Types for BioAmbients*

Sara Capecchi Angelo Troina
Dipartimento di Informatica, Università di Torino
{capecchi,troina}@di.unito.it

The BioAmbients calculus is a process algebra suitable for representing compartmentalization, molecular localization and movements between compartments. In this paper we enrich this calculus with a static type system classifying each ambient with group types specifying the kind of compartments in which the ambient can stay. The type system ensures that, in a well-typed process, ambients cannot be nested in a way that violates the type hierarchy. Exploiting the information given by the group types, we also extend the operational semantics of BioAmbients with rules signalling errors that may derive from undesired ambients' moves (i.e. merging incompatible tissues). Thus, the signal of errors can help the modeller to detect and locate unwanted situations that may arise in a biological system, and give practical hints on how to avoid the undesired behaviour.

1 Introduction

BioAmbients [23] is a variant of the Ambient Calculus [11], in which compartments are described as a hierarchy of boundary ambients. This hierarchy can be modified by suitable operations that have an immediate biological interpretation; for example, the interactions between compounds that reside in the cytosol and in the nucleus of a cell could be modelled via parent–child communications. Thus, BioAmbients is quite suitable for the representation of various aspects of molecular localization and compartmentalization, such as the movement of molecules between compartments, the dynamic rearrangement that occurs between cellular compartments, and the interaction between the molecules in a compartmentalized context.

A stochastic semantics for BioAmbients is given in [8], and an abstract machine for this semantics is developed in [20]. In [17] BioAmbients is extended with an operator modelling chain-like biomolecular structures and applied within a DNA transcription example. In [21] a technique for pathway analysis is defined in terms of static control flow analysis. The authors then apply their technique to model and investigate an endocytic pathway that facilitates the process of receptor mediated endocytosis.

In this paper we extend the BioAmbients calculus with a static type system that classifies each ambient with a group type G specifying the kind of compartments in which the ambient can stay [10]. In other words, a group type G describes the properties of all the ambients and processes of that group. Group types are defined as pairs $(\mathcal{S},\mathcal{C})$, where $\mathcal{S}$ and $\mathcal{C}$ are sets of group types. Intuitively, given $G=(\mathcal{S},\mathcal{C})$, $\mathcal{S}$ denotes the set of ambient groups where ambients of type G can stay, while $\mathcal{C}$ is the set of ambient groups that can be crossed by ambients of type G. On the one hand, the set $\mathcal{S}$ can be used to list all the elements that are allowed within a compartment (complementary, all the elements which are not allowed, i.e. repelled). On the other hand, the set $\mathcal{C}$ lists all the elements that can cross an ambient, thus modelling permeability properties of a compartment.

Starting from group types as bases, we define a type system ensuring that, in a well-typed process, ambients cannot be nested in a way that violates the group hierarchy. Then, we extend the operational

*This research is funded by the BioBITs Project (*Converging Technologies* 2007, area: Biotechnology-ICT), Regione Piemonte.

FBTC 2010
EPTCS ??, 20??, pp. 1–13, doi:10.4204/EPTCS.??.??

semantics of BioAmbients, exploiting the information given by the group types, with rules rising warnings and signalling errors that may derive from undesired compartment interactions. For example, while correctness of the *enter*/*accept* capabilities (that are used to move a compartment to the inside of another compartment) can be checked statically, the *merge* capability (which merges two compartments into one) and the *exit*/*expel* capabilities (which are used to move a compartment from the inside to the outside of another compartment) could cause the movement of an ambient of type G within an ambient of type G' which does not accept it. In these cases, for example when incompatible tissues come in contact, an error signal is raised dynamically and the execution of the system is blocked. The modeller can exploit these signals as helpful *debugging* information in order to detect and locate the unwanted situations that may arise in a biological system. Intuitively, they give practical hints on how to avoid the undesired behaviour.

In the last few years there has been a growing interest on the use of type disciplines to enforce biological properties. In [3] a type system has been defined to ensure the wellformedness of links between protein sites within the Linked Calculus of Looping Sequences (see [4]). In [16] three type systems are defined for the Biochemical Abstract Machine, BIOCHAM (see [1]). The first one is used to infer the functions of proteins in a reaction model, the second one to infer activation and inhibition effects of proteins, and the last one to infer the topology of compartments. In [15] we have defined a type system for the Calculus of Looping Sequences (see [6]) to guarantee the soundness of reduction rules with respect to the requirement of certain elements, and the repellency of others. Finally, in [14] we have proposed a type system for the Stochastic Calculus of Looping sequences (see [5]) that allows for a quantitative analysis and models how the presence of catalysers (or inibitors) can modify the speed of reactions.

1.1 Summary

The remainder of the paper is organised as follows. In Section 2 we recall the original BioAmbients' syntax. In Section 3 we define our type system and in Section 4 we give our typed operational semantics. In Section 5 me formulate two motivating examples, namely we use our type system to analyse blood transfusions (rising errors in the case incompatible blood types get mixed) and spore protection against bacteriophage viruses. Finally, in Section 6 we draw our conclusions.

2 BioAmbients: Syntax

In this section we recall the BioAmbients calculus. *Ambients* represent bounded mobile entities that can be nested forming hierarchies. They provide an intuitive mean to model both *membrane-bound compartments*, where the components of a compartment are isolated from the external environment, and *molecular compartments* i.e. multi molecular complexes in which molecules can be partially isolated from the environment. *Capabilities* are used to model movements changing ambients hierarchies: they can be employed to model membranes fusion, molecules movement, complexes formation. Finally, *communications* model interactions between components within or across ambients boundaries.

The syntax is defined in Figure 1 and is the same as that of [23]. The only difference is that in our syntax ambients names are not optional. We give a name to each ambient in order to associate its type to it. *Ambient names* are ranged over by $a, a_1, b \ldots$, *channel names* are ranged over by $c, c_1, \ldots$, *capability names* are ranged over by $h, h_1, \ldots$. We use n, m to range over unspecified names and P,Q, R, T to range over processes.

Capabilities syncronise using names (h) and allow an ambient (1) to enter in a sibling ambient accepting it (enter h /accept h), (2) to leave the parent ambient (exit h /expel h), (3) to merge with a sibling forming a unique ambient (merge$\oplus$ h /merge$-$ h). *Communications* on channels ($\$c!\{m\}$,$\$c?\{m\}$) are prefixed by *directions* ($) denoting different kinds of communications: local communications (loc) within the same ambient, sibling communications (sts) between sibling ambients, parent/child (ptc,ctp) between nested ambients. Concerning processes syntax: inaction $\mathbf{0}$ is a special case of summation ($I = \emptyset$) and denotes the process doing nothing; restriction $(\nu\ n)P$ restricts the scope of the name n to P; $P \mid Q$ denotes the parallel composition of P and Q; $!P$ stands for process replication; $a[\![P]\!]$ describes a process P confined in an ambient named a; communication and capability choices ($\sum_{i\in I}\pi_i.P_i$, $\sum_{i\in I}M_i.P_i$) generalise communication and capability prefixes respectively ($\pi.P$, $S.P$) and represent standard choices.

3 The Type System

We classify ambients names with *group types* as in [10, 12]. Intuitively, the type G of an ambient denotes the set of ambients where that particular ambient can stay: it describes, in terms of other group types (possibly including G), the properties of all the ambients and processes of that group.

Group types consist of two components and are of the form $(\mathcal{S},\mathcal{C})$, where $\mathcal{S}$ and $\mathcal{C}$ are sets of group types. The intuitive meanings of the types' sets are the following:

- $\mathcal{S}$ is the set of ambient groups where the ambients of group G can stay;
- $\mathcal{C}$ is the set of ambient groups that G-ambients can cross, i.e., those that they may be driven into or out of, respectively, by enter and exit capabilities.

Clearly for all G $\mathcal{C}(G)\subseteq\mathcal{S}(G)$. If $G=(\mathcal{S},\mathcal{C})$ is a group type, we write $\mathcal{S}(G)$ and $\mathcal{C}(G)$ respectively to denote the components $\mathcal{S}$ and $\mathcal{C}$ of G. We call G_{Univ} the type of universal environments where each ambient can stay in. Types syntax is given in Figure 2.

Besides group types we have:

- Capability types: $(\overline{G}_1,\overline{G}_2)^{\ell}$ is the type associated to a name h through which ambients of types $\overline{G}_1$ and $\overline{G}_2$ can perform the movements described by ℓ.
- Channel types γ: the types of the channels arguments which can be groups (G) , capabilities (s) or channels (γ).

Notation 1. *Let M be a capability prefix and $s=(\overline{G}_1,\overline{G}_2)^{(M_1,M_2)}$ be a capability type, we say $M \in s$ if either $M = M_1$ or $M = M_2$.*

We now define well-formedness for capability types.

Definition 1 (s-Well-formedness)**.** *A capability type $(\overline{G}_1,\overline{G}_2)^{\ell}$ is well formed iff none of the following holds:*

1. $\ell=$ enter/accept *and* $\exists G_i \in \overline{G}_2,\ G_j \in \overline{G}_1 : G_i \notin \mathcal{C}(G_j)$
2. $\ell=$ exit/expel *and* $\exists G_i \in \overline{G}_2,\ G_j \in \overline{G}_1 : G_i \notin \mathcal{C}(G_j)$

Intuitively, a capability type $(\overline{G}_1,\overline{G}_2)^{\ell}$ describing the entrance(exit) of an ambient of type $G_j \in \overline{G}_1$ into(out of) an ambient of type $G_i \in G_2$ is not correct if G_j cannot (cross)stay in G_i.
We now define the environment Γ mapping names to types:

$$\Gamma ::= \emptyset \mid \Gamma, m : t$$

π	::=		Actions
	|	$\$c!\{m\}$	Output
	|	$\$c?\{m\}$	Input
$\$$	::=		Directions
	|	loc	Intra-Ambient
	|	sts	Inter-siblings
	|	ptc	Parent to child
	|	ctp	Child to parent
M	::=		Capabilities Prefixes
	|	enter	Entry
	|	accept	Accept
	|	exit	Exit
	|	expel	Expel
	|	merge$\oplus$	Merge with
	|	merge$-$	Merge into
S	::=	Mh	Capabilities
P	::=		Processes
	|	$\mathbf{0}$	Empty process
	|	$(\nu\ n)P$	Restriction
	|	$P \mid Q$	Composition
	|	$!P$	Replication
	|	$a[\![P]\!]$	Ambient
	|	$\pi.P$	Communication prefix
	|	$S.P$	Capability prefix
	|	$\sum_{i\in I} \pi_i.P_i$	Communication choice
	|	$\sum_{i\in I} M_i.P_i$	Capability choice

Figure 1: BioAmbients: Syntax.

$G_1 \ldots G_n$			Group types
t	::=	$G \mid s \mid \gamma$	Channels arguments
s	::=	$(\overline{G}_1, \overline{G}_2)^\ell$	Capability types
ℓ	::=	enter/accept | exit/expel | merge$\oplus$/merge$-$	Labels
γ	::=	$ch\{t\}$	Channels

Figure 2: Type syntax.

we assume that we can write $\Gamma, m{:}\ t$ only if m does not occur in Γ, i.e. $m \notin Dom(\Gamma)$ ($Dom(\Gamma)$ denotes the domain of Γ, i.e., the set of names occurring in Γ). An environment Γ is well formed if for each name the associated type is well formed.

In the following we define compatibility between a group type and an argument type.

Definition 2 (s-G Compatibility)**.** *Given a capability type $s{=}(\overline{G}_1, \overline{G}_2)^\ell$ and a group type G we define their compatibility as follows:*

$$s \asymp G \text{ iff } s \text{ is well formed and at least one between } G \in \overline{G}_1 \text{ and } G \in \overline{G}_2 \text{ holds.}$$

We can check the safety of BioAmbients processes using the rules in Figure 3. Let Γ be type environment from which we derive the type of names (rule $\lfloor$Name$\rfloor$); typing rules for processes have the shape :

$$\Gamma \vdash P : \overline{G} \triangleright \Delta$$

where P is a process, $\overline{G}$ is a set of group types representing the types of the ambients in P and Δ is a set of capability types collecting the capabilities in P.

$\lfloor$Inact$\rfloor$ derives any group type G for the empty process (indeed the empty process can stay in every type of ambient); $\lfloor$Par$\rfloor$ gives to parallel composition of processes P and Q the union of the sets of groups $\overline{G}_1$ and $\overline{G}_2$ obtained by typing P and Q; rule $\lfloor$Amb$\rfloor$ checks whether a process P can be safely nested in an ambient a of type G: if P is typed with a set of types $\overline{G}$ we have to ensure that every type G_k in $\overline{G}$ can stay in an ambient of type G; moreover, all capability types collected in Δ while typing P must be compatible with G; since the scope of the capabilities is the enclosing ambient, once the capabilities in Δ have been checked to be admissible, Δ is emptied; rule $\lfloor$Cap$\rfloor$ verifies the correspondence between the type of a name used for capability synchronization and the capability prefix used with it and then adds the type to Δ; rule $\lfloor$Choice$\rfloor$ gives to the choice between P and Q the union of the sets of groups $\overline{G}_1$ and $\overline{G}_2$ derived by typing P and Q.

We now show an example motivating the presence of the C set of ambient groups that can be crossed. Hydrophilic molecules are typically charge–polarized and capable of hydrogen bonding, thus enabling it to dissolve more quickly in water than in oil. Hydrophobic molecules instead tend to be non–polar and thus prefer other neutral molecules and non–polar solvents. As a consequence, hydrophobic molecules in water tend to cluster together forming micelles. Hydrophobic molecules can cross cell membranes in a natural (and slow) way, even if there is no particular transporter on the membrane. On the contrary, hydrophilic molecules can cross membranes only with dedicated transporters (conveyers). We can model these *crossing* properties with our type system. Namely, we can represent cells with or without conveyers as ambients of type G_{CConv} and G_C respectively; molecules can be of type G_{Hphi} (hydrophilic) and G_{Hpho} (hydrophobic). Finally, transporters have type G_{Conv}. Molecules of types G_{Hphi} and G_{Hpho} can stay in both G_{CConv} and G_C cells but only G_{Hpho} molecules can cross G_C cells. The sets $\mathcal{S}$ and C associated to the types are given in Figure 4.

$$\Gamma \vdash \mathbf{0} : G \triangleright \emptyset \ \lfloor \textsc{Inact} \rfloor \qquad \Gamma, n : t \vdash n : t \ \lfloor \textsc{Name} \rfloor$$

$$\frac{\Gamma \vdash P : \overline{G} \triangleright \Delta}{\Gamma \vdash (\nu\, c)P : \overline{G} \triangleright \Delta} \lfloor \textsc{Restr} \rfloor \qquad \frac{\Gamma \vdash P : \overline{G}_1 \triangleright \Delta_1 \quad \Gamma \vdash Q : \overline{G}_2 \triangleright \Delta_2}{\Gamma \vdash P \mid Q : \overline{G_1, G_2} \triangleright \Delta_1 \cup \Delta_2} \lfloor \textsc{Par} \rfloor \qquad \frac{\Gamma \vdash P : \overline{G} \triangleright \Delta}{\Gamma \vdash !P : \overline{G} \triangleright \Delta} \lfloor \textsc{Repl} \rfloor$$

$$\frac{\Gamma \vdash a : G \quad \Gamma \vdash P : \overline{G} \triangleright \Delta \quad G \in \mathcal{S}(G_k), \quad \forall G_k \in \overline{G} \quad s \asymp G \ \ \forall \ \ s \in \Delta}{\Gamma \vdash a[[P]] : G \triangleright \emptyset} \lfloor \textsc{Amb} \rfloor$$

$$\frac{\Gamma \vdash c : ch\{t\} \quad \Gamma, n : t \vdash P : \overline{G} \triangleright \Delta}{\Gamma \vdash \$c?\{n\}.P : \overline{G} \triangleright \Delta} \lfloor \textsc{Input} \rfloor \qquad \frac{\Gamma \vdash c : ch\{t\} \quad \Gamma \vdash P : \overline{G} \triangleright \Delta \quad \Gamma \vdash m : t}{\Gamma \vdash \$c!\{m\}.P : \overline{G} \triangleright \Delta} \lfloor \textsc{Out} \rfloor$$

$$\frac{\Gamma \vdash h : s \quad M \in s \quad \Gamma \vdash P : \overline{G} \triangleright \Delta}{\Gamma \vdash Mh.P : \overline{G} \triangleright \Delta \cup \{s\}} \lfloor \textsc{Cap} \rfloor \qquad \frac{\Gamma \vdash P : \overline{G}_1 \triangleright \Delta_1 \quad \Gamma \vdash Q : \overline{G}_2 \triangleright \Delta_2}{\Gamma \vdash P + Q : \overline{G_1, G_2} \triangleright \Delta_1 \cup \Delta_2} \lfloor \textsc{Choice} \rfloor$$

Figure 3: Typing rules.

Group types G	$\mathcal{S}(G)$	$C(G)$
G_C	G_{Univ}	G_{Univ}
G_{CConv}	G_{Univ}	G_{Univ}
G_{Conv}	G_{CConv}	G_{Univ}
G_{Hphi}	G_{CConv}, G_C	G_{CConv}
G_{Hpho}	G_{CConv}, G_C	G_{CConv}, G_C

Figure 4: Types for molecules and cells.

Let

$$\Gamma = cellC : G_{CConv},\ cell : G_C,\ conv : G_{Conv},\ h' : (G_{Hphi}, G_C)^{\text{exit/expel}},\ h'' : (G_{Hpho}, G_C)^{\text{enter/accept}},\ mol_1 : G_{Hphi},\ mol_2 : G_{Hpho},\ h_1 : (\{G_{Hphi}, G_{Hpho}\}, G_{Conv})^{\text{enter/accept}},\ h_2 : (G_{Conv}, G_{CConv})^{\text{exit/expel}},\ h_3 : (G_{Conv}, G_{CConv})^{\text{enter/accept}},\ h_4 : (\{G_{Hphi}, G_{Hpho}\}, G_{Conv})^{\text{exit/expel}}$$

A cell with conveyors can be modeled as an ambient of type G_{CConv} with nested conveyors and molecules:

$$cellC[[!conv[[P]] \mid mol_1[[\mathsf{enter}\ h.\mathsf{exit}\ h_4]] \mid mol_2[[\mathsf{exit}\ h' + \mathsf{enter}\ h.\mathsf{exit}\ h_4]] \mid \mathsf{expel}\ h']]$$

where $P = \mathsf{accept}\ h.\mathsf{enter}\ h_1.\mathsf{exit}\ h_2.\mathsf{enter}\ h_3.\mathsf{expel}\ h_4$. Thus we model the conveyor as first accepting molecules through h then exiting the current cell through h_2, entering a new cell through h_3 and finally releasing it through h_4. Molecules of type G_{Hpho} (mol_1) can enter inside the conveyor through h and finally be expelled by it, after the transport, through h_4; instead molecules of type G_{Hpho} (mol_2) can also pass the membrane cell without the use of a conveyor (through h').

4 Typed Operational semantics

In this section we extend the semantics of BioAmbients by adding rules which rise errors as a consequence of undesired behaviour. The structural congruence of BioAmbients remains unchanged, we recall

$P \mid Q \equiv Q \mid P$ $\quad$ $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$

$P \mid \mathbf{0} \equiv P$ $\quad$ $(\nu\, n)\mathbf{0} \equiv \mathbf{0}$

$(\nu\, n)(\nu\, m)P \equiv (\nu\, m)(\nu\, n)P$ $\quad$ $(\nu\, n)P \mid Q \equiv P \mid (\nu\, n)Q$ if $n \notin \mathrm{fn}(P)$

$(\nu\, n)a[[P]] \equiv a[[(\nu\, n)P]]$ $\quad$ $\$c?\{m\}.P \equiv \$c?\{n\}.P[m \leftarrow n]$ if $n \notin \mathrm{fn}(P)$

$(\nu\, m)P \equiv (\nu\, n)P[m \leftarrow n]$ if $n \notin \mathrm{fn}(P)$ $\quad$ $!\mathbf{0} \equiv \mathbf{0}$

$!P \equiv P \mid !P$

Figure 5: Structural congruence.

it in Figure 5. Rules for ambients movements and communications model *reactions* which may happen when two complementary prefixes on the same name n occur in parallel. Safety of communications and enter/accept capabilities can be statically checked by typing rules: enter/accept capabilities are ensured to be well formed, i.e. they cannot move an ambient $a[[\ldots]]$ of type G in an ambient $b[[\ldots]]$ of type G' if $G' \notin \mathcal{S}(G)$ (see Definiton 1). On the other hand, a static control of exit/expel and merge$\oplus$/merge$-$ capabilities would require too many constraints in the definition of group types: we should check the relation between the group types involved in all possible exit/expel and merge$\oplus$/merge$-$ interactions; as a consequence the type system would be very restrictive discarding also safe reductions just because of the presence of potentially unsafe capability prefixes in a choice. For this reason we check exit/expel and merge$\oplus$/merge$-$ reductions at run-time, signalling errors when they arise. The reduction rules are in Figure 6.

Rule $\lfloor$RedIn$\rfloor$ reduces the synchronization (thorough a name h) of enter h/accept h capability to the entrance of an ambient $a[[\ldots]]$ in an ambient $b[[\ldots]]$. As explained above, if this rule is applied to a well typed process after the reduction the nesting of ambients is safe. Rule $\lfloor$RedOut$\rfloor$ reduces the synchronization of exit h/expel h prefixes to the exit of an ambient $b[[\ldots]]$ out of an ambient $a[[\ldots]]$. We put a warning $W(G_b)$ in parallel with the new sibling ambients, since we do not know in which ambient $b[[\ldots]]$ will arrive once exited from $a[[\ldots]]$: e.g. $b[[\ldots]]$ could be nested in an ambient where it cannot stay. Two rules model the reduction of warned ambients inside another ambient: $\lfloor$Red Amb WarnOK$\rfloor$ reduces the warning parallel to a safe one if the exit did not produced an unsafe nesting; on the contrary, in case of unsafe nesting $\lfloor$Red Amb Warning$\rfloor$ generates an error; finally rule $\lfloor$Red Amb$\rfloor$ models reduction inside an ambient when there are no warnings.

Rule $\lfloor$Merge$\rfloor$ reduces the synchronization of merge$\oplus$ h/merge$-$ h prefixes to the fusion of two sibling ambients into a single one: the merge$-$ prefix "brings" all the processes in parallel with the prefixed one into the sibling ambient. We cannot check statically which processes will be in parallel with the prefixed one when the reduction rule is applied: we perform this check at runtime raising an error in case of unsafe nesting due to the merging of two sibling ambients (rule $\lfloor$Red Merge Warning$\rfloor$). Concerning communications, rules are unchanged w.r.t. [23], they model names substitutions due to communications between processes located in same ambient ($\lfloor$Red Local$\rfloor$), in parent-child ambients ($\lfloor$Red Parent Output$\rfloor$, $\lfloor$Red Parent Input$\rfloor$) and in sibling ambients $\lfloor$Red Sibling$\rfloor$.

Note that in the rules of our operational semantics there are no checks on the sets C since capability types should be well formed (thus satisfying the conditions for C sets).
Let Error $G_i[[G_j]]$ range over expelError $G_i[[G_j]]$ and mergeError $G_i[[G_j]]$. A well typed process either reduces to another well typed process or generates an error.

Theorem 1. *If* $\Gamma \vdash P : \overline{G} \triangleright \Delta$ *then*

- *either* $P \longrightarrow P'$ *or* $P \longrightarrow P' \mid W(G)$ *and* $\exists\, \Gamma', \overline{G}', \Delta'$ *such that* $\Gamma \vdash P' : \overline{G}' \triangleright \Delta'$
- *or* $\exists\, G_i, G_j$ *such that* $P \longrightarrow$ Error $G_i[[G_j]]$

$$a[\![(T + \mathsf{enter}\ h.P) \mid Q]\!] \mid b[\![(T' + \mathsf{accept}\ h.R) \mid S]\!] \longrightarrow b[\![a[\![P \mid Q]\!] \mid R \mid S]\!] \quad \lfloor \textsc{Red In} \rfloor$$

$$\frac{\Gamma \vdash b : G_b}{a[\![b[\![(T + \mathsf{exit}\ h.P) \mid Q]\!] \mid (T' + \mathsf{expel}\ h.R) \mid S]\!] \longrightarrow b[\![P \mid Q]\!] \mid a[\![R \mid S]\!] \mid W(G_b)} \quad \lfloor \textsc{Red Out} \rfloor$$

$$\frac{P \longrightarrow R \mid W(G_i) \quad \Gamma \vdash a : G_a \quad G_a \in \mathcal{S}(G_i)}{a[\![P]\!] \longrightarrow a[\![R]\!]} \quad \lfloor \textsc{Red Amb WarnOK} \rfloor$$

$$\frac{P \longrightarrow R \mid W(G_i) \quad \Gamma \vdash a : G_a \quad G_a \notin \mathcal{S}(G_i)}{a[\![P]\!] \longrightarrow \mathsf{expelError}\ \ G_a[\![G_i]\!]} \quad \lfloor \textsc{Red Amb Warning} \rfloor$$

$$\frac{P \longrightarrow Q}{a[\![P]\!] \longrightarrow a[\![Q]\!]} \quad \lfloor \textsc{Red Amb} \rfloor$$

$$\frac{\Gamma \vdash a : G_a \quad \Gamma \vdash R : \overline{G}_R \quad \Gamma \vdash S : \overline{G}_S \quad \forall G_i \in (\overline{G}_R, \overline{G}_S),\ G_a \in \mathcal{S}(G_i)}{a[\![(T + \mathsf{merge}\oplus\ h.P) \mid Q]\!] \mid b[\![(T' + \mathsf{merge}-\ h.R) \mid S]\!] \longrightarrow a[\![P \mid Q \mid R \mid S]\!]} \quad \lfloor \textsc{Red Merge} \rfloor$$

$$\frac{\Gamma \vdash a : G_a \quad \Gamma \vdash R : \overline{G}_R \quad \Gamma \vdash S : \overline{G}_S \quad \exists G_i \in (\overline{G}_R, \overline{G}_S),\ G_a \notin \mathcal{S}(G_i)}{a[\![(T + \mathsf{merge}\oplus\ h.P) \mid Q]\!] \mid b[\![(T' + \mathsf{merge}-\ h.R) \mid S]\!] \longrightarrow \mathsf{mergeError}\ \ G_a[\![G_i]\!]} \quad \lfloor \textsc{Red Merge Error} \rfloor$$

$$(T + \mathsf{loc}\ c?\{m\}.P) \mid (T' + \mathsf{loc}\ c!\{n\}.Q) \longrightarrow P[m \leftarrow n] \mid Q \quad \lfloor \textsc{Red Local} \rfloor$$

$$(T + \mathsf{ptc}\ c!\{n\}.P) \mid a[\![(T' + \mathsf{ctp}\ c?\{m\}.Q)]\!] \mid R \longrightarrow P \mid a[\![Q[m \leftarrow n] \mid R]\!] \quad \lfloor \textsc{Red Parent Output} \rfloor$$

$$a[\![(T + \mathsf{ctp}\ c!\{n\}.P) \mid R]\!] \mid (T' + \mathsf{ptc}\ c?\{n\}.Q) \longrightarrow a[\![R \mid P]\!] \mid Q[m \leftarrow n] \quad \lfloor \textsc{Red Parent Input} \rfloor$$

$$a[\![(T + \mathsf{sts}\ c!\{n\}.P) \mid R]\!] \mid b[\![(T' + \mathsf{sts}\ c?\{m\}.Q) \mid S]\!] \longrightarrow a[\![R \mid P]\!] \mid b[\![Q[m \leftarrow n] \mid S]\!] \quad \lfloor \textsc{Red Sibling} \rfloor$$

$$\frac{P \longrightarrow Q}{(\nu\ n)P \longrightarrow (\nu\ n)Q} \lfloor \textsc{Red Res} \rfloor \qquad \frac{P \longrightarrow Q}{P \mid R \longrightarrow Q \mid R} \lfloor \textsc{Red Par} \rfloor \qquad \frac{P \equiv P', P \longrightarrow Q, Q \equiv Q'}{P' \longrightarrow Q'} \lfloor \textsc{Red} \equiv \rfloor$$

Figure 6: Operational Semantics

Proof. *By induction on the definition of* $\rightarrow$.

Note that our semantics does not reduce the warnings $W(G)$ at top level. While they do not affect the system evolution, they could be useful in a compositional setting. In particular, if the entire process should be nested, at some point, into another ambient, the warnings keep the conditions on the admissible ambients (without the need to recompute the whole type of the system).

5 Motivating Examples

In this section we provide a couple of simple but motivating examples. In the following we will use a instead of $a[\![\mathbf{0}]\!]$ and we assume $\mathcal{S}(G)$=$\mathcal{C}(G)$ whenever C is not explicitly represented.

Group types of basic elements	$\mathcal{S}(G)$
$\mathcal{S}(G_a)$	G_{A-}, G_{AB-}, G_{A+}, G_{AB+}
$\mathcal{S}(G_b)$	G_{B-}, G_{AB-}, G_{B+}, G_{AB+}
$\mathcal{S}(G_r)$	G_{A+},G_{B+}, G_{AB+}, G_{O+}
$\mathcal{S}(G_{\overline{a}})$	G_{B+}, G_{B-}, G_{O+} ,G_{O-}
$\mathcal{S}(G_{\overline{b}})$	G_{A+}, G_{A-}, G_{O+}, G_{O-}
$\mathcal{S}(G_{\overline{r}})$	G_{A-},G_{B-}, G_{AB-}, G_{O-}

Figure 7: Types for blood groups.

5.1 Blood transfusion

This example has been inspired by [7]. A blood type is a classification of blood based on the presence or absence of inherited antigenic substances on the surface of red blood cells: these antigens are the A antigen and the B antigen. Blood type A contains only A antigens, blood type B contains only B antigens, blood type AB contains both and the blood type O contains none of them.

The immune system will produce antibodies that can specifically bind to a blood group antigen that is not recognized as self: individuals of blood type A have Anti-B antibodies, individuals of blood type B have Anti-A antibodies, individuals of blood type O have both Anti-A and Anti-B antibodies, and individuals of blood type AB have none of them. These antibodies can bind to the antigens on the surface of the transfused red blood cells, often leading to the destruction of the cell: for this reason, it is vital that compatible blood is selected for transfusions.

Another antigen that refines the classification of blood types is the RhD antigen: if this antigen is present, the blood type is called positive, else it is called negative. Unlike the ABO blood classification, the RhD antigen is immunogenic, meaning that a person who is RhD negative is very likely to produce Anti-RhD antibodies when exposed to the RhD antigen, but it is also common for RhD-negative individuals not to have Anti-RhD antibodies. We model blood transfusion as a system consisting of a set of closed tissues. Tissues contain blood cells and antibodies according to the classification described above, then they can join each other performing a transfusion of different blood types. We model a red blood cell as an ambient whose type represents the blood type; thus, the groups representing blood types are: G_{A+}, G_{A-}, G_{B+}, G_{B-}, G_{AB+}, G_{AB-}, G_{O+}, G_{O-}. We represent A,B, RhD antigen and Anti-A, Anti-B and Anti-RhD antibodies as ambients of type G_a, G_b, G_r, $G_{\overline{a}}$, $G_{\overline{b}}$, $G_{\overline{r}}$ respectively. The sets $\mathcal{S}(G)$ associated to the different blood types are given in Figure 7. Finally, we model a tissue (which contains the red cells) as an ambient of type $G_i \in \{G_{A+}, G_{A-}, G_{B+}, G_{B-}, G_{AB+}, G_{AB-}, G_{O+}, G_{O-}\}$.

We model blood transfusion as the reduction between two tissues having complementary merge capabilities ($\mathsf{merge}-$ for the donor, $\mathsf{merge}\oplus$ for the receiver). For instance let us consider a tissue t_1 represented by an ambient of type G_{A+} and two potential donors t_2 and t_3 of types G_{B+} and G_{O+} respectively:

$$P = t_1[\![\,!(\mathsf{merge}\oplus h_1 + \mathsf{merge}\oplus h_2 + \ldots \mathsf{merge}\oplus h_n) \mid a_1 \mid \overline{b_1} \mid r_1]\!] \mid t_2[\![\mathsf{merge}- h_1 \mid b_1 \mid r_2]\!] \mid t_3[\![\mathsf{merge}- h_2 \mid r_3]\!]$$

P is well typed with

$$\Gamma = t_1 : G_{A+},\ t_2 : G_{B+},\ t_3 : G_{O+},\ a_1 : G_a,\ \overline{b_1} : G_{\overline{b}},\ r_1 : G_r,\ r_2 : G_r,\ r_3 : G_r,\ b_1 : G_b,$$
$$h_1 : (G_{A+}, G_{B+})^{\mathsf{merge}\oplus/\mathsf{merge}-}, h_2 : (G_{A+}, G_{O+})^{\mathsf{merge}\oplus/\mathsf{merge}-}, \ldots$$

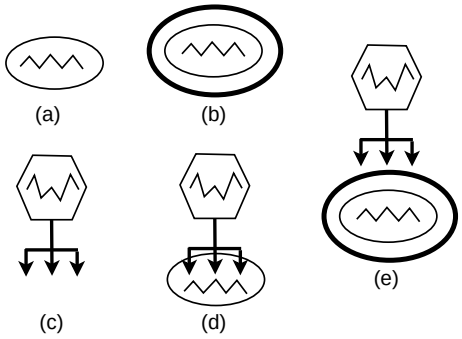

Figure 8: A bacterium (a) could be represented as a membrane containing the bacterium's DNA. A resistent (coated) spore (b) is represented as a bacterium surrounded by its coat. A bacteriophage (c) is depicted with the outer capsid, containing the genetic material, and the hypodermic syringe, used to inject its genetic material into the bacteria cells (d). They cannot inject coated cells (e).

Thus, the tissue t_1 can potentially receive blood from many donors ($\mathsf{merge}\oplus h_1 + \mathsf{merge}\oplus h_2 + \ldots \mathsf{merge}\oplus h_n$), and, because for example of some human error, may also receive blood which is not compatible to its own. Let us consider two possible reductions. The first one:

$$t_1[\![!(\mathsf{merge}\oplus h_1 + \mathsf{merge}\oplus h_2 + \ldots \mathsf{merge}\oplus h_n) \mid a_1 \mid \overline{b_1} \mid r_1]\!] \mid t_2[\![\mathsf{merge}- h_1 \mid b_1 \mid r_2]\!] \longrightarrow \mathsf{mergeError}\ G_{A+}[\![G_b]\!]$$

results in an error because of a wrong transfusion causing the presence of an antigen of type G_b in a tissue of type G_{A+}. The second one:

$$t_1[\![!(\mathsf{merge}\oplus h_1 + \mathsf{merge}\oplus h_2 + \ldots \mathsf{merge}\oplus h_n) \mid a_1 \mid \overline{b_1} \mid r_1]\!] \mid t_3[\![\mathsf{merge}- h_2 \mid r_3]\!] \rightarrow$$
$$t_1[\![!(\mathsf{merge}\oplus h_1 + \mathsf{merge}\oplus h_2 + \ldots \mathsf{merge}\oplus h_n) \mid a_1 \mid \overline{b_1} \mid r_1 \mid r_3]\!]$$

models a transfusion between compatible blood types, namely $A+$ and $O+$.

5.2 Bacteriophage viruses

In this section we use our system to model the interaction between bacteria and bacteriophage viruses (see Figure 8).

We assume that a bacterium consists of a cellular membrane containing its DNA. The sporulation mechanism allows producing inactive and very resistant bacteria forms, called spores which are surrounded by a membrane (coat) protecting them from virus attacks. A spore can germinate and then produce a new bacterium. A bacterium can safely stay in ambients containing viruses if it is protected by its coat. The types involved in this model are: G_{EnvOk}, $G_{EnvVirus}$ are the types of environments respectively virus-free and virus-friendly; G_{Bact},G_{Coat} are the types of the bacteria and the protecting membrane. G_{Vir} is the type of viruses. The corresponding (relevant) $\mathcal{S}$ groups are shown in Figure 9.
Now let us consider a bacteria b_2 surrounded by a coat b_1 (we omit the description of the DNA inside the bacterium):

$$P = b_1[\![b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!] \mid !(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_2)]\!]$$

P is well typed with:

$$\Gamma = h : (G_{Bact},G_{Coat})^{\mathsf{exit/expel}},\ h_1 : (G_{Coat},G_{EnvOK})^{\mathsf{enter/accept}},\ h_2 : (G_{Coat},G_{EnVirus})^{\mathsf{enter/accept}},$$
$$b_1 : G_{Coat},\ b_2 : G_{Bact},\ a_1 : G_{EnvVirus} a_2 : G_{EnvOk}$$

Group types of basic elements	$\mathcal{S}(G)$
G_{Bact}	G_{EnvOk}
G_{Coat}	G_{EnvOk}, $G_{EnvVirus}$
G_{Vir}	$G_{EnvVirus}$

Figure 9: Bacteria-Viruses Example: Types

We represent the chance of the bacterium to get rid of the coat as an exit/expel capability through the name h which allows the bacterium to germinate (exiting from its protecting membrane). The coat (containing the bacterium) can move in every environment, while the bacterium can only enter G_{EnvOk} environments; this is modeled by the use of suitable enter/accept capabilities. We now put the two environments a_1 (allowing viruses) and a_2 (virus-free) in parallel with the bacterium b_1. There are three possible behaviour (in the following we put labels on transitions for the sake of readability):

1. The bacterium gets rid of the coat and then can enter only in a_2:
$$b_1[\![b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!] \mid !(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_1)]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!] \mid a_2[\![!\mathsf{accept}\ h_2]\!]$$
$$\xrightarrow{\mathsf{exit/expel}(h)}$$
$$b_1[\![!(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_1)]\!] \mid b_2[\![\mathsf{enter}\ h_2]\!] \mid W(a_1[\![!\mathsf{accept}\ h_1]\!]) \mid a_2[\![!\mathsf{accept}\ h_2]\!]$$
$$\xrightarrow{\mathsf{enter/accept}(h_2)}$$
$$b_1[\![!(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_1)]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!] \mid a_2[\![b_2[\![]\!] \mid !\mathsf{accept}\ h_2]\!]$$
2. The coat can move in the ambient a_1 and then expel the bacterium in this hostile environment (thus generating an error):
$$b_1[\![b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!] \mid !(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_1)]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!] \mid a_2[\![!\mathsf{accept}\ h_2]\!]$$
$$\xrightarrow{\mathsf{enter/accept}(h_1)}$$
$$a_1[\![b_1[\![b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!] \mid !(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_2)]\!] \mid !\mathsf{accept}\ h_1]\!] \mid a_2[\![!\mathsf{accept}\ h_2]\!]$$
$$\xrightarrow{\mathsf{exit/expel}(h)}$$
$$a_1[\![b_1[\![!(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_2)]\!] \mid W(b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!]) \mid !\mathsf{accept}\ h_1]\!] \mid a_2[\![!\mathsf{accept}\ h_2]\!]$$
$$\rightarrow$$
$$\mathsf{expelError}\ \ G_{EnvVirus}[\![G_{Bact}]\!]$$
3. The coat can move in the ambient a_2 and then expel the bacterium:
$$b_1[\![b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!] \mid !(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_1)]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!] \mid a_2[\![!\mathsf{accept}\ h_2]\!]$$
$$\xrightarrow{\mathsf{enter/accept}(h_2)}$$
$$a_2[\![b_1[\![b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!] \mid !(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_2)]\!] \mid !\mathsf{accept}\ h_2]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!]$$
$$\xrightarrow{\mathsf{exit/expel}(h)}$$
$$a_2[\![b_1[\![!(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_2)]\!] \mid W(b_2[\![\mathsf{exit}\ h \mid \mathsf{enter}\ h_2]\!]) \mid !\mathsf{accept}\ h_2]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!]$$
$$\rightarrow$$
$$b_1[\![!(\mathsf{expel}\ h + \mathsf{enter}\ h_1 + \mathsf{enter}\ h_1)]\!] \mid a_1[\![!\mathsf{accept}\ h_1]\!] \mid a_2[\![b_2[\![\mathsf{enter}\ h_2]\!] \mid !\mathsf{accept}\ h_2]\!]$$

6 Conclusions

The most common approach of biologists to describe biological systems is based on the use of deterministic mathematical means (like, e.g., ODE), and makes it possible to abstractly reason on the behaviour of

biological systems and to perform a quantitative *in silico* investigation. This kind of modelling, however, becomes more and more difficult, both in the specification phase and in the analysis processes, when the complexity of the biological systems taken into consideration increases. This has probably been one of the main motivations for the application of Computer Science formalisms to the description of biological systems [24]. Other motivations can also be found in the fact that the use of formal methods from Computer Science permits the application of analysis techniques that are practically unknown to biologists, such as, for example, static analysis and model checking.

Different formalisms have either been applied to (or have been inspired from) biological systems. The most notable are automata-based models [2, 18], rewrite systems [13, 19], and process calculi [24, 25, 23, 9, 22]. Automata-based models have the advantage of allowing the direct use of many verification tools, such as, for example, model checkers. On the other side, models based on rewrite systems describe biological systems with a notation that can be easily understood by biologists. However, automata-like models and rewrite systems are not compositional. The possibility to study in a componentwise way the behaviour of a system is, in general, naturally ensured by process calculi, included those commonly used to describe biological systems.

In this paper we have laid the foundations for a type system for the BioAmbients calculus suitable to guarantee compatible compartments nesting (due to some intrinsical biological properties). In this framework, the correctness of the *enter*/*accept* capabilities can be checked statically, while the *merge* capability and the *exit*/*expel* capabilities could cause the movement of an ambient of type G within an ambient of type G' and dynamically rise an error. We used our type discipline to model how incompatible blood transfusion could cause the system to rise an error, or to represent the movement of bacteria spore into friendly environments where they can germinate and restart their activity.

Acknowledgments We would like to warmly thank Mariangiola Dezani-Ciancaglini who encouraged us to write this paper and gave us crucial suggestions.

References

[1] Biocham. available at `http://contraintes.inria.fr/BIOCHAM/`.

[2] Rajeev Alur, Calin Belta, Vijay Kumar, and Max Mintz. Hybrid modeling and simulation of biomolecular networks. In *Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, pages 19–32. Springer-Verlag, 2001.

[3] Bogdan Aman, Mariangiola Dezani-Ciancaglini, and Angelo Troina. Type disciplines for analysing biologically relevant properties. In *Proc. of MeCBIC'08*, volume 227 of *ENTCS*, pages 97 – 111. Elsevier, 2009.

[4] Roberto Barbuti, Andrea Maggiolo-schettini, and Paolo Milazzo. Extending the calculus of looping sequences to model protein interaction at. In *Proc. of ISBRA'07*, volume 4463 of *LNBI*, pages 638 – 649. Springer-Verlag, 2006.

[5] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo, Paolo Tiberi, and Angelo Troina. Stochastic calculus of looping sequences for the modelling and simulation of cellular pathways. *Transactions on Computational Systems Biology*, IX:86 – 113, 2008.

[6] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo, and Aangelo Troina. A calculus of looping sequences for modelling microbiological systems. *Fundamenta Informaticæ*, 72(1–3):21–35, 2006.

[7] Livio Bioglio, Mariangiola Dezani-Ciancaglini, Paola Giannini, and Angelo Troina. Type directed semantics for the calculus of looping sequences. *Submitted to IJSI*, 2009.

[8] Linda Brodo, Pierpaolo Degano, and Corrado Priami. A stochastic semantics for Bioambients. In *Proc. of PaCT'07*, volume 4671 of *LNCS*, pages 22 – 34. Springer-Verlag, 2007.

[9] Luca Cardelli. Brane calculi - interactions of biological membranes. In *Computational Methods in Systems Biology*, pages 257 – 278. Springer, 2004.

[10] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Types for the ambient calculus. *Information and Computation*, 177(2):160 – 194, 2002.

[11] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Proc. of FOSSACS'98*, volume 1378 of *LNCS*, pages 140 – 155. Springer-Verlag, 1998.

[12] Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Ivano Salvo. M3: Mobility types for mobile processes in mobile ambients. *Electronic Notes in Theoretical Computer Science*, 78:144 – 177, 2003. CATS'03, Computing: the Australasian Theory Symposium.

[13] Vincent Danos and Cosimo Laneve. Core formal molecular biology. In *Proc. of ESOP'03*, volume 2618 of *LNCS*, pages 302 – 318. Springer-Verlag, 2003.

[14] Mariangiola Dezani-Ciancaglini, Paola Giannini, and Angelo Troina. A Type System for a Stochastic CLS. In *Proc. of MeCBIC'09*, volume 11 of *EPTCS*, pages 91 – 106, 2009.

[15] Mariangiola Dezani-Ciancaglini, Paola Giannini, and Angelo Troina. A Type System for Required/Excluded Elements in CLS. In *Proc. of DCM'09*, volume 9 of *EPTCS*, pages 38 – 48, 2009.

[16] François Fages and Sylvain Soliman. Abstract interpretation and types for systems biology. *Theoretical Computer Science*, 403(1):52–70, 2008.

[17] Cheng Fu, Zhengwei Qiand, and Jinyuan You. A bioambients based framework for chain-structured biomolecules modelling. 314:455 – 459, 2005.

[18] Hiroshi Matsuno, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Hybrid Petri Net representation of gene regulatory networks. In *Proc. of PSB'00*, pages 338 – 349, 2000.

[19] Gheorghe Păun. *Membrane Computing. An Introduction*. Springer-Verlag, 2002.

[20] Andrew Phillips. An abstract machine for the stochastic bioambient calculus. In *Proc. of MeCBIC'08*, volume 227 of *ENTCS*, pages 143 – 159, 2009.

[21] Henrik Pilegaard, Flemming Nielson, and Hanne Riis Nielson. Pathway analysis for bioambients. *Journal of Logic and Algebraic Programming*, 77(1-2):92 – 130, 2008. Proc. of NWPT'06.

[22] Corrado Priami and Paola Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology*, volume 3082, pages 20–33, 2005.

[23] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients: an abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141 – 167, 2004. Computational Systems Biology.

[24] Aviv Regev and Ehud Shapiro. Cells as computation. *Nature*, 419(6905):343, September 2002.

[25] Aviv Regev and Ehud Shapiro. The π-calculus as an abstraction for biomolecular systems. *Modelling in Molecular Biology*, pages 219 – 266, 2004.