Spatial Analysis of BioAmbients*

Hanne Riis Nielson, Flemming Nielson, and Henrik Pilegaard

Technical University of Denmark
{riis,nielson,hepi}@imm.dtu.dk

Abstract. Programming language technology can contribute to the development and understanding of Systems Biology by providing formal calculi for specifying and analysing the dynamic behaviour of biological systems. Our focus is on BioAmbients, a variation of the ambient calculi developed for modelling mobility in computer systems. We present a static analysis for capturing the spatial structure of biological systems and we illustrate it on a few examples.

1 Introduction and Motivation

Systems biology is an approach to studying biological phenomena that is based on a *high level* view of biological systems. The main focus is not the *structure* of biological components but rather the *dynamics* of these components. This poses a challenge for computer science: can programming language technology be used to model and analyse not only the structure of biological processes but also their evolution?

Pioneering work by Shapiro et al [15] demonstrated how biological processes could be specified in the π-calculus [7]; the formalism showed its strength at the molecular and biochemical level but it was less successful at the higher abstraction levels where compartments play a central role. Here, on the other hand, a version of the Ambient Calculus [3], called BioAmbients [13, 14], shows promise as the hierarchical structure of the ambients is very similar to that of compartments; the main difference between the two calculi is in the choice of the primitives for modelling the interaction between ambients or compartments. Surely biological systems are very complex and the scope for developing calculi of computation that capture various aspects of these systems is endless; recent work includes [2, 5, 8, 12].

The main goal, of course, is to capture the behaviour of biological systems in a faithful manner. Over the years biologists have collected observations about biological systems in large databases and it is important to investigate to what extent our models can explain these data in a satisfactory way. It turns out that many of the observations collected by the biologists concern *spatial properties* (as opposed to temporal properties) and this is where static analysis – and the present paper – gets into the picture.

* This research has been supported by the LoST project (number 21-02-0507) funded the Danish Natural Science Research Council.

R. Giacobazzi (Ed.): SAS 2004, LNCS 3148, pp. 69–83, 2004.

Overview of the Paper. In Section 2 we present the syntax and semantics of BioAmbients. The spatial analysis is developed in two stages: First a *compatibility analysis* is developed in Section 3; it computes an over-approximation of the possible interactions within the system of interest. This information is then used in the *spatial analysis* presented in Section 4; this analysis contains a novel treatment of recursion and a new technique for reducing the space complexity of the analysis. Finally, Section 5 illustrates our approach on a few examples and contains our concluding remarks.

2 BioAmbients

BioAmbients [13, 14] differ from Mobile Ambients [3] and its siblings Safe Ambients [6], Boxed Ambients [1] and Discretionary Ambients [11] in a number of ways. The most important difference is that the names (or identities) of the ambients do not control the interaction between ambients, but rather names (of channels) serve that purpose. BioAmbients follow the approach of safe and discretionary ambients and specify interactions by matching capabilities and co-capabilities; the communication primitives have some reminiscents of boxed ambients in that communication can occur across ambient boundaries but it is based on channels as in the π-calculus.

BioAmbients deviate from the other ambient calculi in having a non-deterministic choice operation in addition to the construct for parallelism (just as the π-calculus [7]). The pioneering development presented in [13, 14] observes the need to use a general recursion construct in order to faithfully model biological systems but the theoretical development is only performed for the classical replication construct. To be able to analyse such examples we shall therefore study a version of BioAmbients with a general recursion operator and, as we shall see in later sections, this poses some interesting technical challenges for the theoretical properties of the analysis.

The syntax of BioAmbients is given in Table 1; here we write P for processes and M for capabilities. Each ambient has an *identity* $\mu \in \mathbf{Ambient}$ and each capability has a *label* $\ell \in \mathbf{Lab}$; these annotations have no semantic significance but are useful as "pointers" into the process and also serve a rôle in the analysis. (We shall not require that identities or labels are unique.) Furthermore, each name has a *canonical name* $\lfloor n \rfloor \in \mathbf{Name}$ and we shall demand that alpha-renaming preserves the canonical name; consequently it will be the canonical name rather than the name that will be recorded in the analysis. For the sake of simplicity, we shall assume that a subset $\mathbf{C} \subseteq \mathbf{Name}$ of the canonical names is reserved for constants and below we shall require that names introduced by $(n)P$ satisfy $\lfloor n \rfloor \in \mathbf{C}$. The capabilities M are based on names and hence we shall write $\lfloor M \rfloor \in \mathbf{Cap}$ for the corresponding *canonical capability* obtained by replacing the names with the corresponding canonical names. The input capabilities ($n?\{p\}$, etc.) introduce new names p acting as placeholders (or variables); below we shall require that $\lfloor p \rfloor \in \mathbf{V}$ where $\mathbf{V} = \mathbf{Name} \setminus \mathbf{C}$. Finally, processes may be recursively defined using the construct $\mathsf{rec}\, X.\, P$ and to simplify the development

Table 1. Syntax of processes P and capabilities M.

$P ::=$	$\mathsf{0}$		inactive process
$\mid$	$(n)P$		binding box for the name n
$\mid$	$[P]^{\mu}$		ambient P with the identity μ
$\mid$	$M^{\ell}.P$		prefixing with the capability M labelled ℓ
$\mid$	$P \mid P'$		parallel processes
$\mid$	$P + P'$		non-deterministic (external) choice
$\mid$	$\mathsf{rec}\, X.\, P$		recursive process ($X = P$)
$\mid$	X		process identifier
$M ::=$	$\mathsf{enter}\ n$	$\mid \mathsf{accept}\ n$	enter movement
$\mid$	$\mathsf{exit}\ n$	$\mid \mathsf{expel}\ n$	exit movement
$\mid$	$\mathsf{merge}\text{–}\ n$	$\mid \mathsf{merge}\text{+}\ n$	merge movement
$\mid$	$n!\{m\}$	$\mid n?\{p\}$	local communication
$\mid$	$n\_!\{m\}$	$\mid n\hat{}?\{p\}$	communication to child
$\mid$	$n\hat{}!\{m\}$	$\mid n\_?\{p\}$	communication to parent
$\mid$	$n\#!\{m\}$	$\mid n\#?\{p\}$	communication between siblings

we shall require that X indeed occurs inside P; obviously the usual replication operation $!\,P$ can be obtained as $\mathsf{rec}\, X.\,(P \mid X)$ (assuming X does not occur free in P). Analogously to the treatment of names we shall require that each process identifier X has a canonical identity $\lfloor X \rfloor$ that is preserved by alpha-renaming.

Programs will be processes $P_\star$ satisfying the predicate $\mathsf{PRG}_{\mathbf{C}}(P_\star)$ defined as the conjunction of the following conditions (explained below):

- $P_\star$ has no free process identifiers; formally $\mathsf{fpi}(P_\star) = \emptyset$.
- $P_\star$ only has free names from $\mathbf{C}$; formally $\lfloor \mathsf{fn}(P_\star) \rfloor \subseteq \mathbf{C}$.
- $P_\star$ is well-formed wrt. $\mathbf{C}$; formally $\mathbf{C} \vdash P_\star$.

Here we write $\mathsf{fpi}(P)$ for the set of *free process identifiers* of P and $\mathsf{fn}(P)$ for the *free names* of P; the canonicity operation $\lfloor \cdot \rfloor$ is extended in a pointwise manner to sets of names. The *well-formedness predicate* $\mathbf{C} \vdash P$ serves two purposes: first, it enforces the implicit typing requirements imposed by the division of **Name** into the two disjoint subsets $\mathbf{C}$ and $\mathbf{V}$; secondly, it imposes the condition that process identifiers are actually used recursively in the processes they define. The predicate is formally defined in Table 2 and uses $\mathsf{bn}(M)$ to denote the *bound names* of the capability M. Note that the condition will reject processes like $(n)\,[\mathsf{enter}\ n^{\ell_1} \mid m?\{n\}^{\ell_2}.\,\mathsf{enter}\ n^{\ell_3}]^{\mu}$ where the same name n is introduced as a constant and also introduced in an input capability; a simple alpha-renaming will, of course, solve the problem.

We shall write $P[m/n]$ for the process that is as P except that all free occurrences of the name n are replaced by the name m. Similarly, we shall write $P[Q/X]$ for the process that is as P except that all free occurrences of the process identifier X are replaced by the process Q. In both cases we take care to perform the necessary alpha-renamings (preserving canonicity) to avoid capturing free names or process identifiers.

Table 2. Well-formedness of processes with respect to $\mathbf{C}$: $\mathbf{C} \vdash P$.

$$\mathbf{C} \vdash 0 \qquad \frac{\mathbf{C} \vdash P}{\mathbf{C} \vdash (n)P} \text{ if } \lfloor n \rfloor \in \mathbf{C} \qquad \frac{\mathbf{C} \vdash P}{\mathbf{C} \vdash [P]^{\mu}} \qquad \frac{\mathbf{C} \vdash P}{\mathbf{C} \vdash M^{\ell}.P} \text{ if } \lfloor \mathsf{bn}(M) \rfloor \cap \mathbf{C} = \emptyset$$

$$\frac{\mathbf{C} \vdash P \quad \mathbf{C} \vdash P'}{\mathbf{C} \vdash P \mid P'} \qquad \frac{\mathbf{C} \vdash P \quad \mathbf{C} \vdash P'}{\mathbf{C} \vdash P + P'} \qquad \frac{\mathbf{C} \vdash P}{\mathbf{C} \vdash \mathsf{rec}\, X.\, P} \text{ if } X \in \mathsf{fpi}(P) \qquad \mathbf{C} \vdash X$$

Table 3. Structural congruence: $P \equiv Q$ is the least congruence defined by the above.

Alpha-renaming of bound names and bound process identifiers:

$P \equiv Q$ if P may be alpha-renamed to Q (preserving canonicity)

Reordering of parallel processes:

$$P \mid P' \equiv P' \mid P$$
$$(P \mid P') \mid P'' \equiv P \mid (P' \mid P'')$$
$$P \mid 0 \equiv P$$

Reordering of choice processes:

$$P + P' \equiv P' + P$$
$$(P + P') + P'' \equiv P + (P' + P'')$$
$$P + 0 \equiv P$$

Scope rules for name bindings:

$$(n)0 \equiv 0$$
$$(n)(m)P \equiv (m)(n)P$$
$$(n)([P]^{\mu}) \equiv [(n)P]^{\mu}$$
$$(n)(P \mid P') \equiv ((n)P) \mid P' \quad \text{if } n \notin \mathsf{fn}(P')$$
$$(n)(P + P') \equiv ((n)P) + P' \quad \text{if } n \notin \mathsf{fn}(P')$$

Recursion:

$$\mathsf{rec}\, X.\, P \equiv P[\mathsf{rec}\, X.\, P/X]$$

Example 1. To illustrate the development we consider the following program P_{virus}; as we shall see shortly it models how the gene of a virus may infect a cell:

$$\begin{array}{l} [\ \mathsf{rec}\, X.\ \mathsf{enter}\ n_1^{\ell_1}.\ X + \mathsf{exit}\ n_2^{\ell_2}.\ X + c\hat{}?\{x\}^{\ell_3}.\ \mathsf{expel}\ x^{\ell_4}.\ X \\ \quad \mid [\ \mathsf{exit}\ n_3^{\ell_5}.\ 0\ ]^{gene}\ ]^{virus} \\ \mid [\ \mathsf{rec}\, Y.\ \mathsf{accept}\ n_1^{\ell_6}.\ Y + \mathsf{expel}\ n_2^{\ell_7}.\ Y + c\_!\{n_3\}^{\ell_8}.\ Y\ ]^{cell} \end{array}$$

It is trivial to check that the well-formedness condition is fulfilled for $\mathbf{C} = \{\lfloor c \rfloor, \lfloor n_1 \rfloor, \lfloor n_2 \rfloor, \lfloor n_3 \rfloor\}$. □

Semantics. The semantics follows the standard approach and is specified by the *structural congruence relation* $P \equiv Q$ in Table 3 and the *transition relation* $P \rightarrow Q$ in Table 4. The congruence relation uses a disciplined notion of alpha-renaming that preserves canonicity. The movement interactions merely give rise to a rearrangement of the ambient structure where some potential continuations are excluded (due to the presence of the non-deterministic choice operation). The communication interactions also exclude some potential continuations but they

Table 4. Transition relation: $P \to Q$.

Movement of ambients:

$$[(\mathsf{enter}\ n^{\ell_1}.\,P + P') \mid P'']^{\mu_1} \mid [(\mathsf{accept}\ n^{\ell_2}.\,Q + Q') \mid Q'']^{\mu_2} \to [\,[P \mid P'']^{\mu_1} \mid Q \mid Q'']^{\mu_2}$$

$$[\,[(\mathsf{exit}\ n^{\ell_1}.\,P + P') \mid P'']^{\mu_1} \mid (\mathsf{expel}\ n^{\ell_2}.\,Q + Q') \mid Q'']^{\mu_2} \to [P \mid P'']^{\mu_1} \mid [Q \mid Q'']^{\mu_2}$$

$$[(\mathsf{merge}\text{–}\ n^{\ell_1}.\,P + P') \mid P'']^{\mu_1} \mid [(\mathsf{merge}\text{+}\ n^{\ell_2}.\,Q + Q') \mid Q'']^{\mu_2} \to [P \mid P'' \mid Q \mid Q'']^{\mu_2}$$

Communication between ambients:

$$(n!\{m\}^{\ell_1}.\,P + P') \mid (n?\{p\}^{\ell_2}.\,Q + Q') \to P \mid Q[m/p]$$

$$(n\_!\{m\}^{\ell_1}.\,P + P') \mid [(n\hat{}?\{p\}^{\ell_2}.\,Q + Q') \mid Q'']^{\mu} \to P \mid [Q[m/p] \mid Q'']^{\mu}$$

$$[(n\hat{}!\{m\}^{\ell_1}.\,P + P') \mid P'']^{\mu} \mid (n\_?\{p\}^{\ell_2}.\,Q + Q') \to [P \mid P'']^{\mu} \mid Q[m/p]$$

$$[(n\#!\{m\}^{\ell_1}.\,P + P') \mid P'']^{\mu_1} \mid [(n\#?\{p\}^{\ell_2}.\,Q + Q') \mid Q'']^{\mu_2} \to [P \mid P'']^{\mu_1} \mid [Q[m/p] \mid Q'']^{\mu_2}$$

Execution in context:

$$\frac{P \to Q}{(n)P \to (n)Q} \qquad \frac{P \to Q}{[P]^{\mu} \to [Q]^{\mu}} \qquad \frac{P \to Q}{P \mid R \to Q \mid R} \qquad \frac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q}$$

do not modify the overall ambient structure; however, some of the processes are modified in order to reflect the new binding of names. The semantics of recursion amounts to a straightforward unfolding in the congruence relation; this is more general than the overly restrictive semantics used in [9].

Example 2. The semantics of the program P_{virus} of Example 1 is illustrated on Figure 1. The initial configuration is shown in the upper leftmost frame where the tree structure reflects that *cell* and *virus* are siblings (with a common father denoted $\top$) and *gene* is a subambient of *virus*. The first step of the semantics will be for *virus* to move into *cell* using the pair $(\mathsf{enter}\ n_1^{\ell_1}, \mathsf{accept}\ n_1^{\ell_6})$ of capabilities and we obtain the configuration depicted in the bottom leftmost frame of the figure. Now there are two possibilities: either *virus* moves out of *cell* using the pair $(\mathsf{exit}\ n_2^{\ell_2}, \mathsf{expel}\ n_2^{\ell_7})$ of capabilities and we are back in the initial configuration at the top or, alternatively, there is a communication from *cell* to *virus* over the name c using the pair $(c\_!\{n_3\}^{\ell_8}, c\hat{}?\{x\}^{\ell_3})$ of capabilities during which x is bound to n_3 as indicated in the corresponding frame of the figure. The pair $(\mathsf{exit}\ n_3^{\ell_5}, \mathsf{expel}\ n_3^{\ell_4})$ of capabilities will now move *gene* out of *virus* and we reach a configuration where *virus* can exit and enter *cell* any number of times or the communication over c may happen again after which the system ends in a stuck configuration (shown in the top rightmost frame of the figure). □

3 Compatibility Analysis

The aim of the spatial analysis is to extract an over-approximation of the possible hierarchial structures of the ambients. For this we need to approximate the potential interactions between the ambients and motivated by [4] we shall

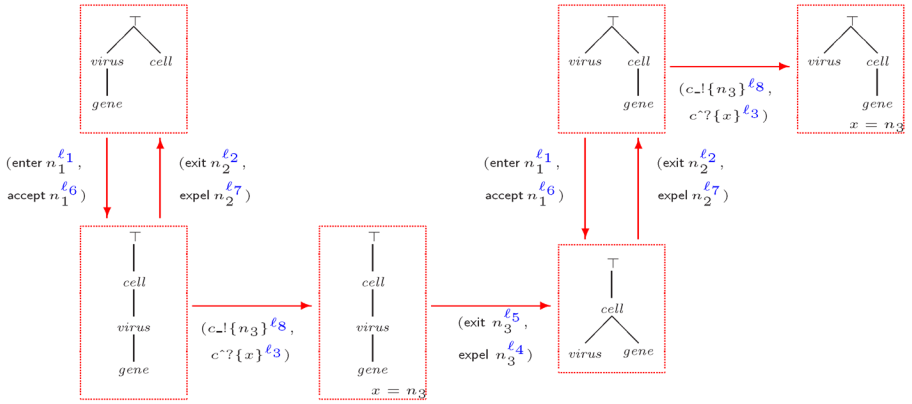

Fig. 1. Illustration of the semantics of the running example.

develop a compatibility analysis. Given a process P, the aim of the compatibility analysis is to identify pairs of labelled capabilities that, from a syntactic point of view, *may* engage in a transition. Intuitively, this means that the two capabilities must match and that it must be possible for them to occur in parallel processes. As an example, in $[\mathsf{enter}\ n^{\ell_1}]^{\mu_1} \mid [\mathsf{accept}\ m^{\ell_2}]^{\mu_2}$ the capabilities labelled ℓ_1 and ℓ_2 may interact because from a syntactic point of view we cannot preclude that n and m may turn out to be equal; however, if we replace the parallel composition with a non-deterministic choice then they will never be able to interact.

The matching condition will ignore the actual names occurring in the capabilities (because even the canonical names are not preserved under reduction) and to formalise this we shall introduce the notion of a *skeleton capability*: $\lceil M \rceil$ is simply obtained from M by replacing all names in M with the token " $\cdot$ ". The *matching condition* on skeleton capabilities can now be expressed by the predicate

$$\mathsf{match}(\lceil M_1 \rceil, \lceil M_2 \rceil)$$

that holds if and only if

$$(\lceil M_1 \rceil, \lceil M_2 \rceil) \in \{\ (\mathsf{enter}\ \cdot, \mathsf{accept}\ \cdot),\ (\mathsf{exit}\ \cdot, \mathsf{expel}\ \cdot),\ (\mathsf{merge}\text{–}\ \cdot, \mathsf{merge}\text{+}\ \cdot),\ (\cdot?\{\cdot\}, \cdot!\{\cdot\}),\ (\cdot\hat{}?\{\cdot\}, \cdot\_!\{\cdot\}),\ (\cdot\_?\{\cdot\}, \cdot\hat{}!\{\cdot\}),\ (\cdot\#?\{\cdot\}, \cdot\#!\{\cdot\})\}$$

In order to define the compatibility information we shall first need to extract the set of *labelled skeleton capabilities* occurring within a process. This is done using the function $\mathsf{caps}_\Gamma(P)$ of Table 5; here Γ is a mapping that to each process identifier associates a set of labelled skeleton capabilities. The mapping Γ is useful later (in the definition of comp) when we encounter subprocesses with free process identifiers.

The *compatibility information* is then obtained using the function $\mathsf{comp}_{\Gamma\Delta}(P)$ of Table 5. Here Γ is as above whereas Δ is a mapping that to each process

Table 5. Capabilities, $\mathsf{caps}_\Gamma(P)$, and compatible pairs of capabilities, $\mathsf{comp}_{\Gamma\Delta}(P)$.

$$
\begin{array}{rlrl}
\mathsf{caps}_\Gamma(0) &= \emptyset & \mathsf{caps}_\Gamma((n)P) &= \mathsf{caps}_\Gamma(P) \\
\mathsf{caps}_\Gamma([P]^\mu) &= \mathsf{caps}_\Gamma(P) & \mathsf{caps}_\Gamma(M^\ell.P) &= \{\lceil M\rceil^\ell\} \cup \mathsf{caps}_\Gamma(P) \\
\mathsf{caps}_\Gamma(P \mid P') &= \mathsf{caps}_\Gamma(P) \cup \mathsf{caps}_\Gamma(P') & \mathsf{caps}_\Gamma(P + P') &= \mathsf{caps}_\Gamma(P) \cup \mathsf{caps}_\Gamma(P') \\
\mathsf{caps}_\Gamma(\mathsf{rec}\, X.\, P) &= \mathsf{caps}_{\Gamma[X\mapsto\emptyset]}(P) & \mathsf{caps}_\Gamma(X) &= \Gamma(X)
\end{array}
$$

$$
\begin{array}{rlrl}
\mathsf{comp}_{\Gamma\Delta}(0) &= \emptyset & \mathsf{comp}_{\Gamma\Delta}((n)P) &= \mathsf{comp}_{\Gamma\Delta}(P) \\
\mathsf{comp}_{\Gamma\Delta}([P]^\mu) &= \mathsf{comp}_{\Gamma\Delta}(P) & \mathsf{comp}_{\Gamma\Delta}(M^\ell.P) &= \mathsf{comp}_{\Gamma\Delta}(P) \\
\mathsf{comp}_{\Gamma\Delta}(P \mid P') &= \mathsf{comp}_{\Gamma\Delta}(P) & \mathsf{comp}_{\Gamma\Delta}(P + P') &= \mathsf{comp}_{\Gamma\Delta}(P) \\
& \quad \cup \mathsf{comp}_{\Gamma\Delta}(P') & & \quad \cup \mathsf{comp}_{\Gamma\Delta}(P') \\
& \quad \cup \mathsf{cross}_\Gamma(P, P') & & \\
\mathsf{comp}_{\Gamma\Delta}(\mathsf{rec}\, X.\, P) &= \mathsf{comp}_{\Gamma'\Delta[X\mapsto\emptyset]}(P) & \mathsf{comp}_{\Gamma\Delta}(X) &= \Delta(X) \\
\multicolumn{4}{l}{\text{where } \Gamma' = \Gamma[X \mapsto \mathsf{caps}_{\Gamma[X\mapsto\emptyset]}(P)]}
\end{array}
$$

$$
\begin{array}{rl}
\mathsf{cross}_\Gamma(P, P') = & \{(\lceil M_1\rceil^{\ell_1}, \lceil M_2\rceil^{\ell_2}) \in (\mathsf{caps}_\Gamma(P) \times \mathsf{caps}_\Gamma(P')) \cup (\mathsf{caps}_\Gamma(P') \times \mathsf{caps}_\Gamma(P)) \\
& \mid \mathsf{match}(\lceil M_1\rceil, \lceil M_2\rceil)\}
\end{array}
$$

identifier associates a set of pairs of labelled skeleton capabilities; again we are parametric on Γ and Δ so that we can handle processes with free process identifiers. In the case of parallel composition the definition of comp uses the auxiliary operation cross to record that capabilities in the two branches may interact with one another and the caps function is used in order to specify this. This is in contrast to the definition provided for non-deterministic choice where it is known that capabilities from the two branches never will interact.

Example 3. For the running example P_{virus} of Examples 1 and 2 we get:

$$
\begin{array}{rl}
\mathsf{CP}_{\mathsf{virus}} = \{ & (\mathsf{enter}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_6}), \\
& (\mathsf{exit}\ \cdot^{\ell_2}, \mathsf{expel}\ \cdot^{\ell_7}),\ (\mathsf{exit}\ \cdot^{\ell_5}, \mathsf{expel}\ \cdot^{\ell_4}),\ (\mathsf{exit}\ \cdot^{\ell_5}, \mathsf{expel}\ \cdot^{\ell_7}), \\
& (\cdot\_!\{\cdot\}^{\ell_8}, \cdot\hat{}?\{\cdot\}^{\ell_3})\ \}
\end{array}
$$

Comparing with Figure 1 we see that this is indeed an over-approximation of the actual interactions that can take place: the pair $(\mathsf{exit}\ \cdot^{\ell_5}, \mathsf{expel}\ \cdot^{\ell_7}) \in \mathsf{CP}_{\mathsf{virus}}$ has no analogue in Figure 1. □

Example 4. Consider the artificial variant P'_{virus} of the process of Example 1 where the virus exists in two variants, one with a gene much as before and one with a harmless gene:

$$
\begin{array}{l}
[\,\mathsf{rec}\, X.\ \mathsf{enter}\ n_1^{\ell_1}.\ X + \mathsf{exit}\ n_2^{\ell_2}.\ X + c\hat{}?\{x\}^{\ell_3}.\ \mathsf{expel}\ x^{\ell_4}.\ X \\
\quad \mid (\ [\mathsf{exit}\ n_3^{\ell_5}.\ 0 + \mathsf{accept}\ n_4^{\ell_6}.\ 0\,]^{gene_1} + [\mathsf{enter}\ n_4^{\ell_7}.\ 0\,]^{gene_2}\ )\,]^{virus} \\
\mid [\mathsf{rec}\, Y.\ \mathsf{accept}\ n_1^{\ell_8}.\ Y + \mathsf{expel}\ n_2^{\ell_9}.\ Y + c\_!\{n_3\}^{\ell_{10}}.\ Y\,]^{cell}
\end{array}
$$

The compatibility analysis will compute the following information:

$$
\begin{array}{rl}
\mathsf{CP}'_{\mathsf{virus}} = \{ & (\mathsf{enter}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_6}),\ (\mathsf{enter}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_8}),\ (\mathsf{enter}\ \cdot^{\ell_7}, \mathsf{accept}\ \cdot^{\ell_8}), \\
& (\mathsf{exit}\ \cdot^{\ell_2}, \mathsf{expel}\ \cdot^{\ell_9}),\ (\mathsf{exit}\ \cdot^{\ell_5}, \mathsf{expel}\ \cdot^{\ell_4}),\ (\mathsf{exit}\ \cdot^{\ell_5}, \mathsf{expel}\ \cdot^{\ell_9}), \\
& (\cdot\_!\{\cdot\}^{\ell_{10}}, \cdot\hat{}?\{\cdot\}^{\ell_3})\ \}
\end{array}
$$

Note that despite the over-approximation this correctly captures that for example the capabilities labelled ℓ_7 and ℓ_6 of the two genes never will be able to interact. □

The correctness of the compatibility analysis follows from:

Lemma 1. *If* $P \equiv Q$ *and* $\mathbf{C} \vdash P$ *then* $\mathsf{comp}_{\Gamma\Delta}(P) = \mathsf{comp}_{\Gamma\Delta}(Q)$. *If* $P \rightarrow Q$ *and* $\mathbf{C} \vdash P$ *then* $\mathsf{comp}_{\Gamma\Delta}(Q) \subseteq \mathsf{comp}_{\Gamma\Delta}(P)$.

In the subsequent analyses we shall make use of the compatibility relation for the overall program $P_\star$ of interest. Writing $[\,]$ for the empty mapping we shall use the abbreviation $\mathsf{CP}_\star$ for $\mathsf{comp}_{[\,][\,]}(P_\star)$ thereby exploiting that $P_\star$ has no free process identifiers. Thus it follows from Lemma 1 that if $\mathsf{PRG}_{\mathbf{C}}(P_\star)$ and $P_\star \rightarrow^* Q$ then $\mathsf{comp}_{[\,][\,]}(Q) \subseteq \mathsf{CP}_\star$ so $\mathsf{CP}_\star$ remains a correct over-approximation.

4 Spatial Analysis

We are now ready to embark on the spatial analysis: for a program $P_\star$ we want to approximate what ambients may turn up inside what other ambients. To extract this information we shall develop an analysis extracting the following information:

- An approximation of the contents of ambients:

$$\mathcal{I} \subseteq \mathbf{Ambient} \times (\mathbf{Ambient} \cup (\mathbf{Cap} \times \mathbf{Lab}))$$

  Here $\mu' \in \mathcal{I}(\mu)$ means that μ' may be a subambient of the ambient μ and $\lfloor M \rfloor^\ell \in \mathcal{I}(\mu)$ means that the labelled *canonical* capability $\lfloor M \rfloor^\ell$ may be within the ambient μ.
- An approximation of the relevant name bindings:

$$\mathcal{R} \subseteq \mathbf{V} \times \mathbf{C} \qquad (\subseteq \mathbf{Name} \times \mathbf{Name})$$

  Here $\nu' \in \mathcal{R}(\nu)$ means that the constant (canonical) name ν' may be bound to the variable (canonical) name ν.

The judgements of the analysis take the form

$$(\mathcal{I}, \mathcal{R}) \models^\mu P$$

and express that when the subprocess P (of $P_\star$) is enclosed within an ambient with the identity $\mu \in \mathbf{Ambient}$ then $\mathcal{I}$ and $\mathcal{R}$ correctly capture the behaviour of P – meaning that $\mathcal{I}$ will reflect the contents of the ambients as P evolves inside $P_\star$ and $\mathcal{R}$ will contain all the bindings of names that take place. The analysis is specified in Table 6 and refers to Table 7 for auxiliary information about the recursion construct and to Table 8 for a specification of the closure conditions $\mathsf{closure}_{\lceil M \rceil}$. Below we comment on the clauses.

Table 6. Analysis specification: $(\mathcal{I},\mathcal{R}) \models^{\mu} P$.

$(\mathcal{I},\mathcal{R}) \models^{\mu} \mathsf{0}$	iff	true
$(\mathcal{I},\mathcal{R}) \models^{\mu} (n)P$	iff	$(\mathcal{I},\mathcal{R}) \models^{\mu} P$
$(\mathcal{I},\mathcal{R}) \models^{\mu} [P]^{\mu'}$	iff	$\mu' \in \mathcal{I}(\mu) \wedge (\mathcal{I},\mathcal{R}) \models^{\mu'} P$
$(\mathcal{I},\mathcal{R}) \models^{\mu} M^{\ell}.P$	iff	$\lfloor M \rfloor^{\ell} \in \mathcal{I}(\mu) \wedge (\mathcal{I},\mathcal{R}) \models^{\mu} P \wedge \mathsf{closure}_{\lceil M \rceil}$
$(\mathcal{I},\mathcal{R}) \models^{\mu} P \mid P'$	iff	$(\mathcal{I},\mathcal{R}) \models^{\mu} P \wedge (\mathcal{I},\mathcal{R}) \models^{\mu} P'$
$(\mathcal{I},\mathcal{R}) \models^{\mu} P + P'$	iff	$(\mathcal{I},\mathcal{R}) \models^{\mu} P \wedge (\mathcal{I},\mathcal{R}) \models^{\mu} P'$
$(\mathcal{I},\mathcal{R}) \models^{\mu} \mathsf{rec}\, X.\, P$	iff	$\forall \mu' : \mu' \in \mathcal{L}_{\lfloor X \rfloor}(\mathsf{G}^{\mu}(\mathsf{rec}\, X.\, P)) \Rightarrow (\mathcal{I},\mathcal{R}) \models^{\mu'} P$
$(\mathcal{I},\mathcal{R}) \models^{\mu} X$	iff	true

Table 7. Auxiliary analysis information for recursion: $\mathsf{G}^{\delta}(P)$.

$\mathsf{G}^{\delta}(\mathsf{0}) = \emptyset$	$\mathsf{G}^{\delta}((n)P) = \mathsf{G}^{\delta}(P)$
$\mathsf{G}^{\delta}([P]^{\mu}) = \mathsf{G}^{\mu}(P)$	$\mathsf{G}^{\delta}(M^{\ell}.P) = \mathsf{G}^{\delta}(P)$
$\mathsf{G}^{\delta}(P \mid P') = \mathsf{G}^{\delta}(P) \cup \mathsf{G}^{\delta}(P')$	$\mathsf{G}^{\delta}(P + P') = \mathsf{G}^{\delta}(P) \cup \mathsf{G}^{\delta}(P')$
$\mathsf{G}^{\delta}(\mathsf{rec}\, X.\, P) = \mathsf{G}^{\lfloor X \rfloor}(P) \cup \{\lfloor X \rfloor \to \delta\}$	$\mathsf{G}^{\delta}(X) = \{\lfloor X \rfloor \to \delta\}$

Table 6 specifies a simple syntax directed traversal of the process with the clauses for ambients and capabilities being two of the more interesting ones as they check that $\mathcal{I}$ contains the correct initial information. The clause for $(n)P$ is very simple since n is a constant (in contrast to a variable); in particular there is no need to impose any requirements on $\mathcal{R}$. The clauses for the parallel and the choice constructs look exactly the same; however, the use of the compatibility information in the closure conditions of Table 8 ensures that they are indeed handled differently.

The clause for recursion ensures that the analysis result is valid in *all* the contexts in which the recursion construct $\mathsf{rec}\, X.\, P$ may be encountered including those arising from its unfolding. These contexts are provided by the auxiliary operation $\mathsf{G}^{\delta}(P)$ (see Table 7) that constructs a simple *regular grammar* for the potential contexts of the process identifiers. The non-terminals of the grammar are the canonical process identifiers, the terminal symbols are the ambient identities and the right hand side of the productions will contain exactly one (non-terminal or terminal) symbol. The language generated by the grammar $\mathsf{G}^{\mu}(\mathsf{rec}\, X.\, P)$ when $\lfloor X \rfloor$ is the start symbol is written $\mathcal{L}_{\lfloor X \rfloor}(\mathsf{G}^{\mu}(\mathsf{rec}\, X.\, P))$ and it approximates the contexts in which the recursion construct may be encountered. This language is clearly finite. As an example, for the process $[\mathsf{rec}\, X.\, \mathsf{rec}\, Y.\, (X \mid [Y]^{\mu_2})]^{\mu_1}$ we obtain a grammar with the productions $\{\lfloor X \rfloor \to \mu_1, \lfloor Y \rfloor \to \lfloor X \rfloor, \lfloor X \rfloor \to \lfloor Y \rfloor, \lfloor Y \rfloor \to \mu_2\}$. The language generated by this grammar by the non-terminal $\lfloor X \rfloor$ is $\{\mu_1, \mu_2\}$ reflecting that the outermost recursion may occur in both contexts as can be seen by unfolding both X and Y once.

Turning to the *closure conditions* of Table 8 we first observe that there are two clauses for each matching pair of skeleton capabilities and one of these is trivial. In each case the pre-condition of the non-trivial clause checks whether

Table 8. Closure condition on $\mathcal{I}$ and $\mathcal{R}$.

$$
\begin{array}{ll}
\mathsf{closure}_{\mathsf{enter}\ \cdot} & = \forall \mu, \mu_1, \mu_2, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \mathsf{enter}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu_1) \wedge \mu_1 \in \mathcal{I}(\mu) \wedge \mathsf{accept}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_2) \wedge \mu_2 \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\mathsf{enter}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_2}) \\
& \quad \Rightarrow \quad \mu_1 \in \mathcal{I}(\mu_2) \\
\mathsf{closure}_{\mathsf{accept}\ \cdot} & = \mathsf{true} \\
\mathsf{closure}_{\mathsf{exit}\ \cdot} & = \forall \mu, \mu_1, \mu_2, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \mathsf{exit}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_2) \wedge \mathsf{expel}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_2) \wedge \mu_2 \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\mathsf{exit}\ \cdot^{\ell_1}, \mathsf{expel}\ \cdot^{\ell_2}) \\
& \quad \Rightarrow \quad \mu_1 \in \mathcal{I}(\mu) \\
\mathsf{closure}_{\mathsf{expel}\ \cdot} & = \mathsf{true} \\
\mathsf{closure}_{\mathsf{merge-}\ \cdot} & = \forall \mu, \mu_1, \mu_2, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \mathsf{merge-}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu_1) \wedge \mu_1 \in \mathcal{I}(\mu) \wedge \mathsf{merge+}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_2) \wedge \mu_2 \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\mathsf{merge-}\ \cdot^{\ell_1}, \mathsf{merge+}\ \cdot^{\ell_2}) \\
& \quad \Rightarrow \quad \mathcal{I}(\mu_1) \subseteq \mathcal{I}(\mu_2) \\
\mathsf{closure}_{\mathsf{merge+}\ \cdot} & = \mathsf{true} \\
\mathsf{closure}_{\cdot!\{\cdot\}} & = \forall \mu, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \nu_1 ! \{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu) \wedge \nu_2 ? \{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\cdot ! \{\cdot\}^{\ell_1}, \cdot ? \{\cdot\}^{\ell_2}) \\
& \quad \Rightarrow \quad \langle \mathcal{R} \rangle(\nu_m) \subseteq \mathcal{R}(\nu_p) \\
\mathsf{closure}_{\cdot?\{\cdot\}} & = \mathsf{true} \\
\mathsf{closure}_{\cdot\_!\{\cdots\}} & = \forall \mu, \mu_c, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \nu_1 \_ ! \{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu) \wedge \nu_2 \hat{} ? \{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_c) \wedge \mu_c \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\cdot \_ ! \{\cdot\}^{\ell_1}, \cdot \hat{} ? \{\cdot\}^{\ell_2}) \\
& \quad \Rightarrow \quad \langle \mathcal{R} \rangle(\nu_m) \subseteq \mathcal{R}(\nu_p) \\
\mathsf{closure}_{\cdot\hat{}?\{\cdot\}} & = \mathsf{true} \\
\mathsf{closure}_{\cdot\hat{}!\{\cdot\}} & = \forall \mu, \mu_c, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \nu_1 \hat{} ! \{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu_c) \wedge \mu_c \in \mathcal{I}(\mu) \wedge \nu_2 \_ ? \{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\cdot \hat{} ! \{\cdot\}^{\ell_1}, \cdot \_ ? \{\cdot\}^{\ell_2}) \\
& \quad \Rightarrow \quad \langle \mathcal{R} \rangle(\nu_m) \subseteq \mathcal{R}(\nu_p) \\
\mathsf{closure}_{\cdot\_?\{\cdot\}} & = \mathsf{true} \\
\mathsf{closure}_{\cdot\#!\{\cdot\}} & = \forall \mu, \mu_1, \mu_2, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
& \quad \nu_1 \# ! \{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu_1) \wedge \mu_1 \in \mathcal{I}(\mu) \wedge \nu_2 \# ? \{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_2) \wedge \mu_2 \in \mathcal{I}(\mu) \wedge \\
& \quad \langle \mathcal{R} \rangle(\nu_1) \cap \langle \mathcal{R} \rangle(\nu_2) \neq \emptyset \wedge \mathsf{CP}_\star(\cdot \# ! \{\cdot\}^{\ell_1}, \cdot \# ? \{\cdot\}^{\ell_2}) \\
& \quad \Rightarrow \quad \langle \mathcal{R} \rangle(m) \subseteq \mathcal{R}(\nu_p) \\
\mathsf{closure}_{\cdot\#?\{\cdot\}} & = \mathsf{true}
\end{array}
$$

an abstract version of the firing conditions of the corresponding transition rule is fulfilled and the conclusion then records an abstract version of the resulting configuration. The $\mathcal{I}$ relation is used to check the spatial conditions, the $\mathcal{R}$ relation is used to check the potential agreement of names, and the compatibility information of $\mathsf{CP}_\star$ is used to check whether the current pairs of canonical capabilities may interact at all. Since the relation $\mathcal{R}$ is only concerned with the names that act as variables we shall use a slightly modified version of $\mathcal{R}$ namely

Table 9. Three BioAmbient example processes from [14].

Membranal pore :

$$[\,\mathsf{rec}\,X_1.\ \mathsf{enter}\ cell_1^{\ell_1}.\ X_1 + \mathsf{exit}\ cell_2^{\ell_2}.\ X_1\,]^{mol_1}$$
$$\mid [\,\mathsf{rec}\,X_2.\ \mathsf{enter}\ cell_1^{\ell_3}.\ X_2 + \mathsf{exit}\ cell_2^{\ell_4}.\ X_2\,]^{mol_2}$$
$$\mid [\,\mathsf{rec}\,X_3.\ \mathsf{accept}\ cell_1^{\ell_5}.\ X_3 + \mathsf{expel}\ cell_2^{\ell_6}.\ X_3\,]^{cell}$$

Single substrate enzymatic reaction :

$$[\,\mathsf{rec}\,X.\ \mathsf{accept}\ esbind^{\ell_1}.\ (\mathsf{expel}\ unbind^{\ell_2}.\ X + \mathsf{expel}\ react^{\ell_3}.\ X)$$
$$+\mathsf{accept}\ epbind^{\ell_4}.\ (\mathsf{expel}\ unbind^{\ell_5}.\ X + \mathsf{expel}\ react^{\ell_6}.\ X)\,]^{enzyme}$$
$$\mid [\,\mathsf{rec}\,X_1.\ \mathsf{enter}\ esbind^{\ell_7}.\ \mathsf{rec}\,X_2.\ (\mathsf{exit}\ unbind^{\ell_8}.\ X_1$$
$$+\mathsf{exit}\ react^{\ell_9}.\ \mathsf{enter}\ epbind^{\ell_{10}}.\ X_2)\,]^{mol}$$

Two-protein complex :

$$[\,\mathsf{rec}\,X_1.\ \mathsf{merge}\text{–}\ cplx^{\ell_1}.\ brk?\{u\}^{\ell_2}.\ \mathsf{expel}\ brk^{\ell_3}.\ X_1\,]^{mol_1}$$
$$\mid [\,(bb)\,\mathsf{rec}\,X_2.\ \mathsf{merge}+\ cplx^{\ell_4}.\ brk!\{d\}^{\ell_5}.\ bb!\{d\}^{\ell_6}.\ [\mathsf{merge}+\ bb^{\ell_7}.\ \mathsf{exit}\ brk^{\ell_8}.\ X_2\,]^{mol_1}$$
$$\mid \mathsf{rec}\,X_3.\ bb?\{v\}^{\ell_9}.\ [\mathsf{merge}\text{–}\ bb^{\ell_{10}}.\ X_3\,]^{mol_2}\,]^{mol_2}$$

$$\langle\mathcal{R}\rangle \subseteq (\mathbf{V} \cup \mathbf{C}) \times \mathbf{C}$$

that takes care of variables as well as constants; it is defined by:

$$(\forall n : n \in \mathbf{C} \Rightarrow \langle\mathcal{R}\rangle(n,n)) \quad \wedge \quad (\forall n,m : \mathcal{R}(n,m) \Rightarrow \langle\mathcal{R}\rangle(n,m))$$

The analysis result for the program $P_\star$ is then the minimal $\mathcal{I}$ and $\mathcal{R}$ such that $(\mathcal{I},\mathcal{R}) \models^{\top} P_\star$ where $\top$ is the identity of an artificial top-level ambient.

Example 5. The analysis of the running example P_{virus} gives rise to the following minimal $\mathcal{I}$ and $\mathcal{R}$:

μ	$\mathcal{I}(\mu)$
$cell$	$gene, virus, c\_!\{n_3\}^{\ell_8}, \mathsf{expel}\ n_2^{\ell_7}, \mathsf{accept}\ n_1^{\ell_6}$
$gene$	$\mathsf{exit}\ n_3^{\ell_5}$
$virus$	$gene, \mathsf{expel}\ x^{\ell_4}, c\hat{}?\{x\}^{\ell_3}, \mathsf{exit}\ n_2^{\ell_2}, \mathsf{enter}\ n_1^{\ell_1}$
$\top$	$gene, cell, virus$

n	$\mathcal{R}(n)$
x	n_3

Figure 2 (a) gives a graphical representation of the ambient part of the relation $\mathcal{I}$. There is one node for each of the ambient identities and an edge from the node representing μ_1 to the one representing μ_2 if and only if $(\mu_1,\mu_2) \in \mathcal{I}$. The edge is *solid* if (μ_1,μ_2) is introduced into $\mathcal{I}$ by the initialisation rules of Table 6 and it is *dotted* if it is introduced by the closure conditions of Table 8. Note that the trees of the individual frames of Figure 1 are all subgraphs of this figure (as should be expected from the semantic correctness result to be presented below). The example also shows that the analysis is indeed an over-approximation: although it is reported that the gene may occur at the top-level, it will never happen. □

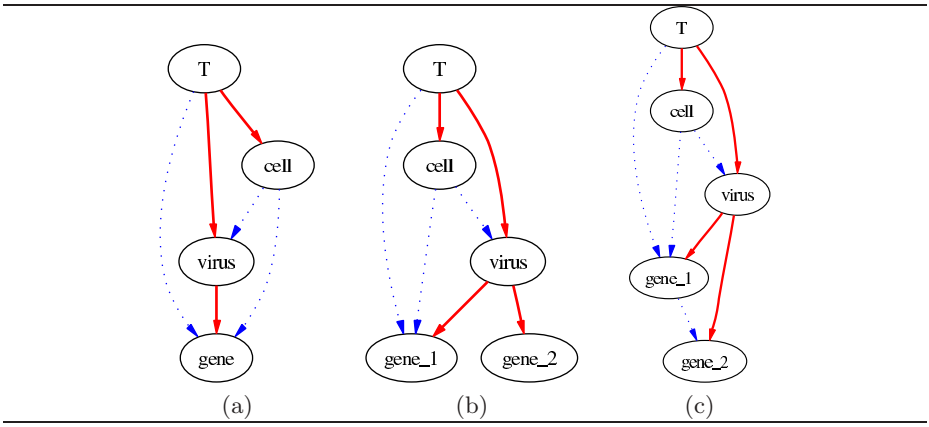

Fig. 2. Spatial analysis of the running examples P_{virus} (a) and P'_{virus} ((b) and (c)).

Example 6. To illustrate the importance of the comp relation consider the artificial variant of the virus process of Example 4. Figure 2 (b) gives a graphical representation of the $\mathcal{I}$ component of the analysis result and as expected we observe that the harmless gene does not change its position within the ambient hierarchy.

If we were to remove the tests on the compatibility relation in the closure condition of Table 8 then we would obtain a more imprecise result as illustrated on Figure 2 (c): it now seems that one of the genes may move into the other. The reason for this is, of course, that without the compatibility test the analysis does not observe that the two genes will never be present at the same time. □

Turning to the correctness of the analysis we shall state that the analysis result is invariant under the structural congruence:

Lemma 2. *If* $P \equiv Q$ *and* $\mathbf{C} \vdash P$ *then* $(\mathcal{I}, \mathcal{R}) \models^{\mu} P$ *if and only if* $(\mathcal{I}, \mathcal{R}) \models^{\mu} Q$.

To express the correctness of the analysis result under reduction we shall first introduce a new operation that expands the $\mathcal{I}$ component of the analysis to take the bindings of the variables into account as specified by the $\mathcal{R}$ component. Thus if $\mathsf{enter}\ \nu^{\ell} \in \mathcal{I}(\mu)$ then ν may be the canonical name of a variable and we shall construct the relation $\mathcal{I}@\mathcal{R}$ such that $\mathsf{enter}\ \nu'^{\ell} \in \mathcal{I}@\mathcal{R}(\mu)$ for all possible constants ν' that can be bound to ν, that is, for all $\nu' \in \langle\mathcal{R}\rangle(\nu)$. More generally, we define $\mathcal{I}@\mathcal{R}$ as follows:

$$\text{If } \lfloor M \rfloor^{\ell} \in \mathcal{I}(\mu),\ \nu \in \mathsf{fn}(\lfloor M \rfloor) \text{ and } \nu' \in \langle\mathcal{R}\rangle(\nu) \text{ then } \lfloor M \rfloor^{\ell}[\nu'/\nu] \in \mathcal{I}@\mathcal{R}(\mu).$$

We can now express that the analysis result is preserved under reduction in the following sense:

Lemma 3. *Assume* $\mathsf{PRG}_{\mathbf{C}}(P)$ *and* $\mathsf{comp}_{[\ ][\ ]}(P) \subseteq \mathsf{CP}_{\star}$*; if furthermore* $P \rightarrow Q$ *and* $(\mathcal{I}, \mathcal{R}) \models^{\top} P$ *then* $(\mathcal{I}@\mathcal{R}, \mathcal{R}) \models^{\top} Q$.

It is immediate to show that $\mathcal{I}@\mathcal{R} = (\mathcal{I}@\mathcal{R})@\mathcal{R}$ and hence we can state the overall correctness result as follows:

Theorem 1. *If* $\mathsf{PRG}_{\mathbf{C}}(P_\star)$, $(\mathcal{I},\mathcal{R}) \models^\top P_\star$ *and* $P_\star \rightarrow^* Q$ *then* $(\mathcal{I}@\mathcal{R},\mathcal{R}) \models^\top Q$.

5 Concluding Remarks

We have presented a spatial analysis for a version of BioAmbients with a general recursion construct that allows us to express mutual recursion as seems to be required in order to model biological systems. The analysis has been implemented using the Succinct Solver [10] and has subsequently been applied to a number of examples including three small examples from [13, 14] presented below. We conclude with a comparison with related work – indicating those techniques that are new to this paper.

Three Examples. The first example of Table 9 is a *membranal pore* allowing molecules to pass through a membrane. The example is specialised to the case of a single cell and two molecules and when executed the two membranes may enter and leave the cell any number of times and independently of one another. This is clearly captured by the analysis result of Figure 3. Also the analysis tells us that the cell will never enter one of the molecules and that the molecules will never enter one another; while this may be easy to see for a small example it may not be so obvious for a larger system.

The second example of Table 9 models a *single-substrate enzymatic process* and compared with the previous example its control structure is more complex in that it uses a double recursion and a number of names to control the interaction between the ambients. The analysis result depicted in Figure 3 exhibits the underlying spatial structure.

The final example of Table 9 models the formation and breakage of a *two-protein complex*. Initially the system consists of two molecules and the complex is formed by the merge operation. The breakage is initiated by a communication followed by a communication over a private name and finally the complex is separated into two molecules with the same structure as in the initial configuration. The rather complex control structure is reflected in the analysis result presented in Figure 3 showing that both molecules can be inside one another and that they both have the ability to reconstruct themselves.

Comparison with Related Work. The work presented in this paper is one of the first static analyses of calculi for modelling biological systems; to the best of our knowledge, the only preceeding work is that of [9] and the present work comprises a number of improvements and novelties.

One important difference is the way names are handled. In [9] we follow the traditional approach of control flow analysis and use an environment $\mathcal{R}$ that corresponds more closely to the auxiliary environment $\langle\mathcal{R}\rangle$ used here. Hence, in [9] we make an entry into $\mathcal{R}$ whenever a name is introduced (and in the case of a constant it is mapped to itself) and when we make an entry with a

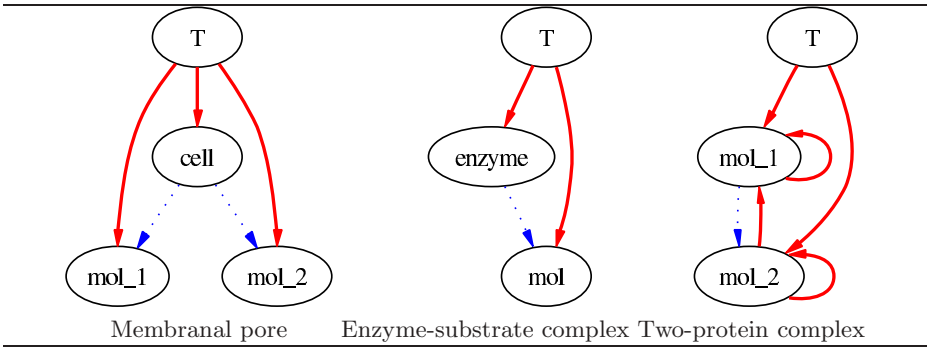

Fig. 3. Spatial analysis of examples in Table 9.

free name into $\mathcal{I}$ we make sure to make entries corresponding to all bindings of the free name as recorded in the environment (i.e. $\mathcal{R}$). While this leads to a rather natural formulation of the clauses and straighforward formulations of the semantic correctness result, the relations become overly large. Hence in the interest of obtaining more manageable implementations we have chosen *not* to add constants into environments and only to make *representative* entries into $\mathcal{I}$ that are then expanded "on the fly" during look-up. Essentially we are trading space for time which generally is a good strategy when using the Succinct Solver. To formulate the semantic correctness of the analysis we therefore need to make a similar expansion and this is achieved using $\mathcal{I}@\mathcal{R}$.

Another important difference is our treatment of recursion which is technically much more complex than the traditional treatment of replication (as in $!P$). The treatment of recursion in [9] was unsatisfactory in that the unfolding of the recursion construct was part of the transition relation rather than the congruence as in the present paper, and hence [9] misses some of the interactions correctly captured here. (To the best of our knowledge the analysis in [9] is correct with respect to the semantics.) For a correct treatment of this general way of unfolding recursion we have had to ensure that the body of the recursion is analysed in all contexts that may arise dynamically. While this may sound like just another component that could be added to the analysis (e.g. tracking occurrences of process identifiers in $\mathcal{I}$) it actually turns out to be important *not* to include this information into the analysis in order for the analysis to be semantically correct. Hence we have defined an operation G^{δ} for constructing a simple *regular grammar* deriving the possible contexts; it is essential for semantic correctness of the analysis that this information is not stored in components like $\mathcal{I}$ and $\mathcal{R}$ but rather computed "on the fly". This technique is likely to be useful for other calculi also outside the realm of biological systems.

Acknowledgements

The authors would like to thank Corrado Priami and Debora Schuch da Rosa for fruitful discussions.

References

1. M. Bugliesi, G. Castagna, and S. Crafa. Boxed Ambients. In *Theoretical Aspects in Computer Science (TACS 2001)*, volume 2215 of *Lecture Notes in Computer Science*, pages 37–63. Springer, 2001.
2. L. Cardelli. Brane calculi. 2003. Available from `http://www.luca.demon.co.uk`.
3. L. Cardelli and A. D. Gordon. Mobile Ambients. In *Foundations of Software Science and Computation Structures (FoSSaCS 1998)*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.
4. C.Bodei, P.Degano, C. Priami, and N. Zannone. An enhanced cfa for security policies. In *Proceedings of the Workshop on Issues on the Theory of Security (WITS'03) (co-located with ETAPS'03)*, 2003.
5. V. Danos and C. Laneve. Core formal molecular biology. In *European Symposium on Programming (ESOP03)*, volume 2618. Springer Lecture Notes in Computer Science, 2004.
6. F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2000)*, pages 352–364. ACM Press, 2000.
7. R. Milner. *Communicating and Mobile Systems: The pi-Calculus*. Cambridge University Press, 1999.
8. M. Nagasaki, S. Onami, S. Miyano, and Kitano H. Bio-calculus: Its concept and molecular interaction. *Genome Informatics*, 10:133–143, 1999.
9. F. Nielson, H. Riis Nielson, C.Priami, and D. Schuch da Rosa. Control Flow Analysis for BioAmbients. *Proceedings of BioConcur, to appear in ENTCS*, 2004.
10. F. Nielson, H. Riis Nielson, and H. Seidl. A succinct solver for ALFP. *Nordic Journal of Computing*, 9:335–372, 2002.
11. Hanne Riis Nielson, Flemming Nielson, and Mikael Buchholtz. Security for Mobility. In *Foundations of Security Analysis and Design II*, volume 2946. Springer Lecture Notes in Computer Science, 2004.
12. C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic passing-name calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.
13. A. Regev. *Computational system biology: A calculus for biomolecular knowledge*. PhD thesis, Tel Aviv University, 2003.
14. A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: An abstraction for biological compartments. *Theoretical Computer Science, to appear*, 2004.
15. A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Pacific Symposium of Biocomputing (PSB2001)*, pages 459–470, 2001.