Dynamic Compartments in the Imperative π-Calculus

Mathias John[1], Cédric Lhoussaine[2,4], and Joachim Niehren[3,4]

[1] University of Rostock, Computer Science, Modeling and Simulation Group
[2] University of Lille 1
[3] INRIA, Lille, Mostrare
[4] BioComputing, LIFL (CNRS UMR8022) and IRI (CNRS USR3078)

Abstract. Dynamic compartments with mutable configurations and variable volumes are of basic interest for the stochastic modeling of biochemistry in cells. We propose a new language to express dynamic compartments that we call the *imperative π-calculus*. It is obtained from the *attributed π-calculus* by adding imperative assignment operations to a global store. Previous approaches to dynamic compartments are improved in flexibility or efficiency. This is illustrated by an appropriate model of osmosis and a correct encoding of BioAmbients.

1   Introduction

Concurrent control is crucial for the stochastic modeling of biochemical processes in living cells [19,2,13]. The regulation of such systems depends on all kinds of physical or chemical aspects, such as volume, surface, temperature, pressure, pH value, spatial coordinates and structures. Most of these aspects are of global nature, so they require modeling languages in which global concurrent control can be expressed [20]. In this paper, we present a new modeling language, that permits to express many aspects with global control in a uniform manner, and illustrate its usefulness by modeling dynamic compartments with mutable volumes and surfaces.

Dynamic compartments may change their nesting structure dynamically, by applying operations for compartment creation, removal and merging. These operations may influence the speed of diverse reactions within compartments, in particular when compartment volumes change (global to local interactions). Vice versa, local reactions within a single compartment may effect global numeric attributes such as volume and surface (local to global interaction). Various languages for modeling systems with dynamic compartments were proposed for systems biology [18,14,21], but none of them can express physical, chemical, and compartmental aspects in a uniform manner, while providing efficient stochastic simulation. Spatial languages such as the Brane Calculi [2] or BioAmbients [18] fix a particular set of operators on compartments, and provide a special purpose solution for these operations. The π-calculus with polyadic synchronization and global priorities π@ is more flexible, in that it permits to

encode all kinds of compartment structures, including those of Brane Calculi and BioAmbients [20]. Unfortunately, such priority-based encodings are complex, low level, and inefficient. Consider e.g. the dissolving of a compartment with n equal molecules. Informing all of them requires $O(n)$ interactions rather than $O(1)$ by updating all at once. Furthermore, $\pi@$ lacks general support for stochastic rates and numeric attributes such as volumes and pH-values. The only solution to compartments with variable volumes so far [21] was expressed in the special purpose dialect called $S\pi@$. Numerical attributes of compartments are equally lacking in *Bigraphs* [14,12], a modeling language for spatial dynamics based on a particular form of hypergraph rewriting. Thus, the question is whether there exists a better general purpose language for expressing dynamic compartments.

In this paper, we start from the *attributed π-calculus* [11], and enrich it by an imperative store for global control. The attributed π-calculus is parametrized by a sequential higher-order language $\mathcal{L}$ for describing all kinds of values (symbolic and numeric) and constraints. It features "attributed" processes $A(e_1, \ldots, e_n)$ with values defined by expressions $e_1, \ldots, e_n$ of $\mathcal{L}$. For instance, cells with variable volumes *vol* can be modeled by using a single attribute:

$$Cell(vol) \triangleq enter[\lambda r. \text{ if } r{<}0.1 \text{ then } (\texttt{val } enter)]?(v).Cell(vol + v)$$

The input prefix contains a function in square brackets, that tests for every matching output prefix, whether the reaction is permitted and returns its stochastic rate in this case. Cells as above can be entered by elements $Ele(r, v)$ of radius r and volume v, if r is smaller than 0.1:

$$Ele(r, v) \triangleq enter[r]!(v).0$$

Under this condition, the stochastic rate of the *enter* reaction is obtained by evaluating the expression $(\texttt{val } enter)$, i.e., by accessing the value of channel *enter* from the environment. As a result of the reaction, the cell volume is increased by v. The entered elements disappear, since we chose to not represent elements in cells explicitly here.

We obtain the *imperative π-calculus* $\pi^{imp}(\mathcal{L})$, by allowing imperative programming languages $\mathcal{L}$ as attribute language. Thereby, we enrich the π-calculus by a global imperative store. More precisely, we add assignment expressions to $\mathcal{L}$ by which to change the values of channels dynamically, such as for instance $enter := \texttt{val } enter + 1.5$, whose evaluation increases the value of channel *enter* by 1.5. The expressions of $\mathcal{L}$ are evaluated as transactions, so that the evaluator cannot be interrupted by any other process. We present a stochastic semantics for $\pi^{imp}(\mathcal{L})$ that properly accounts for transactions with imperative assignments. We show how to compile processes of $\pi^{imp}(\mathcal{L})$ to stochastic simulators, independently of the choice of parameter $\mathcal{L}$. We have implemented the compiler and can report on first experimental results. To this purpose, we model a simple example of osmosis in $\pi^{imp}(\mathcal{L})$ where variable volumes and surfaces matter. Practical simulation experiments confirm higher accuracy compared to [21] due to variable surfaces (not only volumes) and good efficiency.

In order to provide a more systematic treatment of dynamic compartments, we present a compositional encoding of BioAmbients in $\pi^{imp}(\mathcal{L})$ and prove its correctness. The constraints of $\pi^{imp}(\mathcal{L})$ permit us to express the application conditions of BioAmbients operators on compartment level. This way, we obtain a stochastic simulator for BioAmbients, without special purpose implementation as in [15]. We finally discuss how to extend our encoding to a stochastic version of BioAmbients that accounts for variable volumes.

Omitted details and proofs can be found in the extended version.

Related work. Existing stochastic semantics of BioAmbients as in [1,15] consider only local stochastic aspects ignoring variable volumes or surfaces. The rates of compartment operations simply are assigned to the interaction channel, rather than depending on the compartements volume as one might expect.

Bigraphs [14] are able to express compartment merging as in BioAmbients [18] but no variable volumes. Kappa [6] is a graph rewrite language (without hypergraphs), which seems to be too limited for expressing compartment merging. Modeling languages with model checking facilities, such as BIOCHAM [3] and BioPEPA [4] are less expressive by design. BioPEPA allows for the representation of variable compartment volumes but not dynamic structures, see [5]. BlenX (or Beta binders) [7] supports compartments with some global dynamics but no variable volumes or surfaces. Stochastic simulators are available for all these languages.

2   Imperative π-Calculus

We introduce the imperative π-calculus $\pi^{imp}(\mathcal{L})$ by extending the attributed π-calculus with imperative assignments. As vocabulary, we fix an infinite set *Chans* whose elements x, y, z are called channels. They will name communication channels in the π-calculus (and thus chemical reactions) and serve as variables in $\mathcal{L}$.

Values and Expressions. An attribute language over *Chans* is a triple $\mathcal{L} = (Consts, Succ, \Downarrow)$. It defines a call-by-value lambda calculus, whose values $v \in$ *Vals* and expressions $e \in$ *Exprs* are given in Fig. 1. Besides the usual concept of variables $x \in$ *Chans*, abstractions $\lambda x.e$, and applications, there are expressions $e_1 := e_2$ for imperative assignments. Additionally, we assume function constants $\mathtt{val}, \mathtt{ref}^\star \in$ *Consts* in order to access values of variables in the environment. Furthermore, we include pairs $\langle e_1, e_2 \rangle$ with selectors $\mathtt{fst}, \mathtt{snd}$ and conditionals if e then e_1 else e_2 with Boolean constants $\mathtt{true}, \mathtt{false} \in$ *Consts*. Equality tests on constants are provided by a constant $\mathtt{=}$ of type *Consts* $\times$ *Consts* $\to \mathbb{B}$. There may be many further constants in *Consts* such as for arithmetics. As usual, we write $fn(e)$ and $bn(e)$ for the sets of free and bound variables in e. We use infix syntax without extra notice, for instance, writing $e_1\mathtt{=}e_2$ instead of $\mathtt{=}\langle e_1, e_n \rangle$. The shortcuts in Fig. 2 provide let expressions, sequential composition, conditionals without else, and simple pattern matching functions.

Channels in *Chans*	$x ::= \ldots$
Constants in *Consts*	$c ::= \texttt{val} \mid \texttt{ref}^\star \mid \texttt{fst} \mid \texttt{snd} \mid \texttt{true} \mid \texttt{false} \mid = \mid \texttt{unit} \mid \ldots$
Values	$v ::= x \mid c \mid \langle v_1, v_2 \rangle \mid \lambda x.e$
Expressions	$e ::= x \mid c \mid \langle e_1, e_2 \rangle \mid \lambda x.e \mid ee' \mid e_1 := e_2 \mid \text{if } e \text{ then } e_1 \text{ else } e_2$

Fig. 1. Values and expressions of the imperative call-by-value lambda calculus

$\text{LET } x = e_1 \text{ IN } e_2 =_{df} (\lambda x.e_2)e_1$ $\qquad$ $\text{IF } e \text{ THEN } e_1 =_{df} \text{ if } e \text{ then } e_1 \text{ else } \texttt{false}$

$e_1; e_2 =_{df} \text{LET } \_ = e_1 \text{ IN } e_2$ $\qquad$ $\text{IF NOT } e \text{ THEN } e_1 =_{df} \text{ if } e \text{ then } \texttt{false} \text{ else } e_1$

$$\lambda \langle c, x \rangle.e =_{df} \lambda p. \text{ IF } (\texttt{fst } p) = c \text{ THEN } (\lambda x.e)(\texttt{snd } p)$$

Fig. 2. Shortcuts for expressions

$$\frac{(e_1, \rho) \Downarrow (v_1, \rho_1) \quad (e_2, \rho_1) \Downarrow (v_2, \rho_2) \quad (v_1 v_2, \rho_2) \Downarrow (v, \rho')}{(e_1 e_2, \rho) \Downarrow (v, \rho')} \qquad \frac{c \in Consts}{(c, \rho) \Downarrow (c, \rho)}$$

$$\frac{(e_1, \rho) \Downarrow (x, \rho_1) \quad (e_2, \rho_1) \Downarrow (v, \rho') \quad x \in Chans}{(e_1 := e_2, \rho) \Downarrow (v, \rho'[x \mapsto v])} \qquad \frac{\rho(x) = v}{(\texttt{val } x, \rho) \Downarrow (v, \rho)}$$

$$\frac{\rho(x) \notin Chans}{(\texttt{ref}^\star x, \rho) \Downarrow (x, \rho)} \qquad \frac{(\texttt{ref}^\star \rho(x), \rho) \Downarrow (y, \rho)}{(\texttt{ref}^\star x, \rho) \Downarrow (y, \rho)}$$

$$\frac{(e_1, \rho) \Downarrow (v_1, \rho_1) \quad (e_2, \rho_1) \Downarrow (v_2, \rho')}{(\langle e_1, e_2 \rangle, \rho) \Downarrow \langle v_1, v_2 \rangle, \rho')} \qquad \frac{true}{(\texttt{fst } \langle v_1, v_2 \rangle, \rho) \Downarrow (v_1, \rho)}$$

$$(\texttt{snd } \langle v_1, v_2 \rangle, \rho) \Downarrow (v_2, \rho)$$

$$\frac{(e, \rho) \Downarrow (\texttt{true}, \rho_1) \quad (e_1, \rho_1) \Downarrow (v_1, \rho')}{(\text{if } e \text{ then } e_1 \text{ else } e_2, \rho) \Downarrow (v_1, \rho')} \qquad \frac{(e, \rho) \Downarrow (\texttt{false}, \rho_2) \quad (e_2, \rho_2) \Downarrow (v_2, \rho')}{(\text{if } e \text{ then } e_1 \text{ else } e_2, \rho) \Downarrow (v_2, \rho')}$$

$$\frac{(e_1, \rho) \Downarrow (v, \rho_1) \quad (e_2, \rho_1) \Downarrow (v, \rho') \quad v \in Chans \cup Consts}{(e_1 = e_2, \rho) \Downarrow (\texttt{true}, \rho')}$$

Fig. 3. Big-step evaluator for call-by-value lambda calculus

An environment for an expression $e \in Exprs$ is a total function $\rho : fn(e) \rightarrow Vals$ that maps free variables of e to values. We write $dom(\rho) = fn(e)$ for the domain of ρ and let *Env* be the set of all environments for arbitrary expressions. We write $\rho[x_1 \mapsto v_1, \ldots, x_n \rightarrow v_n]$ for the environment that maps distinct variable x_i to v_i for all $1 \leq i \leq n$ and all other variables y in the domain of ρ to $\rho(y)$. Environments such as $[x \mapsto \langle x, y \rangle, y \mapsto \langle x, x \rangle]$ can store any type of data structure, including graphs and hypergraphs. In a stochastic setting, they are useful to assign rates to reactions.

The third component of $\mathcal{L}$, the big-step evaluator $\Downarrow$, is a binary relation of type $(Exprs \times Env) \times (Vals \times Env)$. It fixes the semantics of all expressions. A relationship $(e, \rho) \Downarrow (v, \rho')$ states that expression e in environment ρ evaluates to value v with new environment ρ'. The big-step evaluator must satisfy the rules in Fig. 3. Assignments $x := v$ change the value of x in the current environment to v. Function `val` returns the value of a channel in the current environment. Function `ref`* serves for dereferentiation, i.e. it returns the last channel of acyclic reference

chains. In the environment $[x_1 \mapsto x_2, \ldots, x_{n-1} \mapsto x_n, x_n \mapsto v]$, $(\texttt{ref}^\star\, x_i)$ e.g. evaluates to x_n for all $1 \leq i \leq n$ if $v \notin$ *Chans*, while evaluation does not terminate if $v = x_n$.

The second component of $\mathcal{L}$ is a subset $Succ \subseteq$ *Vals*. We call the elements of $Succ$ successful values. Their role in $\pi^{imp}(\mathcal{L})$ is to describe the rate constants of communication actions. Considering a stochastic semantics, $Succ$ equals $\mathbb{R}^+$. Otherwise, it typically contains $\texttt{true}$ but not $\texttt{false}$.

Processes. The syntax of $\pi^{imp}(\mathcal{L})$, as given in Fig. 4, is equal to that of the attributed π-calculus [11], except that we now permit imperative assignments in $\mathcal{L}$. It extends on the usual syntax of the stochastic π-calculus [17,16,13], by permitting expressions to describe channel values, adding conditions to receivers and senders, and generalizing stochastic rate constants of channels to arbitrary values.

We assume a set of process names ranged over by A, each with a fixed arity $ar(A) \geq 0$. Furthermore, we freely use sequence notion, writing $\tilde{e}$ for a sequence of expressions, $\tilde{x}$ and $\tilde{y}$ for a sequence of channels, and $\tilde{v}$ for a sequence of values. Their lengths are denoted by $|\tilde{e}|$, $|\tilde{v}|$, and $|\tilde{x}|$, respectively.

A program consists of an initial process P_0 and a set of process definitions $\{D_1, \ldots, D_n\}$, exactly one per process name in P_0. A definition D of A has the form $A(\tilde{x}) \triangleq P$, where P is a process and $|\tilde{x}| = ar(A)$. Process P is a parallel composition of sums, channel creators, and defined processes. A channel creator $(\nu x{:}v)\ P$ asks for the creation of a new channel x with scope P that is mapped to v by the global environment. A sender $v[e]!\tilde{v}$, which conveys a sequence of values $\tilde{v}$ on channel v, is constrained by expression e. A receiver $v[e]?\tilde{y}$ of a sequence of values for parameters $\tilde{y}$ on channel v is conditioned by expression e. A call of a defined process $A(\tilde{e})$ consists of a process name A and a sequence $\tilde{e} \in$ *Exprs* where $|\tilde{e}| = ar(A)$. A sum Σ offers a choice $\pi_1.P_1 + \ldots + \pi_n.P_n$ between senders or receivers $\pi_i.P_i$, i.e., where π_i it either a sender or receiver prefix.

Processes	$P, Q ::= A(\tilde{e})$	defined process
	$\mid\ P_1 \mid P_2$	parallel composition
	$\mid\ (\nu x{:}v)\ P$	channel creation
	$\mid\ \Sigma$	sums
	$\mid\ \mathbf{0}$	empty solution
Sums	$\Sigma ::= \pi.P$	prefixed process
	$\mid\ \Sigma + \Sigma'$	summation
Prefixes	$\pi ::= v[e]?\tilde{y}$	receiver
	$\mid\ v[e]!\tilde{v}$	sender
Definitions	$D ::= A(\tilde{x}) \triangleq P$	parametric process definition

Fig. 4. Syntax of $\pi^{imp}(\mathcal{L})$: $e, \tilde{e}$ are expressions and $v, \tilde{v}$ values of $\mathcal{L}$, and $x, \tilde{x} \in$ *Chans*

Nondeterministic Operational Semantics. We start with a nondeterministic operational semantics for $\pi^{imp}(\mathcal{L})$ with an arbitrary attribute language $\mathcal{L}$. The sets of free and bound names of processes $fn(P)$ and $bn(P)$ are defined as usual, except that free and bound names in expressions are to be considered too.

The usual structural congruence on π-calculus processes $P \equiv P'$ is the least congruence containing alpha conversion $P =_\alpha P'$, where summation $+$ and parallel composition $|$ are associative and commutative, the latter with neutral element **0**, and satisfy the usual scoping rules of ν-binders:

$$(\nu x{:}v)\ (P_1 \mid P_2) \equiv (\nu x{:}v)\ P_1 \mid P_2 \quad \text{if } x \notin \mathit{fn}(P_2)$$
$$(\nu x{:}v)\ (\nu y{:}v')\ P \equiv (\nu y{:}v')\ (\nu x{:}v)\ P \ \text{ if } x \notin \mathit{fn}(v') \text{ and } y \notin \mathit{fn}(v)$$

An environment for a process P is a function $\rho : \mathit{fn}(P) \to \mathit{Vals}$.

The nondeterministic operational semantics in Fig. 5 defines judgements $(P_1, \rho_1) \to (P_2, \rho_2)$ meaning that a process P_1 in environment ρ_1 reduces in one step to process P_2 while changing the environment to ρ_2. The structural congruence may silently be applied at any point (CONTEXT). A step may either be a communication or an application of a defined process. A communication step (COM) applies to a sender and a receiver on the same channel x. Let e_1 and e_2 be the conditions of sender and receiver, respectively, and ρ the current environment. The communication step is enabled if $(e_1 e_2, \rho)$ reduces to (v, ρ') for some successful value $v \in \mathit{Succ}$. In this case, the resulting process continues in environment ρ', which may have been altered by assignment operations in $e_1 e_2$. In practice, a big step evaluator for $e_1 e_2$ may first have to change the environment and then run into an irreducible expression (a program error) or an unsuccessful value (where the communication constraint fails). In these cases, all changes done to the environment are to be backtracked. Furthermore, it may happen that the big step evaluator does not terminate (another kind of program error). An application step (REC) of a defined process $A(\tilde{e})$ evaluates all expression in $\tilde{e}$ from the left to the right while threading the environment changes, and if successful, applies the definition of A to the resulting values $\tilde{v}$. Parallel compositions (PAR) may be evaluated in arbitrary order even though the changes of the environment may depend on it. Rule (RES) for channel creation $(\nu x{:}v)\ P$ in environment ρ first adds $[x \mapsto v]$ to the environment, then reduces $(P, \rho[x \mapsto v])$ to some $(P', \rho'[x \mapsto v'])$, and continues with $((\nu x{:}v')\ P', \rho')$ where the new value v' of x is put back into a ν binder.

$$(\text{COM})\ \frac{(e_1 e_2, \rho) \Downarrow (v, \rho') \qquad v \in \mathit{Succ}}{(x[e_1]?\tilde{y}.P + \Sigma_1 \mid x[e_2]!\tilde{v}.Q + \Sigma_2, \rho) \to (P[\tilde{v}/\tilde{y}] \mid Q, \rho')}$$

$$(\text{REC})\ \frac{(\tilde{e}, \rho) \Downarrow (\tilde{v}, \rho') \qquad A(\tilde{x}) \triangleq P}{(A(\tilde{e}), \rho) \to (P[\tilde{v}/\tilde{x}], \rho')} \qquad (\text{PAR})\ \frac{(P, \rho) \to (P', \rho')}{(P \mid Q, \rho) \to (P' \mid Q, \rho')}$$

$$(\text{RES})\ \frac{(P, \rho[x \mapsto v]) \to (P', \rho'[x \mapsto v']) \qquad x \notin \mathit{dom}(\rho) \cup \mathit{dom}(\rho')}{((\nu x{:}v)\ P, \rho) \to ((\nu x{:}v')\ P', \rho')}$$

$$(\text{CONTEXT})\ \frac{P \equiv P' \qquad (P', \rho) \to (Q', \rho') \qquad Q' \equiv Q}{(P, \rho) \to (Q, \rho')}$$

Fig. 5. Nondeterministic operational semantics

Stochastic Operational Semantics. In the stochastic operational semantics all redexes must be computed before reducing one of them. The computation of redexes requires to evaluate $\mathcal{L}$ expressions, which may fail with program errors or nontermination. If the computation of a single redex fails, the whole process is considered erroneous. In any case, all state changes during redex computation need to be backtracked before verifying the next redex candidate. Only the finally selected redex is permitted to definitely commit its changes to the environment.

The stochastic semantics in Fig. 6 applies to programs in *biochemical form* and preserves these forms by reduction. A solution S is a process in biochemical form $N\Pi_{i=1}^{m}A_i(\tilde{e}_i)$, where N is a quantifier prefix $(\nu x_1{:}v_1)\ \ldots(\nu x_n{:}v_n)$ and $\Pi_{i=1}^{m}A_i(\tilde{e}_i) = A_1(\tilde{e}_1)\mid\ldots\mid A_n(\tilde{e}_m)$ a parallel composition of so called *molecules*. Molecules $A_i(\tilde{e}_i)$ must have definitions in biochemical form $A_i(\tilde{x}_i)\triangleq N_i\Sigma_i$ where Σ_i is a sum of prefixed processes in biochemical form. As usual, all process can be brought into *biochemical form* by flattening out nested sums into intermediate definitions.

The stochastic semantics of a program in biochemical normal form is a Markov chain, whose states are pairs $([S]_\equiv, \rho)$, where $[S]_\equiv$ is a class of a solution S wrt. structural congruence $\equiv$, and ρ is an environment for S. In order to compute a

---

Redexes $(1 \le j \le m, i_1, j_1, i_2, j_2 \in \mathbb{N})$

(CHOOSE) $\dfrac{(\tilde{e},\rho)\Downarrow(\tilde{v},\rho')\qquad A(\tilde{x})\triangleq N(\pi_1.S_1+\ldots+\pi_m.S_m)}{\text{choose}_j(A(\tilde{e}),\rho)=(N(\pi_j.S_j)[\tilde{v}/\tilde{x}],\rho')}$

(REDEX) $\dfrac{\begin{array}{cc}\text{choose}_{j_1}(A_{i_1}(\tilde{e}_{i_1}),\rho)=_\alpha(S_1',\rho_1) & S_1'=(\nu\widetilde{y_1{:}v_1})\ (x[e_1']?\tilde{y}.S_1)\\[4pt]\text{choose}_{j_2}(A_{i_2}(\tilde{e}_{i_2}),\rho_1)=_\alpha(S_2',\rho') & S_2'=(\nu\widetilde{y_2{:}v_2})\ (x[e_2']!\tilde{v}.S_2)\end{array}\qquad i_1\neq i_2}{(S_1',S_2',\rho')\in\text{redex}_{(i_1,j_1,i_2,j_2)}(\prod_{i=1}^{n}A_i(\tilde{e}_i),\rho)}$

where $x\in\textit{Chans}$, $x\notin\{\tilde{y_1}\}\cup\{\tilde{y_2}\}$ and $\{\tilde{y_1}\}\cap\{\tilde{y_2}\}=\emptyset$.

Labeled reduction $(r\in\mathbb{R}^+$ and $\ell=(i_1,j_1,i_2,j_2)\in\mathbb{N}^4)$

(COM) $\dfrac{\begin{array}{c}(N_1(x[e_1']?\tilde{y}.S_1),N_2(x[e_2']!\tilde{v}.S_2),\rho_1)\in\text{redex}_\ell(\prod_{i=1}^{n}A_i(\tilde{e}_i),\rho)\\[4pt](e_1'e_2',\rho_1)\Downarrow(r,\rho')\qquad r\in\textit{Succ}\end{array}}{(\prod_{i=1}^{n}A_i(\tilde{e}_i),\rho)\xrightarrow[\ell]{r}(\prod_{i=1,i\neq i_1,i_2}^{n}A_i(\tilde{e}_i)\mid N_1N_2(S_1[\tilde{v}/\tilde{y}]\mid S_2),\rho')}$

(NEW) $\dfrac{(S,\rho[v/x])\xrightarrow[\ell]{r}(S',\rho'[v'/x])\qquad x\notin dom(\rho)\cup dom(\rho')}{((\nu x{:}v)\ S,\rho)\xrightarrow[\ell]{r}((\nu x{:}v')\ S',\rho')}$

Markov chain $(r,r'\in\mathbb{R}^+)$

(CONV) $\dfrac{\forall\ell\in\mathbb{N}^4\forall(N_1(x[e_1']?\tilde{y}.S_1),N_2(x[e_2']!\tilde{v}.S_2))\in\text{redex}_\ell(S,\rho)}{\exists v\in\textit{Vals}\exists\rho':(e_1'e_2',\rho)\Downarrow(v,\rho')}{\dfrac{}{(S,\rho)\Downarrow}}$

(SUM) $\dfrac{(S,\rho)\Downarrow\qquad S\equiv S_1\qquad r=\sum_{\{\ell\mid(S_1,\rho)\xrightarrow[\ell]{r'}(S_2,\rho')\text{ and }S_2\equiv S'\}}r'\qquad r\neq 0}{(S,\rho)\xrightarrow{r}(S',\rho')}$

Fig. 6. Stochastic operational semantics

transition for such pairs, we need to compute all potential reductions of (S, ρ) and sum up their stochastic rates (SUM). The computation of all redexes must converge (CONV) before applying any reduction step. A label $\ell = (i_1, j_1, i_2, j_2) \in \mathbb{N}^4$ fixes the j_1'th alternative of molecules $A_{i_1}(\tilde{e}_{i_1})$ of S and the j_2'th alternative of molecule $A_{i_2}(\tilde{e}_{i_2})$. Label ℓ distinguishes a *redex candidate* if these molecules have distinct indexes $i_1 \neq i_2$, and if the selected alternatives consist of a sender and a receiver on the same channel (REDEX). Label ℓ defines a *redex*, if the sequences of expressions $\tilde{e}_1$ and $\tilde{e}_2$ can be evaluated successfully from left to right (CHOOSE), while starting with environment ρ, threading changes, and ending in some environment ρ'. In this case, we can apply the definitions of A_{i_1} and A_{i_2} to the resulting values, and instantiate the alternatives with index j_1 and j_2 to S_1' and S_2' . Note that the triple $(S_1', S_2', \rho') \in \text{redex}_\ell(S, \rho)$ is unique up to alpha renaming. Rule (COM) performs the actual communication step for a redex with label ℓ under the condition that the constraint of the redex is successful. Rule (NEW) is as for the nondeterministic case.

Consider e.g. a solution $A(x{:}{=}1) \mid B(x{:}{=}2)$. The evaluation order for the two assignments may vary with the redex candidate. For candidates where $A(x{:}{=}1)$ provides the receiver and $B(x{:}{=}2)$ the sender, we have to evaluate $x{:}{=}1$ before $x{:}{=}2$, so that we have to test the communication constraint with store $[x \mapsto 2]$. In the symmetric case, we will have to evaluate in the opposite order and to test the constraint with store $[x \mapsto 1]$.

Stochastic Simulation. A stochastic simulator for $\pi^{imp}(\mathcal{L})$ can be derived from the stochastic operational semantics independently of $\mathcal{L}$. The main difference to the attributed π-calculus [11] is the treatment of imperative expressions, which can either occur in constraints or in applications. Assignments in constraints increase computational complexity, since they force us to not only compare values but also environments for grouping senders and receivers. Furthermore, senders and receivers can not be evaluated separately anymore, but only in combination. However, computational complexity can be reduced by storing differences between environments before and after evaluation, i.e. the set of executed assignments, since then only the latter need to be compared. Assignments in applications make the extraction of multisets from solutions less effective and therefore negatively affect simulation efficiency.

3   A Model of Osmosis: Variable Volumes and Surfaces

Osmosis is a simple example for concurrent systems with compartments of variable volumes. It was modeled already in [21] based on a special purpose dialect $S\pi@$ of $\pi@$ with variable volumes. Here we show how to simulate osmosis in the imperative π-calculus with an attribute language that provides arithmetics. Our solution is more flexible and accurate, in that it accounts for dynamic changes of compartment surfaces, which cannot be expressed in $S\pi@$.

We consider a very simple system which consists of a sphere filled with water (H_2O), sodium (Na^+), and chlorine (Cl^-). The system contains a membrane

through which water may diffuse. This membrane separates an inner compartment `Inn` of spherical shape, from an outer compartment `Out`, which has the form of a sphere shell (a ring in 2D). The center point equals for both compartments. The precise values of these parameters are listed in the extended version.

For simplicity, we adopt the assumption of [21], that the volume of a compartment is determined by summing up the volumes of the contained molecules. However, in general, $\mathcal{L}$ allows for the definition of complex functions to obtain compartment volumes that e.g. consider atomic forces between particles. The volumes of `Inn` and `Out` change with water moving through the membrane. The radius of `Inn` may thus vary with diffusion, while the outer radius R of `Out` always remains fixed. Fig. 7 shows our model of the system in $\pi^{imp}(\mathcal{L}(\mathbb{R}, \mathtt{V}, \mathtt{C}))$. Its attribute language provides real number arithmetics with function constants for division `/`, multiplication `*`, and subtraction `-`, and numeric constants such

---

Parameters
 N: $\{H_2O, Na^+, Cl^-\} \times \{Inn, Out\} \rightarrow \mathbb{N}$ // copy numbers of molecules
Constants
 V: $\{H_2O, Na^+, Cl^-\} \rightarrow \mathbb{R}^+$ // molecule volumes
 C $\in \mathbb{R}$ // diffusion coefficient of water
Expressions
 RAD $=_{df}$ $\lambda v.((3*v)/(4*\pi))^{\frac{1}{3}}$ // volume to radius
 SURF $=_{df}$ $\lambda r. 4*\pi*r^2$ // radius to surface
 DIST $=_{df}$ $\lambda r_1 \lambda r_2. r_1 + ((r_2-r_1)/2)$ // diffusion distance
 R $=_{df}$ RAD $(\sum_{c \in \{Inn, Out\}} \sum_{m \in \{H_2O, Na^+, CL^-\}} V(m)*N(m,c))$ // outer radius of
                                                         // sphere shell
Public channels // initialize volumes of compartments
 inn: $\sum_{m \in \{H_2O, Na^+, CL^-\}}$ V(m)*N(m, Inn) // inner sphere
 out: $\sum_{m \in \{H_2O, Na^+, CL^-\}}$ V(m)*N(m, Out) // outer sphere shell
 diffuse: **unit**   // diffusion channel
Process definitions
 H_2O(ori, des) $\triangleq$
    diffuse$[\lambda_-.$ LET // diffusion from origin to destination
      r = RAD (**val** inn)   // radius of inner sphere
      a = (SURF r)/10   // diffusion area
      s = DIST r R   // diffusion distance
      diff = a*C/(s*(**val** ori)) // diffusion rate
    IN
      ori := **val** ori $-$ V(H_2O); // update volume of origin
      des := **val** des $+$ V(H_2O); // update volume of destination
      diff   // return diffusion rate
    ]?(). H_2O(des, ori)
 Membrane() $\triangleq$ diffuse[**unit**]!(). Membrane()
Solution
    $\prod_{i=1}^{N(H_2O, Inn)} H_2O$(inn, out) $\mid \prod_{i=1}^{N(H_2O, Out)} H_2O$(out, inn) $\mid$ Membrane()

Fig. 7. Modeling osmosis

as 2, 10, or π. Furthermore, there are three problem specific constants, the diffusion coefficient C of H_2O, the constant V for the function that maps molecules to their volumes, and the constant N for the function assigning copy numbers to molecules in compartments. The big-step evaluator for $\mathcal{L}(\mathbb{R}, \mathtt{V}, \mathtt{C})$ is defined as usual. Nonzero positive real numbers are successful, $Succ = \mathbb{R}^+$.

The diffusion rate of H_2O is determined by $\frac{a*C}{d*v}$, where a is the diffusion area, d the diffusion distance, and v the volume of the compartment that the molecule leaves, see [8]. We assume that 1/10 of Inn's surface serves as diffusion area. The radius and surface of Inn are computed from its volume by functions RAD and SURF, see Fig. 7. The diffusion distance represents the average way a molecule travels from one compartment to the other. Following the approach in [8], we assume the diffusion distance to be the distance between the two compartment centers. In the model, it is determined by function DIST applied to the constant outer radius of Out and the variable radius of Inn.

In our model, we represent the compartments Inn and Out as public channels inn and out, respectively, each referring to the variable volume of the corresponding compartment. The public channel diffuse with the dummy value unit represents diffusion reactions. Three processes are defined: H_2O (inn,out), which describes a water molecule in Inn that may diffuse to Out, H_2O (out,inn), its symmetric variant, and Membrane(), which enables diffusion on channel diffuse at all times.

The parametric processes H_2O (ori,des) may perform diffusion by communication on channel diffuse and then continue with H_2O (des,ori). The speed of this reaction is given by the diffusion rate, which varies with volumes and surfaces and is therefore consecutively recomputed. This is done by applying the function in the brackets diffuse[...]?. Every application of this function performs volume changes by assignments ori := val ori - V(H_2O) and des := val ori + V(H_2O). Since the simulator needs to compute the diffusion rates for all possible interactions in the system (there are at most two, water moving in or out), it has to reset the environment every time. Only once some interaction is chosen by the Stochastic Simulation Algorithm [9], it can commit to the changes required by this interaction.

By adapting the diffusion area and distance at each diffusion event, we extend the model presented in [21], where only volume changes are considered. In order
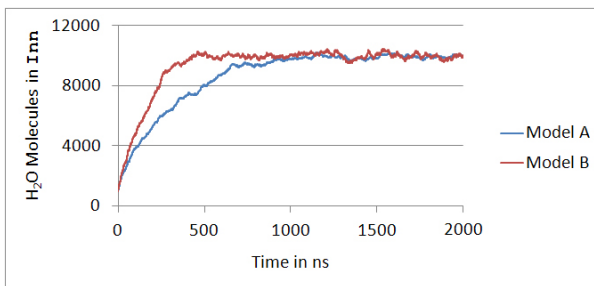

Fig. 8. Experiment results without (Model A) and with (Model B) variable surfaces

to compare both versions of the model, we implemented and simulated them in our tool, which is part of the modeling and simulation framework JamesII [10]. The results can be seen in Fig. 3. Model B, being the one that considers updates of the diffusion area and distance, features a steeper slope. This is due to the fact that with the increasing volume of `Inn`, the diffusion area grows faster than the distance, which raises the resulting diffusion rates.

4   Programming BioAmbients

We encode BioAmbients [18] in the imperative π-calculus, in order to show how to express concurrent systems with compartments and dynamic rearrangement systematically. In a first step, we ignore local stochastic aspects as in [1,15] which would not impose any particular problem, since these do not account for volume changes. See below for a discussion of extensions.

The syntax of BioAmbients is recalled in Fig. 9. It has the same syntactic categories as the π-calculus. Processes P can be enclosed by ambients $[P]$ whose nesting structure restricts interaction capacities similarly to compartments. There are prefixes for two kinds of interactions: communication and rearrangement. *Communication prefixes* "$d\ x?(\tilde{y})$" and "$d\ x!(\tilde{y})$" are prefixes of senders or receivers annotated by a communication direction d, which is either `local`, `s2s`, `c2p`, or `p2c`. They enable message sending either locally in an ambient, between sibling ambients, from a parent to a child, or vice versa. Similarly, there are *rearrangement prefixes*, prefixes of senders "$c\ x!$" and receivers "$c\ x?$" without arguments and annotated by rearrangement capacity c, either `merge`, `in`, or `out`. Rearrangement operations with these prefixes serve for ambient merging, entering into siblings, or exiting the current ambient. The reduction rules of the (nondeterministic) operational semantics of BioAmbients are given in Fig. 10. We refer the reader to [18] for the full operational semantics.

In order to encode BioAmbients, we identify every ambient using a channel r that gives reference to the characteristic values (CV) of the ambient. The CV $\langle n, r' \rangle$ consists of a unique name n naming the ambient and the reference r' of its parent (which is `unit` at the top-level). The ambient is encoded by a store binding r to the CV possibly via a reference chain: $[r \mapsto r_1, \ldots, r_{n-1} \mapsto r_n, r_n \mapsto \langle n, r' \rangle]$. The elements in ambient r will be encoded by defined processes $A(r)$.

Characteristic values can be changed by assignments $r := v$. When assignments are executed, the simulation algorithm automatically updates the communication

| Processes	$P, Q ::= [P] \mid A(\tilde{e}) \mid P\mid Q \mid (\nu x{:}v)\ P \mid \Sigma \mid \mathbf{0}$
Sums	$\Sigma, \Sigma' ::= \pi.P \mid \Sigma + \Sigma'$
Prefixes	$\pi ::= d\ x!\tilde{z} \mid d\ x?\tilde{z} \mid c\ x! \mid c\ x?$
Communication directions	$d ::= \texttt{local} \mid \texttt{s2s} \mid \texttt{c2p} \mid \texttt{p2c}$
Rearrangment capacities	$c ::= \texttt{merge} \mid \texttt{in} \mid \texttt{out}$
Definitions	$D ::= A(\tilde{x}) \triangleq P$

Fig. 9. Syntax of BioAmbients

$Communication: \texttt{local } x!\tilde{z}.P + \Sigma \mid \texttt{local } x?\tilde{y}.Q + \Sigma' \rightarrow P \mid Q\{\tilde{z}/\tilde{y}\}$

$[Q \mid \texttt{c2p } x!\tilde{z}.P + \Sigma] \mid \texttt{c2p } x?\tilde{y}.P' + \Sigma' \rightarrow [Q \mid P] \mid P'\{\tilde{z}/\tilde{y}\}$

$[Q \mid \texttt{p2c } x?\tilde{y}.P + \Sigma] \mid \texttt{p2c } x!\tilde{z}.P' + \Sigma' \rightarrow [Q \mid P\{\tilde{z}/\tilde{y}\}] \mid P'$

$[Q \mid \texttt{s2s } x!\tilde{z}.P + \Sigma] \mid [Q' \mid \texttt{s2s } x?\tilde{y}.P' + \Sigma'] \rightarrow [P \mid Q] \mid [Q' \mid P'\{\tilde{z}/\tilde{y}\}]$

$Rearrangement: [Q \mid \texttt{merge } x!P + \Sigma] \mid [Q' \mid \texttt{merge } x?P' + \Sigma'] \rightarrow [Q \mid P \mid Q' \mid P']$

$[Q \mid \texttt{in } x!P + \Sigma] \mid [Q' \mid \texttt{in } x?P' + \Sigma'] \rightarrow [[Q \mid P] \mid Q' \mid P']$

$[[Q \mid \texttt{out } x!P + \Sigma] \mid Q' \mid \texttt{out } x?P' + \Sigma'] \rightarrow [Q \mid P] \mid [Q' \mid P']$

Fig. 10. Reduction rules of BioAmbients

(a) enter/accept rearrangement leading to the update of entering CV.
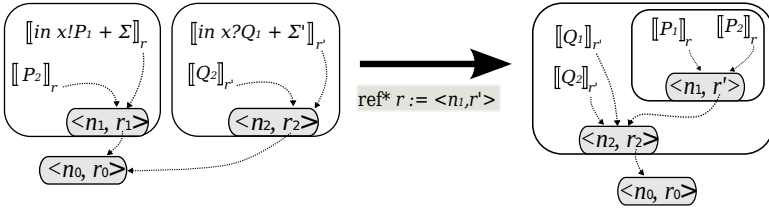
(b) exit/expel rearrangement leading to the update of exiting CV.
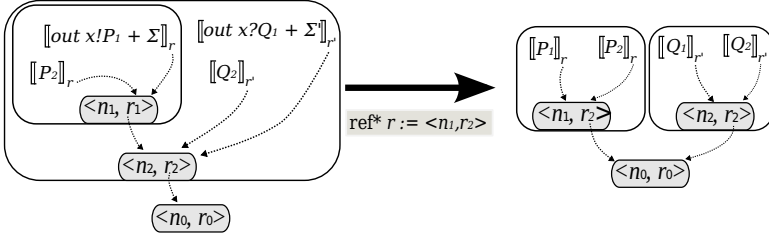
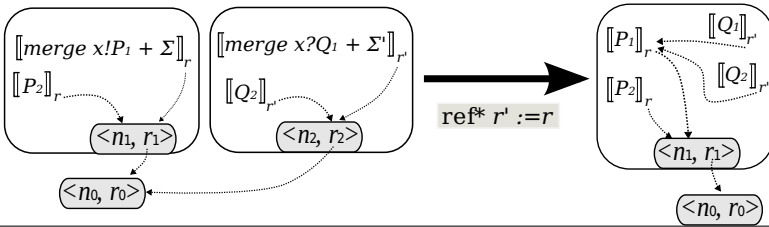(c) merging rearrangement leading to the update of merged CV.

Fig. 11. Simplified diagrams illustrating enter, exit and merge rearrangements

potential of all elements in the compartment. For instance, for a compartment that contains n copies of the same element $A(r)$, all updates can be done by a single inspection of the definition of $A(r)$, not n-times in contrast to priority-based encodings. Since dereferentiation might be required, this might still cost time $O(n)$ in the rare worst case, but will often be more efficient.

$$\text{PARENT} =_{df} \lambda r.\ \text{HERE } (\mathtt{snd}\ (\mathtt{val}\ (\mathtt{ref}^\star\ r))) \qquad \text{HERE} =_{df} \lambda r.\ \mathtt{fst}\ (\mathtt{val}\ (\mathtt{ref}^\star\ r))$$

$$[\![A(\tilde{x}) \triangleq P]\!] =_{df}\ A(\tilde{x}, r) \triangleq [\![P]\!]_r \qquad\qquad \text{where } r \notin \{\tilde{x}\} \cup fn(P)$$

$$[\![A(\tilde{x})]\!]_r =_{df}\ A(\tilde{x}, r) \qquad\qquad\qquad\qquad [\![P \mid Q]\!]_r =_{df}\ [\![P]\!]_r \mid [\![Q]\!]_r$$

$$[\![\Sigma + \Sigma']\!]_r =_{df}\ [\![\Sigma]\!]_r + [\![\Sigma']\!]_r \qquad\qquad\quad [\![(\nu x)P]\!]_r =_{df}\ (\nu x{:}\mathtt{unit})\ [\![P]\!]_r$$

$$[\![[P]]\!]_r =_{df}\ (\nu n : \mathtt{unit})(\nu r' : \langle n, r\rangle)[\![P]\!]_{r'} \qquad [\![\mathbf{0}]\!]_r =_{df}\ \mathbf{0}$$

$$[\![d\ x!(y).P]\!]_r =_{df}\ x[\langle d, r\rangle]!(y).[\![P]\!]_r \qquad \text{for } d \in \{\mathtt{local},\ \mathtt{s2s},\ \mathtt{p2c},\ \mathtt{c2p}\}$$

$$[\![\mathtt{local}\ x?(y).P]\!]_r =_{df}\ x[\lambda\langle\mathtt{local}, r'\rangle.(\text{HERE } r){=}(\text{HERE } r')]?(y).[\![P]\!]_r$$

$$[\![\mathtt{s2s}\ x?(y).P]\!]_r =_{df}\ x[\lambda\langle\mathtt{s2s}, r'\rangle.(\text{PARENT } r){=}(\text{PARENT } r')]?(y).[\![P]\!]_r$$

$$[\![\mathtt{p2c}\ x?(y).P]\!]_r =_{df}\ x[\lambda\langle\mathtt{c2p}, r'\rangle.(\text{HERE } r){=}(\text{PARENT } r')]?(y).[\![P]\!]_r$$

$$[\![\mathtt{c2p}\ x?(y).P]\!]_r =_{df}\ x[\lambda\langle\mathtt{p2c}, r'\rangle.(\text{PARENT } r){=}(\text{HERE } r')]?(y).[\![P]\!]_r$$

$$[\![c\ x!P]\!]_r =_{df}\ x[\langle c, r\rangle]!().[\![P]\!]_r \qquad\qquad \text{for } c \in \{\mathtt{merge},\ \mathtt{in},\ \mathtt{out}\}$$

$$[\![\mathtt{in}\ x?P]\!]_r =_{df}\ x\left[\begin{array}{l} \lambda\langle\mathtt{in}, r'\rangle.\text{IF } (\text{PARENT } r){=}(\text{PARENT } r') \\ \qquad \text{THEN IF NOT } (\text{HERE } r){=}(\text{HERE } r') \\ \qquad \text{THEN } (\mathtt{ref}^\star\ r' := \langle\text{HERE } r', r\rangle) \end{array}\right]?().[\![P]\!]_r$$

$$[\![\mathtt{out}\ x?P]\!]_r =_{df}\ x\left[\begin{array}{l} \lambda\langle\mathtt{out}, r'\rangle.\text{IF } (\text{PARENT } r'){=}(\text{HERE } r) \\ \qquad \text{THEN } (\mathtt{ref}^\star\ r' := \langle\text{HERE } r', \mathtt{snd}(\mathtt{val}(\mathtt{ref}^\star\ r)))\rangle) \end{array}\right]?().[\![P]\!]_r$$

$$[\![\mathtt{merge}\ x?P]\!]_r =_{df}\ x\left[\begin{array}{l} \lambda\langle\mathtt{merge}, r'\rangle.\text{IF } (\text{PARENT } r){=}(\text{PARENT } r') \\ \qquad\quad \text{THEN IF NOT } (\text{HERE } r){=}(\text{HERE } r') \\ \qquad\quad \text{THEN } (\mathtt{ref}^\star\ r' := r) \end{array}\right]?().[\![P]\!]_r$$

Fig. 12. Encoding BioAmbients

We encode BioAmbients into $\pi^{imp}(\mathcal{L}(cap, dir))$, which provides constants for all directions and capacities of BioAmbients. The encoding is given in Fig. 12. We first define two lambda expressions HERE and PARENT which map ambients r to their name HERE $r = n$ and to the name of their parent PARENT $r = $ HERE r'.

For every BioAmbients process P in ambient r, the encoding defines a unique process $[\![P]\!]_r$ in $\pi^{imp}(cap, dir)$. Encoding an ambient $[P]$ with parent r consists in creating a new ambient name n and a reference r' to the CV $\langle n, r\rangle$, and proceed with the encoding $[\![P]\!]_{r'}$. In general, this is how one can dynamically create new ambients. Encodings of rearrangement prefixes are illustrated by the diagrams in Fig. 11. Dashed arrows link references to their CV's. The graphical boxes represent ambients $[P]$ and are annotated by the CV of the ambient.

In Diagram (a), ambient r with CV $\langle n_1, r_1\rangle$ enters ambient r' with CV $\langle n_2, r_2\rangle$. The translation has to specify that the first ambient becomes a child of the second. Therefore, we update the CV of r to $\langle n_1, r'\rangle$, such that its parent is now r'. Note that the rearrangement is allowed only if the ambients are siblings. We thus have to perform the sibling test and the CV update in an atomic manner by a communication constraint on x in $[\![\mathtt{in}\ x?P]\!]_r$:

$$\lambda\langle\mathtt{in}, r'\rangle.\text{IF } (\text{PARENT } r){=}(\text{PARENT } r') \text{ THEN}$$
$$\qquad\quad \text{IF NOT } (\text{HERE } r){=}(\text{HERE } r') \text{ THEN } (\mathtt{ref}^\star\ r') := \langle\text{HERE } r', r\rangle$$

This function matches its argument against the pair $\langle \mathtt{in}, r' \rangle$, checks that the parent of receiver r coincides with the parent of its communication partner r', checks that both processes are not located within the same ambient and finally updates the CV of the sender accordingly. Note that an encoding of BioAmbients with stochastic aspects, as considered in [1,15], would simply make this function return the rate of x (that is $\mathtt{val}\ x$ assuming communication channels refer to their stochastic rate) in the sequence with the reference assignment. Diagram (b) describes the exiting ambients and Diagram (c) ambient merging. These used similar concepts as ambient entering in Diagram (a).

We define the top-level encoding $[\![P]\!]_{\nu r}$ by $(\nu r{:}\langle \mathtt{unit}, \mathtt{unit} \rangle)\ [\![P]\!]_r$ and call an environment ρ for P *ground* if $\rho(x) = \mathtt{unit}$ for all $x \in fn(P)$. We define $\simeq$ as the least congruence such that $\equiv\, \subseteq\, \simeq$ and $(\nu r'{:}v)\ (\nu r{:}r')\ P \simeq (\nu r'{:}v)\ P\{r'/r\}$. This equivalence is preserved by reduction of BioAmbients encodings, that is for any BioAmbients term P and $\pi^{imp}(\mathcal{L}(cap, dir))$ processes $Q_1 = [\![P]\!]_{\nu r}$ and Q_2, such that $Q_1 \simeq Q_2$, then $(Q_1, \rho) \rightarrow (Q'_1, \rho)$ iff $(Q_2, \rho) \rightarrow (Q'_2, \rho)$ with $Q'_1 \simeq Q'_2$.

Theorem 1 (Soundness and completeness of BioAmbients encoding)

1. *For all BioAmbients processes P, P', if $P \rightarrow P'$ then there exists a process $Q' \simeq [\![P']\!]_{\nu r}$ of $\pi^{imp}(\mathcal{L}(cap, dir))$ such that $([\![P]\!]_{\nu r}, \rho) \rightarrow (Q', \rho)$ for every ground environment ρ of $[\![P]\!]_{\nu r}$.*
2. *For all BioAmbients processes P, ground environment ρ of $[\![P]\!]_{\nu r}$, and $\pi^{imp}(\mathcal{L}(cap, dir))$ process Q', if $([\![P]\!]_{\nu r}, \rho) \rightarrow (Q', \rho)$ then there exists a BioAmbients process P' of such that $P \rightarrow P'$ and $[\![P']\!]_{\nu r} \simeq Q'$.*

BioAmbients with Variable Volumes. Stochastic rates of reactions in compartmented systems depend on concentrations of reactants and thus on volumes of compartments. This was already illustrated by the osmosis example in Section 3. In this section, we discuss notions of volumes for ambients, and how to model them in the imperative π-calculus. Which logics for volumes to choose depends on the concrete geometry that is assumed.

When considering spatial systems where compartment nesting corresponds to geometrical nesting, we have to distinguish two notions of volumes: the *molecular volume* of a compartment, which sums up the volumes of all molecules that it contains, and the *geometric volume*, which adds the geometric volumes of all child compartments to the molecular volume. In the osmosis example, the geometric volume of the outer sphere shell (of which is outer radius R depends) does indeed include the volumes of all molecules of the inner sphere.

In order to model BioAmbients with molecular and geometric volumes in the imperative π-calculus, we can enrich the CV's of compartments by these volumes, and define lambda expressions MVOL r and AVOL r to access them when knowing the ambient's reference r. Furthermore, we have to update these volumes for all operations of the calculus, which can be expressed by using assignment operations and real arithmetics. These details need elaboration beyond 15 pages.

5   Conclusion and Outlook

We have shown that imperative assignments for the π-calculus yield global effects, that offer an alternative to priorities. These permit to express operations of compartment dissolution and merging in an efficient, simpler and stochastic manner. The imperative π-calculus thus answers the question for a better modeling language for dynamic compartments. In work, we would like to further investigate on the relation to Bigraphs.

References

1. Brodo, L., Degano, P., Priami, C.: A stochastic semantics for bioambients. In: Malyshkin, V.E. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 22–34. Springer, Heidelberg (2007)
2. Cardelli, L.: Brane calculi. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–278. Springer, Heidelberg (2005)
3. Chabrier-Rivier, N., Fages, F., Soliman, S.: The biochemical abstract machine BIOCHAM. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 172–191. Springer, Heidelberg (2005)
4. Ciocchetta, F., Hillston, J.: Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks. ENTCS 194(3), 103–117 (2008)
5. Ciocchetta, F., Guerriero, M.L.: Modelling Biological Compartments in Bio-PEPA. ENTCS 227, 77–95 (2009)
6. Danos, V., Laneve, C.: Formal molecular biology. TCS 325(1), 69–110 (2004)
7. Dematté, L., Priami, C., Romanel, A.: Modelling and Simulation of Biological Processes in BlenX. SIGMETRICS Perf. Evaluation Review 35(4), 32–39 (2008)
8. Elf, J., Ehrenberg, M.: Spontaneous Separation of Bi-Stable Biochemical Systems into Spatial Domains of Opposite Phases. IEEE Proceedings Systems Biology 1(2), 230–236 (2004)
9. Gillespie, D.T.: Exact Stochastic Simulation of Coupled Chemical Reactions. Journal of Physical Chemistry 81, 2340–2361 (1977)
10. Himmelspach, J., Uhrmacher, A.M.: Plug'n Simulate. In: ANSS 2007, IEEE Proceedings, pp. 137–143 (2007)
11. John, M., Lhoussaine, C., Niehren, J., Uhrmacher, A.M.: The attributed pi calculus. In: Heiner, M., Uhrmacher, A.M. (eds.) CMSB 2008. LNCS (LNBI), vol. 5307, pp. 83–102. Springer, Heidelberg (2008)
12. Krivine, J., Milner, R., Troina, A.: Stochastic bigraphs. ENTCS 218, 73–96 (2008)
13. Kuttler, C., Lhoussaine, C., Niehren, J.: A Stochastic Pi-Calculus for Concurrent Objects. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) Ab 2007. LNCS, vol. 4545, pp. 232–246. Springer, Heidelberg (2007)
14. Milner, R.: Pure bigraphs: Structure and dynamics. Information and Computation 204(1), 60–122 (2006)
15. Phillips, A.: An Abstract Machine for the Stochastic Bioambient Calculus. ENTCS 227, 143–159 (2009)

16. Phillips, A., Cardelli, L.: Efficient, correct simulation of biological processes in the stochastic pi-calculus. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 184–199. Springer, Heidelberg (2007)
17. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a Stochastic Name-Passing Calculus to Representation and Simulation of Molecular Processes. Information Processing Letters 80, 25–31 (2001)
18. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: BioAmbients: An Abstraction for Biological Compartments. TCS 325(1), 141–167 (2004)
19. Regev, A., Shapiro, E.: Cells as Computation. Nature 419, 343 (2002)
20. Versari, C.: A Core Calculus for a Comparative Analysis of Bio-Inspired Calculi. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 411–425. Springer, Heidelberg (2007)
21. Versari, C., Busi, N.: Stochastic Biological Modelling in Presence of Multiple Compartments. TCS (to appear)