{and } S' = S\{I \mapsto i - 1\} \\
&\quad \textbf{and } R' = updates(I, (t, S', R))
\end{aligned}
$$

$$
\frac{(J, F, f, J'), t' = next(t, S, R)}{t, S, R \xrightarrow{F, (J, F, f, J')} f(J' \oplus ((t', S, R) \ominus J))} \quad (7)
$$

DEFINITION 1. *Generic abstract machine.* The notation $S(I)$ returns the corresponding value associated with I in S, while the notation $S\{I \mapsto v\}$ associates the value v with I in S. The function $species(P)$ computes the set of species corresponding to a process P, while the function $reactions(I, J)$ computes the set of reactions between a new species I and an existing set of species J. These two functions are specific to the choice of process calculus. The function $next(T)$ computes the next reaction of a term T, while the function $init(L, T)$ initialises a term T with a set of reactions, and the function $updates(I, T)$ updates the reactions in a term T affected by a given species I. These three functions are specific to the choice of simulation algorithm.

calculus, it is sufficient to define a function $species(P)$ for transforming a process P to a multiset of species, together with a function $reactions(I, \{I_1, \ldots I_N)\}$ for computing the multiset of reactions between a new species I and an existing set of species $\{I_1, \ldots, I_N\}$. The *species* function is used to initialise the abstract machine at the beginning of a simulation, while the *reactions* function is used by a just-in-time compiler to generate the set of possible reactions dynamically. This allows systems with potentially unbounded numbers of species and reactions to be simulated. To instantiate the abstract machine with a given simulation algorithm, it is sufficient to define a function $next(T)$ for choosing the next reaction, together with a function $init(L, T)$ for initialising a term T with a set of reactions L, and a function $updates(I, T)$ for updating the reactions in a term T affected

$$next(t, S, R) \triangleq O, t' \text{ if } R(O) = (a, t') \quad (8)$$
$$\textbf{and } t' = \min\{t \mid R(O) = (a, t)\}$$
$$init(L, (t, SR)) \triangleq \{O \mapsto (t', a) \mid O \in L \quad (9)$$
$$\wedge\; a = propensity(O, S)$$
$$\wedge\; O = (J, F, f, J')$$
$$\wedge\; t' = t + delay(F, a)\}$$
$$updates(I, (t, S, R)) \triangleq \{O \mapsto (t', a') \mid R(O) = (t'', a) \quad (10)$$
$$\wedge\; O = (J, F, f, J') \wedge I \in J$$
$$\wedge\; a' = propensity(O, S)$$
$$\wedge\; t' = t + (a/a')(t'' - t)\}$$

$$propensity((\{I\}, F, J), S) \triangleq rate(F) \cdot i \text{ if } S(I) = i$$
$$propensity((\{I, I\}, F, J), S) \triangleq rate(F) \cdot i \cdot (i - 1)/2$$
$$\textbf{if } S(I) = i$$
$$propensity((\{I_1, I_2\}, F, J), S) \triangleq rate(F) \cdot i_1 \cdot i_2 \text{ if } I_1 \neq I_2$$
$$\textbf{and } S(I_1) = i_1$$
$$\textbf{and } S(I_2) = i_2$$

DEFINITION 2. *Generic abstract machine instantiated for the Next Reaction Method.* Each reaction O is associated with a pair $A = (a, t)$, where a denotes the reaction propensity and t denotes the time at which the reaction is scheduled to occur. The function $delay(F, a)$ computes a time interval from a random variable with probability distribution F and propensity a. We use standard multiset notation $\{Elements \mid Conditions\}$ where $Elements$ represents the elements of the multiset and $Conditions$ represents the conditions the elements must satisfy.

by a given species I.

A process P is added to the machine term (t, S, R) by computing the multiset of species $\{I_1, \ldots, I_N\}$ which correspond to P (1) and adding each of these species to the term (2). If the new species I is already present in the machine ($I \in \text{dom}(S)$) its population is incremented in S and the affected reactions are updated (4). If the species is not already present in the machine, its population is set to 1 in S and new reactions are computed using the function $reactions(I, \text{dom}(S))$, which is specific to the choice of calculus (5). Finally, the operation $T \ominus J$ removes the species J from the machine term T. This removing is done by decrementing the corresponding populations and by updating the affected reactions (3,6).

3. SIMULATION METHOD

This section explains the instantiation of the abstract machine for a chosen simulation method. We first tackle the Next Reaction Method [6]. We then present a new formalisation of the Next Reaction Method extended to the simulation of non-Markovian species, and show its instantiation within the generic abstract machine.

3.1 Next Reaction Method

The instantiation of the generic abstract machine for the

Next Reaction Method (NRM) is detailed in Definition 2. Each reaction O in R is mapped to a pair (a, t), where a is the propensity of the reaction and t is the putative time at which the reaction is scheduled to occur. The next reaction is chosen to be the one with the smallest putative time, as defined by the function $next(T)$, which returns the chosen reaction (J, F, f, J') together with its putative time t' (8).

When a new reaction is created, NRM computes the putative time of the reaction according to its propensity (9). NRM also provides a way to update putative times of Markovian reactions when their propensity changes, without generating a new random variable (10). When a new reaction is added to the machine, its propensity is computed and used to generate a random variable following the probability distribution of the reaction (9). The definition of the propensity for unary and binary reactions is given in Definition 2. Propensities for n-ary reactions can be defined if necessary. Markovian reactions are updated by computing the new propensity and rescaling the putative time (10). It may be that the old propensity is 0, preventing direct use of the rescaling function. This case can be handled by keeping additional variables to register the last non-null propensity and to rescale according to this old value (as discussed in note 11 of [6]). Similarly, if the new propensity is 0, the putative time is set to infinity.

The abstract machine can be readily instantiated to other reaction-based methods such as [7] by defining the appropriate $next$, $init$ and $updates$ functions, though we omit the details here.

3.2 Non-Markovian Next Reaction Method

Using non-Markovian distributions in NRM is equivalent to introducing a new species for every individual molecule with non-Markovian behaviour [6]. Since our abstract machine allows new species to be dynamically created, it can be used directly as the basis for simulating non-Markovian behaviour. The advantages of this approach include simplicity of the algorithm and its implementation. The disadvantage is that the simulation will be computationally expensive if there are large numbers of non-Markovian molecules. Algorithm 1 summarises the necessary steps for the simulation of mixed Markovian and non-Markovian species following the NRM. This algorithm formalises and extends the non-Markovian simulation algorithm discussed in [6].

The generic abstract machine is instantiated for the non-Markovian NRM by associating a unique identifier with each species that can participate in a non-Markovian reaction (Definition 3). The rate of a non-Markovian reaction is always 1 and its propensity is either 1 or 0, since there is at most one molecule of each species participating in the reaction. Generalised kinetic rate functions can be used, provided they are defined in terms of the interacting species. The putative times of non-Markovian reactions are updated by generating a new random variable, as in (9).

4. STOCHASTIC PI-CALCULUS SIMULATION

This section illustrates how the generic abstract machine can be used to simulate a variant of stochastic pi-calculus. In general, there will be many different ways of instantiating the generic abstract machine to a given process calculus, with broad scope for calculus-specific optimisations.

Algorithm 1 *Generalised Next Reaction Method*

1. *Initialise:*

   (a) *Set initial number of molecules, set time $\leftarrow$ 0, generate a dependency graph.*

   (b) *If $O_i \in \{Markovian\ reactions\}$, calculate a propensity function, a_i, generate a putative time, t_i, according to an exponential distribution with propensity a_i. Store t_i values.*

   (c) *If $O_i \notin \{Markovian\ reactions\}$. Set the propensity a_i to 1, store a putative time t_i, according to the general distribution.*

2. *Let O_μ be the reaction whose putative time, t_μ is least.*

3. *Change the number of molecules to reflect execution of reaction O_μ. Set time $\leftarrow t_\mu$.*

4. *Initialise new reactions (step 1.(b) and 1.(c)).*

5. *For each affected reaction O_α,*

   (a) *Update propensity a_α.*

   (b) *If $\alpha \neq \mu$ and $O_\alpha \in \{Markovian\ reactions\}$, set $t_\alpha \leftarrow (a_{\alpha,old}/a_{\alpha,new})(t_\alpha - time) + time$.*

   (c) *If $\alpha \neq \mu$ and $O_\alpha \notin \{Markovian\ reactions\}$, if $a_{\alpha,new} = 0$ then remove the reaction O_α and its putative time t_α.*

   (d) *If $\alpha = \mu$ and $O_\alpha \in \{Markovian\ reactions\}$, generate a random number, ρ, according to an exponential distribution with propensity a_α, and set $t_\alpha \leftarrow \rho + time$.*

   (e) *If $\alpha = \mu$ and $O_\alpha \notin \{Markovian\ reactions\}$, remove the reaction O_α and its putative time t_α.*

6. *Go to Step 2*

$$
\begin{aligned}
species'(P) &\triangleq renames(species(P)) \\
next'(t, S, R) &\triangleq (J, F, f, rename(J')), t' \\
&\quad \textbf{if } (J, F, f, J'), t' = next(t, S, R) \\
rename(J) &\triangleq \{rename(I) \mid I \in J\} \\
rename(I) &\triangleq \textbf{if } NM(I) \textbf{ then } I^{(fresh(ctr))} \textbf{ else } I^0 \\
rename(I^{id}) &\triangleq rename(I)
\end{aligned}
$$

DEFINITION 3. *Generic abstract machine instantiated for the Non-Markovian Next Reaction Method.* The functions *species* and *next* from Definition 1 are replaced with the functions *species'* and *next'*, respectively, to allow for the renaming of species. The function *rename* renames a species by labelling it with a unique identifier. The function *fresh(ctr)* increments a global counter referenced by *ctr* and returns the incremented value. The function $NM(I)$ returns true if species I can participate in at least one non-Markovian reaction and returns false otherwise.

We illustrate this point by discussing optimisations that are specific to stochastic pi-calculus.

4.1 Syntax of Processes and Complexes

The syntax of the variant of stochastic pi-calculus used in this paper is given in Definition 4 and is based on [13]. Processes can evolve by performing delay actions or by interacting with each other over shared channels. A process can evolve on its own by executing a delay τ_r. Two processes can evolve simultaneously by communicating or binding with each other. A communication between two processes is achieved when one process sends data $\tilde{n}$ on a channel x, denoted by $!x(\tilde{n})$, and a parallel process receives this data on the same channel x, denoted by $?x(\tilde{m})$. A binding between two processes can occur if one process sends private data $\nu\tilde{n}$ on a channel x, denoted by $!x(\nu\tilde{n})$, which is then shared only between the sender and receiver, representing the formation of a complex between the two.

The calculus is stochastic because the duration of delays and interactions is determined by a random variable. In the case of Markovian reactions, the random variable follows an exponential distribution parameterised by the so-called *reaction rate*. In the general case, the random variables can follow an arbitrary probability distribution associated with the reaction. Thus we associate a probability distribution F_x with each channel x, where $F_x(t)$ is the probability of firing the reaction after t time units. Similarly, we associate a probability distribution F_r with each delay τ_r.

The structural congruence axioms in the stochastic pi-calculus are defined in Definition 5, and the reduction rules are described in Definition 6. The rules assume that each unguarded action π in the system is associated with a unique identifier i. This allows each reduction to be associated with a unique index w, composed of either a single identifier i denoting a delay, or a pair of identifiers (i_1, i_2) denoting a communication. Since each identifier or pair of identifiers is unique, this allows the total number of distinct reactions in the system to be counted.

4.2 From Processes to Reactions

This section instantiates the generic abstract machine to simulate the stochastic pi-calculus with general distributions. The first step is to define what constitutes a species. Here we assume that a species is either an instance $X(\tilde{n})$ or a complex of instances $\nu\tilde{n}\,((X_1(\tilde{n}_1) \mid \ldots \mid X_M(\tilde{n}_M))$, where each instance corresponds to a choice of actions. Our approach is motivated by the observation that a choice of actions is the basic unit of computation, where two parallel choices interact by communicating over shared channels. An alternative approach could be to assume that a species corresponds directly to a choice of actions, instead of using a named instance $X(\tilde{n})$. Our decision to use a named instance has the advantage that a species can be explicitly identified in a biological model by a meaningful name, and that the results of a simulation can be directly linked to the original model via this name. In order to formalise the notion of a species in stochastic pi-calculus, we define a normal form for processes (Definition 7) and show that all processes are structurally congruent to a normal form (Proposition 1).

PROPOSITION 1. All processes of the stochastic pi-calculus are structurally congruent to a normal form according to Definition 7.

$$P ::= \qquad \mathbf{0} \qquad\qquad \text{Null}$$
$$| \qquad X(\tilde{n}) \qquad\qquad \text{Instance}$$
$$| \qquad P_1 \mid P_2 \qquad\qquad \text{Parallel}$$
$$| \qquad \nu x\, P \qquad\qquad \text{Restriction}$$
$$| \qquad \pi_1.P_1 + \ldots + \pi_N.P_N \qquad \text{Choice}$$
$$E ::= \quad X_1(\tilde{m}_1) \mapsto P_1, \ldots, X_N(\tilde{m}_N) \mapsto P_N \quad \text{Environment}$$

$$\pi ::= \quad \tau_r \qquad \text{Delay}$$
$$| \quad !x(\tilde{n}) \qquad \text{Send}$$
$$| \quad !x(\nu\tilde{m}\,) \qquad \text{Bind}$$
$$| \quad ?x(\tilde{m}) \qquad \text{Receive}$$

DEFINITION 4. *Syntax of stochastic pi-calculus.* For each definition $X(\tilde{m}) = P$ in the environment, we assume that $\tilde{m} \subseteq \text{fn}(P)$, where $\text{fn}(P)$ denotes the free names of P. The restriction $\nu x\, P$ binds the name x in P and both $!x(\nu\tilde{m}).P$ and $?x(\tilde{m}).P$ bind names $\tilde{m}$ in P. We also assume that recursive calls to a definition are guarded inside an action prefix π, to prevent infinite expansion of process definitions.

$$P \mid \mathbf{0} \equiv P \qquad (11)$$
$$P_1 \mid P_2 \equiv P_2 \mid P_1 \qquad (12)$$
$$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3 \qquad (13)$$
$$\pi_1.P_1 + \pi_2.P_2 \equiv \pi_2.P_2 + \pi_1.P_1 \qquad (14)$$
$$\nu x\, \mathbf{0} \equiv \mathbf{0} \qquad (15)$$
$$\nu x\, \nu y\, P \equiv \nu y\, \nu x\, P \qquad (16)$$
$$\nu x\, (P_1 \mid P_2) \equiv P_1 \mid \nu x\, P_2 \text{ if } x \notin \text{fn}(P_1) \qquad (17)$$
$$X(\tilde{n}) \equiv P_{\{\tilde{m}:=\tilde{n}\}} \text{ if } E(X(\tilde{m})) = P \qquad (18)$$

DEFINITION 5. *Structural congruence axioms in the stochastic pi-calculus, assuming a global environment E.* Structural congruence is reflexive, symmetric and transitive, and holds in any context inside a process or a choice.

$$\tau_r.P + C \xrightarrow{F_{r,i}} P$$
$$!x(\tilde{n}).P_1 + C_1 \mid ?x(\tilde{m}).P_2 + C_2 \xrightarrow{F_x,(i_1,i_2)} P_1 \mid P_{2\{\tilde{m}:=\tilde{n}\}}$$
$$!x(\nu\tilde{n}).P_1 + C_1 \mid ?x(\tilde{m}).P_2 + C_2 \xrightarrow{F_x,(i_1,i_2)} \nu\tilde{n}(P_1 \mid P_{2\{\tilde{m}:=\tilde{n}\}})$$
$$P \xrightarrow{F,w} P' \quad\Rightarrow\quad \nu x\, P \xrightarrow{F,w} \nu x\, P'$$
$$P \xrightarrow{F,w} P' \quad\Rightarrow\quad P \mid Q \xrightarrow{F,w} P' \mid Q$$
$$Q \equiv P \xrightarrow{F,w} P' \equiv Q' \quad\Rightarrow\quad Q \xrightarrow{F,w} Q'$$

DEFINITION 6. *Reduction in the stochastic pi-calculus.* We assume that each reduction is associated with a unique index w, composed of either a single identifier i denoting a delay, or a pair of identifiers (i_1, i_2) denoting a communication.

$$P ::= \qquad I_1 \mid \ldots \mid I_N \qquad\qquad \text{Species}$$
$$I ::= \qquad X(\tilde{n}) \qquad\qquad \text{Instance}$$
$$| \qquad \nu\tilde{z}\,((X_1(\tilde{n}_1) \mid \ldots \mid X_M(\tilde{n}_M)) \qquad \text{Complex}$$
$$C ::= \qquad \pi_1.P_1 + \ldots + \pi_N.P_N \qquad \text{Choice}$$
$$E ::= \quad X_1(\tilde{m}_1) \mapsto C_1, \ldots, X_N(\tilde{m}_N) \mapsto C_N \quad \text{Environment}$$

DEFINITION 7. *Normal forms of stochastic pi-calculus processes, where $N \geq 0$ and $M \geq 1$.* A process P is considered to be in normal form if it consists of a parallel composition of species I, where a species can be an instance $X(\tilde{n})$ or a complex of instances $\nu\tilde{z}\,((X_1(\tilde{n}_1) \mid \ldots \mid X_M(\tilde{n}_M))$ and where every instance $X(\tilde{n})$ corresponds to a choice of actions. We assume that $\tilde{z} \cap \tilde{n}_1 \cap \ldots \cap \tilde{n}_M \neq \emptyset$ and $\tilde{z} \subseteq \tilde{n}_1 \cup \ldots \cup \tilde{n}_M$, so as to minimise the scope of restricted names.

$$normal(\mathbf{0}) \triangleq \mathbf{0}$$
$$normal(X(\tilde{n})) \triangleq X(\tilde{n}) \text{ if } E(X(\tilde{n})) = C$$
$$normal(X(\tilde{n})) \triangleq normal(P) \text{ if } E(X(\tilde{n})) = P \neq C$$
$$normal(P_1 \mid P_2) \triangleq normal(P_1) \mid normal(P_2)$$
$$normal(C) \triangleq X(\tilde{n}) \text{ if } E(X(\tilde{n})) = C$$
$$normal(\nu x\, P) \triangleq insert(x, normal(P))$$
$$insert(x, \textstyle\prod_i I_i) \triangleq (\nu\tilde{z} \textstyle\prod_k K_k) \mid \textstyle\prod_j I_j$$
$$\qquad \textbf{if } I_k = \nu\tilde{z}_k\, K_k$$
$$\qquad \textbf{and } x \in \text{fn}(I_k), x \notin \text{fn}(I_j)$$
$$\qquad \textbf{and } \bigcap \tilde{z}_k = \emptyset, \tilde{z} = \{x\} \cup \bigcup \tilde{z}_k$$
$$\qquad \textbf{and } i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}$$
$$\qquad \textbf{and } \mathcal{J} \cap \mathcal{K} = \emptyset, \mathcal{I} = \mathcal{J} \cup \mathcal{K}$$

DEFINITION 8. *Computing the normal form of a stochastic pi-calculus process.* We write $\prod_i P_i$ as short for $P_1 \mid \ldots \mid P_N$, assuming $i \in \{1, \ldots, N\}$.
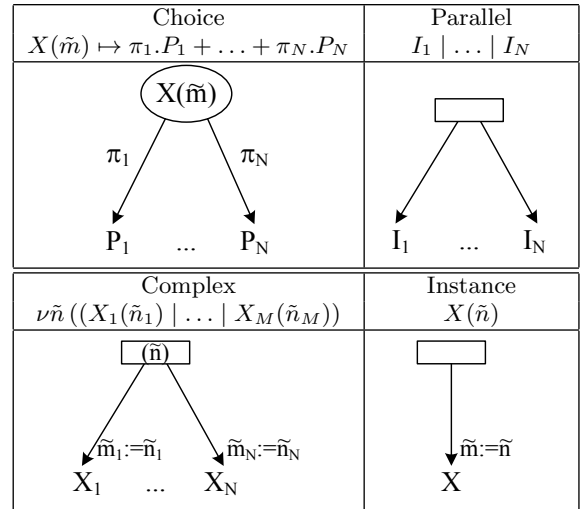
DEFINITION 9. *Graphical representation for the stochastic pi-calculus, based on the normal form of Definition 7.* For each instance $X_i(\tilde{n}_i)$ there is assumed to be a corresponding definition $X_i(\tilde{m}_i) \mapsto C_i$ in the process environment.

If $E(X_1(\tilde{n}_1)) = \kappa_1.P_1 + \ldots + \kappa_n.P_n$
and $E(X_2(\tilde{n}_2)) = \lambda_1.Q_1 + \ldots + \lambda_m.Q_m$ then

$$expand(X_1(\tilde{n}_1) \mid X_2(\tilde{n}_2)) \triangleq \sum_i \kappa_i.(P_i \mid X_2(\tilde{n}_2))$$
$$+ \sum_j \lambda_j.(X_1(\tilde{n}_1) \mid Q_j)$$
$$+ \sum_{\kappa_i \mathrm{comp}\lambda_j} \tau_{r_{ij}}.R_{ij}$$

where $\kappa_i \mathrm{comp}\lambda_j$ (κ_i complements λ_j) iff:

1. κ_i is $!x(\tilde{n})$ and λ_j is $?x(\tilde{m})$ when R_{ij} is $P_i \mid Q_{j\ \{\tilde{m}:=\tilde{n}\}}$ and r_{ij} is F_x, and

2. κ_i is $!x(\nu\tilde{n})$ and λ_j is $?x(\tilde{m})$ when R_{ij} is $\nu\tilde{n}(P_i \mid Q_{j\ \{\tilde{m}:=\tilde{n}\}})$ and r_{ij} is F_x.

DEFINITION 10. *Expansion of a choice, based on standard principles presented in [21]. We write $E(X(\tilde{n})) = C$ as an abbreviation for $E(X(\tilde{m})) = C'$ where $C = C'_{\{\tilde{m}:=\tilde{n}\}}$.*

PROOF. By induction on Definition 8. Using the structural congruence rules of Definition 5, we augment the environment such that all choices are defined separately (18), and we replace all instances that are not a choice with their corresponding process definition (18). Using the structural congruence rule for scoping (17), we modify the scope of a restriction such that a process is a parallel composition of species, where each species is either an instance or a complex. $\square$

The normal form can also be used as the basis for a graphical representation (Definition 9), following the approach of [15], such that there is a one-to-one correspondence between the graphics and programs in normal form. In Sec. 5 the graphical representation is used to construct a stochastic pi-calculus model of plasmid co-transfection.

Using our normal form for processes (Definition 7), we now define the various functions that are needed to instantiate the generic abstract machine for stochastic pi-calculus (Definition 11). The function $reactions(I, J)$ computes the set of reactions that the species I can perform with the set of species J. The set of reactions is given by the set of unary reactions (delays) combined with the set of binary reactions (communications and bindings). The function $actions(I)$ converts a species to a choice. An instance $X(\tilde{n})$ is converted to a choice corresponding to the species definition, while a complex $\nu\tilde{n}((X_1(\tilde{n}_1) \mid \ldots \mid X_M(\tilde{n}_M))$ is converted to a choice by first expanding the parallel composition $(X_1(\tilde{n}_1) \mid \ldots \mid X_M(\tilde{n}_M))$ to a single choice C (Definition 10), and then reducing the scope of the restrictions $\nu\tilde{n} C$ inside the choice (Definition 11).

4.3 Example

We illustrate the application of the generic abstract machine to the stochastic pi-calculus with a simple example of complex formation:

$$A = !x(\nu u).AB(u) \qquad AB(u) = !u.A$$
$$B = ?x(u).BA(u) \qquad BA(u) = ?u.B$$

Initially, 100 copies of processes A and B are added to the empty machine term, written $(100 \cdot A \mid 100 \cdot B) \oplus (0, \emptyset, \emptyset)$,

---

$$species(P) \triangleq \{I_1, \ldots, I_N\}$$
$$\text{if } normal(P) = (I_1 \mid \ldots \mid I_N)$$
$$reactions(I, J) \triangleq unary(I) \cup binary(I, J)$$
$$unary(I) \triangleq \{(\{I\}, F, id, J)$$
$$\mid (F, J) \in delays(actions(I))\}$$
$$delays(C) \triangleq \{(F_r, species(P)) \mid \tau_r.P \in C\}$$
$$binary(I_1, J) \triangleq \{(\{I_1, I_2\}, F, id, J')$$
$$\mid C_1 = actions(I_1) \wedge C_2 = actions(I_2)$$
$$\wedge I_2 \in \{I\} \cup J$$
$$\wedge (F, J') \in interact(C_1, C_2)\}$$
$$interact(C_1, C_2) \triangleq comm(C_1, C_2) \uplus bind(C_1, C_2)$$
$$comm(C_1, C_2) \triangleq \{(F_x, species(P_1 \mid P_{2\{\tilde{m}:=\tilde{n}\}}))$$
$$\mid !x(\tilde{n}).P_1 \in C_1 \wedge ?x(\tilde{m}).P_2 \in C_2$$
$$\vee !x(\tilde{n}).P_1 \in C_2 \wedge ?x(\tilde{m}).P_2 \in C_1\}$$
$$bind(C_1, C_2) \triangleq \{(F_x, species(\nu\tilde{n}(P_1 \mid P_{2\{\tilde{m}:=\tilde{n}\}})))$$
$$\mid !x(\nu\tilde{n}).P_1 \in C_1 \wedge ?x(\tilde{m}).P_2 \in C_2$$
$$\vee !x(\nu\tilde{n}).P_1 \in C_2 \wedge ?x(\tilde{m}).P_2 \in C_1\}$$

$$actions(X(\tilde{n})) \triangleq C_{\{\tilde{m}:=\tilde{n}\}} \text{ if } C = E(X(\tilde{m}))$$
$$actions(\nu\tilde{n}\prod_i X_i(\tilde{n}_i)) \triangleq actions(\nu\tilde{n}\, expand(\prod_i X_i(\tilde{n}_i)))$$
$$actions(\nu\tilde{n}\sum_i \pi_i.P_i) \triangleq \sum_i action(\nu\tilde{n}\,\pi_i.P_i)$$
$$action(\nu\tilde{n}\,\tau_r.P) \triangleq \tau_r.\nu\tilde{n}\,P$$
$$action(\nu\tilde{n}\,!x(\tilde{m}).P) \triangleq !x(\tilde{m}).\nu\tilde{n}\,P \text{ if } \tilde{n} \cap (\tilde{m} \cup \tilde{x}) = \emptyset$$
$$action(\nu\tilde{n}\,!x(\tilde{m}).P) \triangleq !x(\nu\tilde{m}).\nu(\tilde{n} \setminus \tilde{m})\,P \text{ if } x \notin \tilde{n}$$
$$\text{and } \tilde{m} \subseteq \tilde{n}$$
$$action(\nu\tilde{n}\,!x(\nu\tilde{m}).P) \triangleq !x(\nu\tilde{m}).\nu(\tilde{n} \setminus \tilde{m})\,P \text{ if } x \notin \tilde{n}$$
$$action(\nu\tilde{n}\,?x(\tilde{m}).P) \triangleq ?x(\tilde{m}).\nu(\tilde{n} \setminus \tilde{m})\,P \text{ if } x \notin \tilde{n}$$

DEFINITION 11. *Generic abstract machine instantiated for the stochastic pi-calculus. We assume a fixed global environment E containing all instance definitions. $actions(I)$ converts a species to a choice, $actions(\nu\tilde{n}\,C)$ reduces the scope of restrictions inside a choice, $action(\nu\tilde{n}\,\pi.P)$ reduces the scope of restrictions inside a single action, and $\tilde{n} \setminus \tilde{m}$ removes the names $\tilde{m}$ from $\tilde{n}$. If $x \in \tilde{n}$ and $\pi \in \{!x(\tilde{m}), !x(\nu\tilde{m}), ?x(\tilde{m})\}$ then $action(\nu\tilde{n}\,\pi.P)$ gives rise to the empty process $\mathbf{0}$. $expand(X_1(\tilde{n}_1) \mid \ldots \mid X_M(\tilde{n}_M))$ converts a parallel composition of instances to a single choice (Definition 10), and $normal(P)$ converts a process P to normal form (Definition 7). We write $\sum_i \pi_i.P_i$ as short for $\pi_1.P_1 + \ldots + \pi_N.P_N$ and $\prod_i P_i$ as short for $P_1 \mid \ldots \mid P_N$, assuming $i \in \{1, \ldots, N\}$. In this example, the function f is always the identity function (written as id).*

---

where the notation $100 \cdot X$ represents 100 parallel copies of the process X. This gives rise to the following machine term $(0, S, R)$:

$$S = \{A \mapsto 100, B \mapsto 100\}$$
$$R = \{(\{A, B\}, F_x, \{\nu u\,(AB(u) \mid BA(u))\}) \mapsto (10^4 \cdot rate(F_x), t_1)\}$$

The reaction involving species A and B is executed at time t_1, after which one copy of the species A and B are removed and one copy of the complex is added to the resulting machine term:

$$\nu u \left(AB(u) \mid BA(u)\right) \oplus ((t_1, S, R) \ominus \{A, B\})$$

This gives rise to the machine term (t_1, S_1, R_1), where $\text{actions}(\nu u(AB(u) \mid BA(u)))$ is given by $\tau_u.(A \mid B)$:

$S_1 = \{A \mapsto 99, B \mapsto 99, \nu u\left(AB(u) \mid BA(u)\right) \mapsto 1\}$

$R_1 =$
$\{(\{A, B\}, F_x, \{\nu u\left(AB(u) \mid BA(u)\right)\}) \mapsto (9801 \cdot rate(F_x), t_3),$
$(\{\nu u\left(AB(u) \mid BA(u)\right)\}, F_u, \{A, B\}) \mapsto (rate(F_u), t_2)\}$

Note that existing simulation algorithms such as [13] handle N copies of the complex $\nu u(AB(u) \mid BA(u))$ by creating a globally fresh name for each restricted channel u as follows:

$$\nu u_1 \ldots \nu u_N \left(AB(u_1) \mid BA(u_1) \mid \ldots \mid AB(u_N) \mid BA(u_N)\right)$$

In contrast, our approach treats these as N copies of the same complex $\nu u(AB(u) \mid BA(u))$, resulting in fewer species, fewer reactions and therefore a significantly more efficient simulation.

4.4 Correctness of the Simulation

We briefly outline a proof of correctness of the instantiated abstract machine (SPiM) with respect to the stochastic pi-calculus with general distributions (SPi) presented in Definition 4. The function $(|E \vdash P|)$ encodes a *system* $E \vdash P$ in SPi, consisting of a top-level environment E and a process P, to a corresponding system in SPiM (Definition 12). The encoding assumes that all processes are in normal form. A corresponding decoding from SPiM to SPi is also given (Definition 13), where the unique integers which tag individual non-Markovian molecules are discarded. Proposition 2 and Proposition 3 ensure that the calculus and the machine are reduction equivalent, where w and w' stand for reduction identifiers in SPi and SPiM, respectively. In order to preserve the correspondence, we assume a notion of structural congruence for machine terms, where terms are structurally congruent up to renaming of definitions, garbage-collection of unused definitions and structural congruence of processes.

PROPOSITION 2. $\forall E, T \in$ SPiM. $E \vdash T \xrightarrow{F_\theta, w'} E \vdash T' \Rightarrow$ $[|E \vdash T|] \xrightarrow{F_\theta, w} [|E \vdash T'|]$

PROOF. By induction on the derivation of reduction in SPiM. □

PROPOSITION 3. $\forall E, P \in$ SPi. $E \vdash P \xrightarrow{F_\theta, w} E \vdash P' \Rightarrow$ $(|E \vdash P|) \xrightarrow{F_\theta, w'} \equiv (|E \vdash P'|)$

PROOF. By induction on the derivation of reduction in SPi. □

It is worth noticing that the probability of a transition $P \xrightarrow{F_\theta, w} P'$ does not necessarily follow the distribution F_θ in the general case. Let us assume that the following transitions are correct in SPi and follow non-Markovian distributions: $P \xrightarrow{F_1, w_1} P_1$, $P \xrightarrow{F_2, w_2} P_2$, $P_1 \xrightarrow{F_2, w_2} P_2$. The probability of the last transition in the sequence $P \xrightarrow{F_1, w_1} P_1 \xrightarrow{F_2, w_2} P_2$ does not follow F_2 but a modified distribution which takes into account the probability distribution F_1. The relabelling

$$(|E \vdash P|) \quad \triangleq \quad E \vdash P \oplus (0, \emptyset, \emptyset)$$

DEFINITION 12. *Encoding a system from* SPi *to* SPiM.

$$
\begin{aligned}
[|E \vdash T|] &\triangleq E \vdash [|T|] \\
[|t, S, R|] &\triangleq [|S|] \\
[|\emptyset|] &\triangleq \mathbf{0} \\
[|S, (I \mapsto i)|] &\triangleq \underbrace{I \mid \ldots \mid I}_{i} \mid [|S|] \\
[|S, (I^{id} \mapsto i)|] &\triangleq \underbrace{I \mid \ldots \mid I}_{i} \mid [|S|]
\end{aligned}
$$

DEFINITION 13. *Decoding a system from* SPiM *to* SPi. The environment E is unchanged, and for each mapping $I \mapsto i$ in S, i copies of the species are executed in parallel.

of a sequence of transitions $P_0 \xrightarrow{F_0, w_0} P_1 \xrightarrow{F_1, w_1} \ldots \xrightarrow{F_n, w_n} P_{n+1}$ into $P_0 \xrightarrow{F'_0, w_0} P_1 \xrightarrow{F'_1, w_1} \ldots \xrightarrow{F'_n, w_n} P_{n+1}$ where F'_i are the correct distribution functions of each transition is tackled in [17]. It is clear from Proposition 2 and Proposition 3 that to each sequence of transitions in SPi corresponds to a sequence of transitions in SPiM having the same F_i in labels, and vice-versa. It follows that the probabilities of transitions are equivalent in SPi and in SPiM.

5. APPLICATION: A MODEL OF PLASMID CO-TRANSFECTION

In this section we present a model of gene transfer by plasmid co-transfection[1] involving non-Markovian reactions, derived from [23]. We model complex formation for green and red plasmids, together with the main stages of co-transfection (Fig. 1). The process $C(g, r)$ represents a complex of g green plasmids and r red plasmids, where g, r are numbers. A complex can grow in size by receiving the numbers g', r' on channel *bind* and adding these to g, r respectively. Alternatively, it can bind to another complex by sending the numbers g, r on channel *bind*. At any stage a complex $C(g, r)$ can enter the cell, represented by an *enter* reaction to $DC(g, r)$. In this model, the rate of entry is proportional to the square of the size of the complex, where the size is given by the total number of red and green plasmids $(g + r)$. The rate of entry relates to the charge of the molecule, where larger complexes have increased charge and are better able to penetrate the cell wall. Once translocation has occurred, the resulting complex of plasmids ENC can dissociate into individual green (ENG) or red (ENR) plasmids, one at a time. We model this using an *unbind* reaction, which removes a red or green plasmid from the complex. The unbinding rate is proportional to the number of red or green plasmids, respectively. The gene expression of plasmids involves the transcription of plasmids into mRNA and the transla-
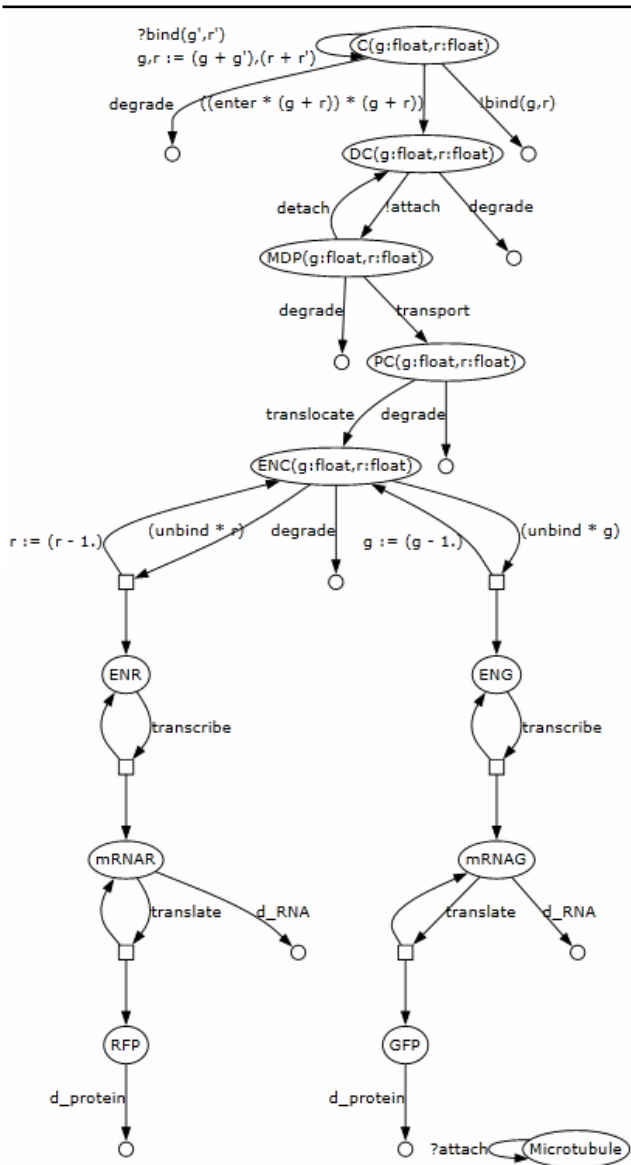
Figure 1: A stochastic pi-calculus model of plasmid co-transfection.

Figure 2: Simulation results of the plasmid co-transfection model of Fig. 1, where the horizontal axis represents time in hours and the vertical axis represents numbers of molecules.

Figure 3: Simulation of the initial entry of complexes into the cell. We simulated the initial stages of the model of Fig. 1, starting with 1000 individual red and green plasmids and allowing these plasmids to form complexes before entering the cell. We let DC(g,r) = () to prevent further movement of the complexes and plot the composition of plasmids DC(g,r) immediately after entry. We used a 3D plot where the x axis represents the number of green plasmids in the complex, the y-axis represents the number of red plasmids in the complex and the height represents the number of complexes with the given composition of red and green plasmids. The largest complex contained 11 red and 9 green plasmids, but the majority of complexes contained less than 10 plasmids.

tion of mRNA into proteins. The degradation of mRNA is a 170-step process which we model as a single reaction with an Erlang distribution. The green plasmids produce green fluorescent proteins (GFP), while the red plasmids produce red fluorescent proteins (RFP). The SPiM code for the model is given in Appendix A.

Fig. 2 shows the results of simulating the model of Fig. 1, using the generic abstract machine instantiated with the non-Markovian stochastic pi-calculus. The model parameters are given in Appendix A. The individual plasmids stochastically bind together to form complexes of different sizes, which then enter the cell and move towards the nucleus. Entire complexes can be degraded while in transit. Once they reach the nucleus the complexes unbind, releasing their plasmid cargo, which is then transcribed to produce red or green
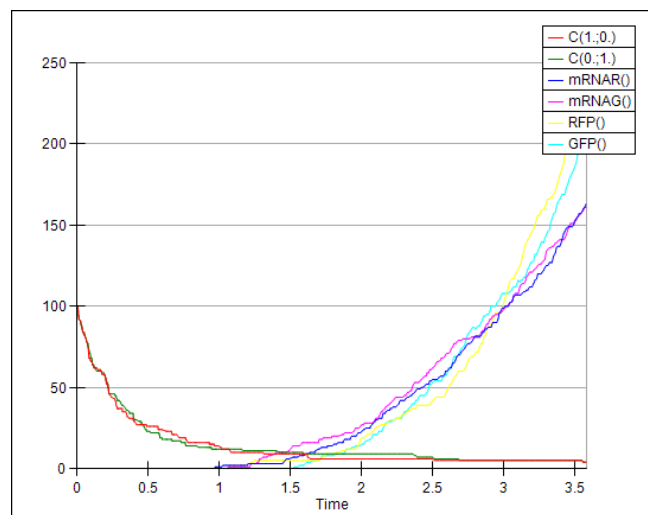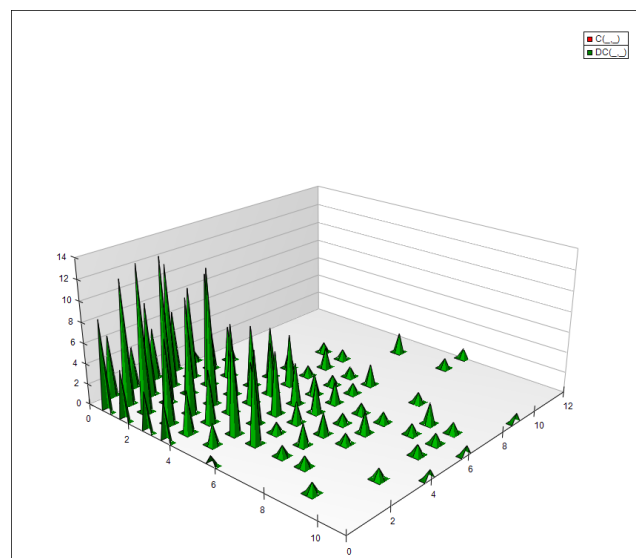
fluorescent proteins. In order to visualise the proportion of complexes of different sizes, we can plot the complexes immediately after entry into the cell (Fig. 3). In general the complexes can be of arbitrary size, depending on the initial populations of plasmids. Control over the ratio between transfected plasmids is a key requirement for the development of novel therapeutic strategies that act in the tissue on the gene level. Here stochasticity plays an important role. Future analysis of the model can be used as a basis for determining optimal co-transfection strategies that result in specific ratios of red and green fluorescent proteins inside individual cells.

6. DISCUSSION

We have introduced a generic abstract machine for the simulation of process calculi with potentially unbounded numbers of species and reactions. Instantiating the machine for a particular calculus requires the definition of functions to extract the species and reactions from the processes of the calculus. We have detailed the simulation of the non-Markovian stochastic pi-calculus using this abstract machine, and have presented a model of plasmid co-transfection that illustrates the flexibility of the proposed framework.

To cope with a large number of reactions when simulating non-Markovian processes with a large number of species, the implementation of the generic abstract machine should use optimised data structures to quickly access reactions affected by propensity changes. This requires either computing an explicit dependency graph between reactions, as suggested in [6], or having efficient hashing structures to access reactions in the machine term. For the Markovian case, we have characterised the improvement in simulation efficiency by the ability to count complexes that share private channels as a single species. More practical comparisons such as benchmark comparisons with existing stochastic pi-calculus simulators are left for future work.

Little work has so far been done to provide efficient non-Markovian simulations. In [1], the Gillespie algorithm is extended to delayed reactions, allowing non-Markovian simulation of these types of reactions. In the scope of process calculi, Priami has formalised the semantics of the stochastic pi-calculus models with general distributions [17]. The simulation proposed in [17] is based on automatic rescaling of the general distribution functions of transitions during the execution, in order to reflect the history of the execution, as needed for non-Markovian reactions. This strategy has been adopted and implemented to allow non-Markovian simulation of BlenX models [10, 16]. While rescaling distribution functions is manageable for distributions like Gamma or Hyper-exponential, more general distributions may be harder to compute. The non-Markovian simulation algorithm we propose does not rely on such a rescaling of distribution functions and therefore provides an efficient and straightforward simulation of reactions with arbitrary probability distributions. The algorithm relies on the fact that our generic abstract machine allows a potentially unbounded number of new species to be dynamically generated.

Our generic abstract machine aims at simulating a broad range of process calculi. This range includes process calculi capable of n-ary reactions, and reactions having arbitrary stoichiometric coefficients. To highlight the flexibility of our approach, we have used our machine to simulate a

variant of the DSD calculus [14][2]. As a proof-of-principle, we have also instantiated the generic abstract machine to a variant of the stochastic bioambient calculus [12]. Since this calculus relates processes that may move between different ambients, it was necessary to extend the encoding of species to correctly translate interactions as "flat" reactions. This instantiation required the addition of a species renaming operator in the generic abstract machine, and is detailed in Appendix B. Although the instantiation can be significantly improved through further optimisations, this nevertheless produces the first simulation algorithm for a non-Markovian stochastic bioambient calculus. Further refinements to the algorithm are left for future work.

Our approach can potentially be used to simulate a range of existing process calculi within the same framework. In future, this could allow models to be constructed from components written in different domain-specific languages, allowing exact stochastic simulation of heterogeneous systems. Our approach could also aid the development of future programming languages and calculi, by reducing the overhead for implementing custom stochastic simulation algorithms.

7. REFERENCES

[1] D. Bratsun, D. Volfson, L. S. Tsimring, and J. Hasty. Delay-induced stochastic oscillations in gene regulation. *Proceedings of the National Academy of Sciences of the United States of America*, 102(41):14593–14598, 2005.

[2] F. Ciocchetta and J. Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065 – 3084, 2009.

[3] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. *CONCUR 2007 - Concurrency Theory*, chapter Rule-Based Modelling of Cellular Signalling, pages 17–41. 2007.

[4] L. Dematté, C. Priami, and A. Romanel. Modelling and simulation of biological processes in BlenX. *SIGMETRICS Performance Evaluation Review*, 35(4):32–39, 2008.

[5] R. Ewald, J. Himmelspach, M. Jeschke, S. Leye, and A. M. Uhrmacher. Flexible experimentation in the modeling and simulation framework JAMES II – implications for computational systems biology. *Brief Bioinform*, Jan 2010.

[6] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[7] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[8] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716–1733, 2001.

[9] P. Lecca and C. Priami. Cell cycle control in eukaryotes: a BioSPI model. In *BioConcur'03*. ENTCS, 2003.

---

[2]Simulator available at http://research.microsoft.com/dna

[10] I. Mura, D. Prandi, C. Priami, and A. Romanel. Exploiting non-Markovian Bio-Processes. *Electr. Notes Theor. Comput. Sci.*, 253(3):83–98, 2009.

[11] M. Pedersen and G. Plotkin. A language for biochemical systems. In *Computational Methods in Systems Biology*, volume 5307 of *LNCS*, pages 63–82. Springer, 2008.

[12] A. Phillips. An abstract machine for the stochastic bioambient calculus. *Electronic Notes in Theoretical Computer Science*, 227:143–159, January 2009.

[13] A. Phillips and L. Cardelli. Efficient, correct simulation of biological processes in the stochastic pi-calculus. In *Computational Methods in Systems Biology*, volume 4695 of *LNCS*, pages 184–199. Springer, September 2007.

[14] A. Phillips and L. Cardelli. A programming language for composable DNA circuits. *Journal of the Royal Society Interface*, 6(S4):419–436, August 2009.

[15] A. Phillips, L. Cardelli, and G. Castagna. A graphical representation for biological processes in the stochastic pi-calculus. *Transactions in Computational Systems Biology*, 4230:123–152, November 2006.

[16] D. Prandi, C. Priami, and A. Romanel. Simulation of Non-Markovian Processes in BlenX. Technical Report TR-11-2008, The Microsoft Research - University of Trento Centre for Computational and Systems Biology, 2008.

[17] C. Priami. Stochastic π-calculus with general distributions. In *Proc. of the 4th Workshop on Process Algebras and Performance Modelling, CLUT*, pages 41–57, 1996.

[18] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.

[19] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Y. Shapiro. Bioambients: an abstraction for biological compartments. *Theor. Comput. Sci.*, 325(1):141–167, 2004.

[20] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, volume 6, pages 459–470, Singapore, 2001. World Scientific Press.

[21] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[22] T. Tian and K. Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *J. Chem. Phys.*, 121:10356–10364, 2004.

[23] C. M. Varga, K. Hong, and D. A. Lauffenburger. Quantitative analysis of synthetic gene delivery vector design properties. *Mol Ther*, 4(5):438–446, Nov 2001.

APPENDIX

A. SPIM CODE FOR PLASMID CO-TRANSFECTION

```
directive sample 4.0 1000
val enter = 0.1
val degrade = 0.01
```

```
val detach = 1.0
val transport = 1.0
val translocate = 1.0
val unbind = 1.0

val transcribe = 4.0
val translate = 1.5
val d_RNA = 0.466
val d_protein = 0.019

new bind@0.01:chan(float,float)
new attach@1.0:chan
new c@1.0:chan

let C(g:float,r:float) =
  do delay@enter*(g+r)*(g+r); DC(g,r)
  or !bind(g,r)
  or ?bind(g',r'); C(g+g',r+r')
  or delay@degrade
and DC(g:float,r:float) =
  do !attach; MDP(g,r)
  or delay@degrade
and ENC(g:float,r:float) =
  do delay@degrade
  or delay@unbind*g; (ENG() | ENC(g-1.0,r))
  or delay@unbind*r; (ENR() | ENC(g,r-1.0))
and MDP(g:float,r:float) =
  do delay@detach; DC(g,r)
  or delay@transport; PC(g,r)
  or delay@degrade; ()
and PC(g:float,r:float) =
  do delay@translocate; ENC(g,r)
  or delay@degrade; ()
and Microtubule() =  ?attach; Microtubule()

and ENG() = delay@transcribe; (ENG() | mRNAG())
and ENR() = delay@transcribe; (ENR() | mRNAR())
and mRNAG() =
  do delay@translate; (mRNAG() | GFP())
  or delay@Erlang(170,d_RNA)
and GFP() = delay@d_protein
and mRNAR() =
  do delay@translate; (mRNAR() | RFP())
  or delay@Erlang(170,d_RNA)
and RFP() = delay@d_protein

run 100 of C(1.0,0.0)
run 100 of C(0.0,1.0)
run 100 of Microtubule()
```

B. SIMULATION OF THE STOCHASTIC BIOAMBIENT CALCULUS

The bioambient calculus was first presented in [19] for modelling mobile compartments in biological processes. This appendix describes how the bioambient calculus can be simulated using the generic abstract machine presented in the main text. Non-Markovian simulation can be achieved by means of Definition 3, producing the first non-Markovian simulation algorithm for the stochastic bioambient calculus.

B.1   Syntax and Reduction

The syntax and reduction rules of the stochastic bioambient calculus used in this section are presented in Definition 14 and are reproduced from [12]. A process P can be a choice of actions M, an instance $X(\tilde{n})$ of a definition X with parameters $\tilde{n}$, a parallel composition of processes $P \mid Q$, a process $\nu x\, P$ with a private channel x, or an ambient $\boxed{P}$ consisting of a process P inside a compartment. A choice M

$$P, Q ::= \quad M \mid X(\tilde{n}) \mid P|Q \mid \nu x\, P \mid \boxed{P} \qquad \text{Process}$$
$$M ::= \quad \pi_1.P_1 + \ldots + \pi_N.P_N \qquad \text{Choice}$$
$$E ::= \quad X_1(\tilde{m}_1) \mapsto P_1, \ldots, X_N(\tilde{m}_N) \mapsto P_N \qquad \text{Environmnent}$$

$\pi ::=$	τ_r	Delay
$\mid$	$\gamma!x(\tilde{n})$	Send
$\mid$	$\gamma?x(\tilde{m})$	Receive
$\mid$	$\mu!x$	Move
$\mid$	$\mu?x$	Accept
$\gamma ::=$	$\texttt{local}$	Local
$\mid$	$\texttt{s2s}$	Sibling
$\mid$	$\texttt{c2p}$	Parent
$\mid$	$\texttt{p2c}$	Child
$\mu ::=$	$\texttt{in}$	Enter
$\mid$	$\texttt{out}$	Leave
$\mid$	$\texttt{merge}$	Merge

$$\tau_r.P + M \xrightarrow{\ r\ } P$$

$$\texttt{local}!x(\tilde{n}).P + M \mid \texttt{local}?x(\tilde{m}).P' + M' \xrightarrow{\ F_x\ } P \mid P'_{\{\tilde{m}:=\tilde{n}\}}$$

$$\boxed{Q \mid \texttt{c2p}!x(\tilde{n}).P + M} \mid Q' \mid \texttt{c2p}?x(\tilde{m}).P' + M' \xrightarrow{\ F_x\ } \boxed{Q \mid P} \mid Q' \mid P'_{\{\tilde{m}:=\tilde{n}\}}$$

$$Q \mid \texttt{p2c}!x(\tilde{n}).P + M \mid \boxed{Q' \mid \texttt{p2c}?x(\tilde{m}).P' + M'} \xrightarrow{\ F_x\ } Q \mid P \mid \boxed{Q' \mid P'_{\{\tilde{m}:=\tilde{n}\}}}$$

$$\boxed{Q \mid \texttt{s2s}!x(\tilde{n}).P + M} \mid \boxed{Q' \mid \texttt{s2s}?x(\tilde{m}).P' + M'} \xrightarrow{\ F_x\ } \boxed{Q \mid P} \mid \boxed{Q' \mid P'_{\{\tilde{m}:=\tilde{n}\}}}$$

$$\boxed{Q \mid \texttt{in}!x.P + M} \mid \boxed{Q' \mid \texttt{in}?x.P' + M'} \xrightarrow{\ F_x\ } \boxed{\boxed{Q \mid P} \mid Q' \mid P'}$$

$$\boxed{\boxed{Q \mid \texttt{out}!x.P + M} \mid Q' \mid \texttt{out}?x.P' + M'} \xrightarrow{\ F_x\ } \boxed{Q \mid P} \mid \boxed{Q' \mid P'}$$

$$\boxed{Q \mid \texttt{merge}!x.P + M} \mid \boxed{Q' \mid \texttt{merge}?x.P' + M'} \xrightarrow{\ F_x\ } \boxed{Q \mid P \mid Q' \mid P'}$$

$$P \xrightarrow{\ r\ } P' \quad \Rightarrow \quad \boxed{P} \xrightarrow{\ r\ } \boxed{P'}$$

DEFINITION 14. Syntax and core reduction rules of the stochastic bioambient calculus, based on [12].

$\Gamma ::=$	a, b	Location
$I ::=$	$X(\tilde{n})^\Gamma$	Species
$MV ::=$	$\varnothing \mid \{\Gamma := \Gamma'\}$	Renaming

$$is\_local(\Gamma_1, \Gamma_2) \triangleq \Gamma_1 = \Gamma_2$$
$$is\_sibling((a_1, b_1), (a_2, b_2)) \triangleq a_1 \neq a_2 \land b_1 = b_2$$
$$is\_child((a_1, b_1), (a_2, b_2)) \triangleq b_1 = a_2$$

$$species(\mathbf{0}) \triangleq \emptyset$$
$$species(P) \triangleq species(P, (root_1, root_2))$$
$$species(\boxed{P}, (a,b)) \triangleq species(P, (a', a)) \textbf{ if } \text{fresh}(a')$$
$$species(X(\tilde{n}), \Gamma) \triangleq X(\tilde{n})^\Gamma \textbf{ if } E(X(\tilde{n})) = C$$
$$species(X(\tilde{n}), \Gamma) \triangleq species(P, \Gamma) \textbf{ if } E(X(\tilde{n})) = P \neq C$$
$$species(\nu x\, P, \Gamma) \triangleq species(P_{\{x:=y\}}, \Gamma) \textbf{ if } \text{fresh}(y)$$
$$species(P_1 \mid P_2, \Gamma) \triangleq species(P_1, \Gamma) \uplus species(P_2, \Gamma)$$

DEFINITION 15. Assigning locations to species, where a, b represent globally unique ambient identifiers.

$$(S, R)\#\varnothing \triangleq (S, R)$$
$$(S, R)\#MV \triangleq (S, R\#MV)\tilde{\#}MV$$
$$R\#MV \triangleq \{(J\#MV, F, MV'\#MV, J'\#MV) \mid (J, F, MV', J') \in R \land \text{releq}(J, J\#MV)\}$$
$$(\{S, (I \mapsto (i, C))\}, R)\tilde{\#}MV \triangleq (S', (I', S') \oplus R') \textbf{ if } I' = I\#MV \textbf{ and } S' = S''\{I' \mapsto (i + i', C)\}$$
$$\textbf{and } (\textbf{ if } I' \in \text{dom}(S) \textbf{ then } S(I') = (i', C) \textbf{ else } i' = 0)$$
$$\textbf{and } (S'', R') = (S, R)\tilde{\#}MV$$

$$releq(\{P_1^{\Gamma_1}, P_2^{\Gamma_2}\}, \{P_1^{\Gamma'_1}, P_2^{\Gamma'_2}\}) \triangleq$$
$$is\_local(\Gamma_1, \Gamma_2) \land is\_local(\Gamma'_1, \Gamma'_2) \quad \lor$$
$$is\_sibling(\Gamma_1, \Gamma_2) \land is\_sibling(\Gamma'_1, \Gamma'_2) \quad \lor$$
$$is\_child(\Gamma_1, \Gamma_2) \land is\_child(\Gamma'_1, \Gamma'_2) \quad \lor$$
$$is\_child(\Gamma_2, \Gamma_1) \land is\_child(\Gamma'_2, \Gamma'_1)$$

$$\{\Gamma_1 := \Gamma_2\}\#MV \triangleq \{\Gamma_1\#MV := \Gamma_2\#MV\}$$
$$X(\tilde{n})^\Gamma \#MV \triangleq X(\tilde{n})^{\Gamma\#MV}$$
$$\Gamma\#\{\Gamma_{old} := \Gamma_{new}\} \triangleq \Gamma_{new} \textbf{ if } \Gamma = \Gamma_{old}$$
$$(c, d)\#\{a, b := a', b'\} \triangleq c, a' \textbf{ if } d = a \textbf{ and } b' = b$$
$$\Gamma\#\{\Gamma_{old} := \Gamma_{new}\} \triangleq \Gamma \textbf{ otherwise}.$$

DEFINITION 16. Instantiation of the generic abstract machine to the stochastic biolambient calculus.

consists of a competition between zero or more actions $\pi.P$, where π is the action that can be performed, after which process P is executed. An action π can be a delay τ_r, a send $\gamma!x(\tilde{n})$ of values $\tilde{n}$ on channel x, or a receive $\gamma?x(\tilde{m})$ of values $\tilde{m}$ on channel x, where γ denotes the type of communication. This can be inside the same ambient (`local`), from one sibling ambient to another (`s2s`), from a child ambient to its parent (`c2p`) or from a parent ambient to a child (`p2c`). In addition, an action π can be a move $\mu!x$ on channel x or an accept $\mu?x$ on channel x, where μ denotes the type of movement. This can be an ambient entering one of its siblings (`in`), a child ambient leaving its parent (`out`) or a merge of two sibling ambients (`merge`).

B.2 Extracting Reactions from Ambients

The main challenge when instantiating the generic abstract machine with the bioambient calculus is to extract a "flat" set of reactions from a bioambient process. To achieve this, processes are labelled with the location in which they evolve. The location is a pair consisting of the identifier of the ambient in which the process is located, together with the identifier of the parent ambient. For instance, in the process $\boxed{P_1 \| \boxed{P_2}}$, P_2 has location (a, b), where a is the identifier the ambient containing P_2 and b is the identifier of the parent ambient. Both a and b denote ambient identifiers that are assumed to be globally unique. The assigning of locations to species is presented in Definition 15. The identifiers $root_1$ and $root_2$ denote the top-level enclosing ambient and the top-level parent ambient, respectively.

When computing reactions, the process locations are used to check if processes are able to interact. The computation of reactions from a bioambient process is given in Definition 17 and relies on the predicates is_local, $is\_sibling$ and is_child. For example, executing an `s2s` interaction between two species $P_1^{a_1, b_1}$ and $P_2^{a_2, b_2}$ requires that their parent ambients are equal but that their enclosing ambients are different, i.e. $b_1 = b_2$ and $a_1 \neq a_2$. When an ambient moves (by using `in`, `out`, `merge` actions), we relabel the species to reflect the change in location. As an example, consider the following reduction:

$$\boxed{Q_1 \mid \text{in}!x.P_1} \| \boxed{Q_2 \mid \text{in}?x.P_2} \longrightarrow \boxed{\boxed{Q_1 \mid P_1} \mid Q_2 \mid P_2}$$

Rewriting this using process locations gives the reduction below:

$$Q_1^{a,c} \mid \text{in}!x.P_1^{a,c} \mid Q_2^{b,c} \mid \text{in}?x.P_2^{b,c} \xrightarrow{F_x} Q_1^{a,b} \mid P_1^{a,b} \mid Q_2^{b,c} \mid P_2^{b,c}$$

and we obtain the corresponding reaction:

$$\text{in}!x.P_1^{a,c} + \text{in}?x.P_2^{b,c} \longrightarrow P_1^{a,b} + P_2^{b,c}$$

As the ambient containing P_1 and Q_1 moves, the locations of P_1 and Q_1 have to be updated. This is specified by attaching to each reaction O the required location renaming (in the above example, the renaming is $\{(a, c) := (a, b)\}$). The renaming operation and the reduction relation obtained are presented in Definition 16. During the renaming of current reactions ($R\#MV$), we must check whether the change in location invalidates any existing reactions. This is achieved using the $releq$ predicate presented in Definition 16. If the change in location means that species are too far apart to interact, the corresponding reaction is removed. As dis-

$$actions(X(\tilde{n})^\Gamma) \triangleq C_{\{\tilde{m}:=\tilde{n}\}} \text{ if } C = E(X(\tilde{m}))$$
$$reactions(I, J) \triangleq unary(I) \uplus binary(I, J)$$
$$unary(I) \triangleq \{(\{I\}, F, f, J) \mid (F, MV, J) \in delays(I)$$
$$\wedge f(T) = T\#MV\}$$
$$delays(I) \triangleq \{(F_r, \varnothing, species(P, \Gamma))$$
$$\mid \tau_r.P \in actions(I)\} \text{ if } I = X(\tilde{n})^\Gamma$$
$$binary(I_1, J) \triangleq \{(\{I_1, I_2\}, F, f, J)$$
$$\mid C_2 = actions(I_2) \wedge C_1 = actions(I_1)$$
$$\wedge I_1 = X_1(\tilde{n})^{\Gamma_1} \wedge I_2 = X_2(\tilde{m})^{\Gamma_2}$$
$$\wedge I_2 \in \{I_1\} \cup J \wedge f(T) = T\#MV$$
$$\wedge (F, MV, J) \in interact(C_1^{\Gamma_1}, C_2^{\Gamma_2})\}$$
$$interact(C_1^{\Gamma_1}, C_2^{\Gamma_2}) \triangleq comm(C_1^{\Gamma_1}, C_2^{\Gamma_2}) \uplus comm(C_2^{\Gamma_2}, C_1^{\Gamma_1})$$
$$\uplus moves(C_1^{\Gamma_1}, C_2^{\Gamma_2}) \uplus moves(C_2^{\Gamma_2}, C_1^{\Gamma_1})$$
$$comm(C_1^{\Gamma_1}, C_2^{\Gamma_2}) \triangleq \{(F_x, \varnothing, J)$$
$$\mid J = species(P_1, \Gamma_1)$$
$$\uplus species(P_{2\{\tilde{m}:=\tilde{n}\}}, \Gamma_2))$$
$$\wedge \gamma!x(\tilde{n}).P_1 \in C_1 \wedge \gamma?x(\tilde{m}).P_2 \in C_2$$
$$\wedge \gamma \in comm\_actions(\Gamma_1, \Gamma_2)\}$$
$$moves(C_1^{\Gamma_1}, C_2^{\Gamma_2}) \triangleq \{(F_x, \{\Gamma_1 := \Gamma_1^{new}\}, J)$$
$$\mid J = species(P_1, \Gamma_1) \uplus species(P_2, \Gamma_2))$$
$$\wedge \mu!x(\tilde{n}).P_1 \in C_1 \wedge \mu?x(\tilde{m}).P_2 \in C_2$$
$$\wedge \Gamma_1^{new} = move\_path(\mu, \Gamma_1, \Gamma_2)$$
$$\wedge \mu \in move\_actions(\Gamma_1, \Gamma_2)\}$$

$$\text{local} \in comm\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_local(\Gamma_1, \Gamma_2)$$
$$\text{s2s} \in comm\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_sibling(\Gamma_1, \Gamma_2)$$
$$\text{c2p} \in comm\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_child(\Gamma_1, \Gamma_2)$$
$$\text{p2c} \in comm\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_child(\Gamma_2, \Gamma_1)$$
$$\text{in} \in move\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_sibling(\Gamma_1, \Gamma_2)$$
$$\text{out} \in move\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_child(\Gamma_1, \Gamma_2)$$
$$\text{merge} \in move\_actions(\Gamma_1, \Gamma_2) \Leftrightarrow is\_sibling(\Gamma_1, \Gamma_2)$$
$$move\_path(\text{in}, (a, c), (b, c)) \triangleq a, b$$
$$move\_path(\text{out}, (a, b), (b, c)) \triangleq a, c$$
$$move\_path(\text{merge}, \Gamma_1, \Gamma_2) \triangleq \Gamma_2$$

DEFINITION 17. Instantiation of the generic machine to the Bioambient calculus. *We assume a fixed global environment E that contains the species definitions. Note that we set f to be the species renaming function $f(T) = T\#MV$.*

cussed in the main text, there is broad scope for calculus-specific optimisations. For example, unused reactions can be garbage-collected, ambients with the same contents can be grouped together as a single species, and complexes in the bioambient calculus can be treated in a similar fashion to complexes in the stochastic pi-calculus. The renaming function can also be optimised by using a hash table to directly access reactions concerned by the renaming. Finally, the proof of correctness of the abstract machine is left for future work.