Context Dependent Analysis of BioAmbients

Henrik Pilegaard, Flemming Nielson, Hanne Riis Nielson [1]

Informatics and Mathematical Modelling
Technical University of Denmark

Abstract

BioAmbients is a derivative of mobile ambients that has shown promise of describing interesting features of the behaviour of biological systems. The technical contribution of this paper is to extend the Flow Logic approach to static analysis with a couple of new techniques in order to give precise information about the behaviour of systems written in BioAmbients. Applying the development to a simple model of a cell releasing nutrients from food compunds we illustrate how the proposed analysis does indeed improve on previous efforts.

Key words: Static analysis, abstract interpretation, BioAmbients

1 Introduction

In *systems biology* the subject of study is the *systemic behaviour* of networks of molecular and cellular entities. For any group of such entities the networks that arise and their properties depend entirely on the interaction capabilities of the individual members; whether a cell is able to release nutrients from food compounds depends entirely on the interactions that may go on between the constituents of the food compounds and those of the cell.

Given the amount of information to be considered formal models are likely required in order to make qualified statements about the possible outcome. The BioAmbients calculus [17,18], described in Section 2, is a sibling to the ambient calculus [2] designed to model such biological systems. This calculus has been used successfully to model various systems [17,18,10], and in this context some biological properties, such as nutrient release, correspond to reachability properties of the model.

Program analysis technology [8] is generally able to provide interesting information about such properties. Previous efforts that approximate the *spatial structure* of BioAmbient processes [10,15] have shown that this is the case for models of biological systems also.

[1] Email:{hepi,nielson,riis}@imm.dtu.dk

In this paper we describe an analysis that incorporates novel ingredients that facilitate increased precision compared to the previous efforts. The analysis keeps track of the contents of ambients and their abilities for participating in various interactions and, thus, shares many characteristics with previous analyses of mobile ambients [9,10,15,14,4].

First, the control flow analysis, described in Section 4, is *context dependent*, i.e. it approximates the contexts of the ambients thereby being able to differentiate between interactions that are possible in different settings. We have found that recording the *two* enclosing ambients suffices for analysing fixed-structure cellular models with simple intruders [16]. This strikes a balance between the classical approaches that completely ignore context [9,10] and the more complex shape analysis of [13].

Second, auxiliary analyses, described in Section 3, capture important *causality* information for the capabilities; again this adds precision compared to [9,10] while avoiding some of the complexities of [13]. There are three ingredients in this:

- We follow [15] and keep track of the capabilities that may occur in parallel processes and hence may interact with one another; this ensures that parallelism and choice are analysed differently.
- For each capability we record the free variables that may occur in parallel processes and hence may be simultaneously live; thus only relevant names from the statically active name space are tracked when movement capabilities execute.
- For each capability and ambient we account for the free variables that may be in scope; this ensures that only relevant new names in the statically active name space are tracked when communication capabilities execute.

The analysis has been implemented using the Succinct Solver [11,12]; hence the figures reported in this paper have all been automatically produced.

2 BioAmbients

The BioAmbients [18,17,1] preserve the notion of ambients as bounded mobile sites of activity from Mobile Ambients [2], thereby allowing the biological concept of compartments to be modelled in an intuitive manner. Contrary to mobile ambients, however, bioambients are cast as nameless entities - the roles of which *may* be indicated by annotations. Both communication and ambient interaction are facilitated by having capability/co-capability pairs react with each other as in [5,14]. As a consequence all reactions are synchronous; the process exposing the capability and the process exposing the corresponding co-capability must agree on a reaction for it to happen. Such an agreement can be reached only if the two parties expose compatible capabilities and share the same name.

The set of control structures extends those traditionally studied for Mobile

$$
\begin{array}{rcll}
P & ::= & (n)P & \text{binding box for the constant } n \\
 & \mid & [P]^{\mu} & \text{ambient } P \text{ with the role } \mu \\
 & \mid & P \mid P' & \text{parallel processes} \\
 & \mid & \sum_{i\in I} M_i^{\ell_i}.P_i & \text{labelled non-deterministic external choice} \\
 & \mid & \mathsf{rec}\, X.\, P & \text{recursive process } (X = P) \\
 & \mid & X & \text{process identifier} \\
M & ::= & \mathsf{enter}\ n \mid \mathsf{accept}\ n & \text{enter movement} \\
 & \mid & \mathsf{exit}\ n \mid \mathsf{expel}\ n & \text{exit movement} \\
 & \mid & \mathsf{merge}{-}\ n \mid \mathsf{merge}{+}\ n & \text{merge movement} \\
 & \mid & n!\{m\} \mid n?\{p\} & \text{local communication binding the variable } p \\
 & \mid & n\_!\{m\} \mid n\hat{}?\{p\} & \text{parent to child communication binding the variable } p \\
 & \mid & n\hat{}!\{m\} \mid n\_?\{p\} & \text{child to parent communication binding the variable } p \\
 & \mid & n\#!\{m\} \mid n\#?\{p\} & \text{sibling communication binding the variable } p
\end{array}
$$

Fig. 1. Syntax of BioAmbients.

$$
\frac{\mathbf{C} \vdash P}{\mathbf{C} \vdash (n)P} \text{ if } \lfloor n \rfloor \in \mathbf{C}
\qquad
\frac{\mathbf{C} \vdash P}{\mathbf{C} \vdash [P]^{\mu}} \text{ if } \mathsf{fpi}(P) = \emptyset
\qquad
\frac{\mathbf{C} \vdash_{\Gamma \cup \{X\}} P}{\mathbf{C} \vdash \mathsf{rec}\ X.P} \text{ if } X \in \mathsf{fpi}(P)
$$

$$
\frac{\mathbf{C} \vdash P_1 \quad \mathbf{C} \vdash P_2}{\mathbf{C} \vdash P_1 | P_2}
\qquad
\mathbf{C} \vdash X
\quad
\frac{\forall i \in I : \mathbf{C} \vdash P_i}{\mathbf{C} \vdash \sum_{i\in I} M_i^{\ell_i}.P_i} \text{ if } \forall i \in I : \lfloor \mathsf{bn}(M_i) \rfloor \cap \mathbf{C} = \emptyset
$$

Fig. 2. Well-formedness of processes $\mathbf{C} \vdash_{\Gamma} P$.

Ambients with non-deterministic (external) choice and a general recursion construct in the manner of CCS [6].

2.1 Syntax

The full syntax of BioAmbients is defined in Fig. 1, where we write $P \in$ **Proc** for *processes* and M for *capabilities*. We associate each ambient with a *role* $\mu \in$ **Role** and annotate the ambient constructs accordingly. Roles (or identities) have no semantic significance but are useful annotations when modelling actual biosystems and will also prove valuable when we define our analysis. Also we associate each capability with a *label* $\ell \in$ **Lab**; these have no semantic significance either but are useful as pointers into the process.

We write n, m, p for *names*. Names are subject to α-renaming and not sufficient for carrying the information recorded by the analysis. Therefore we associate each name n with a corresponding *canonical name* $\lfloor n \rfloor \in$ **Name** that is invariant under α-renaming. We shall make a distinction between names introduced by $(n)\ P$, which we consider to be *constants* ($\lfloor n \rfloor \in \mathbf{C}$), and names introduced by communication capabilities, e.g. the p in $n?\{p\}$, which we consider to be *variables* ($\lfloor p \rfloor \in \mathbf{V}$). We require the sets of constants and variables to be mutually disjoint so that we are dealing with a finite set $\mathbf{Name} = \mathbf{C} \uplus \mathbf{V}$ of canonical names; we write ν for elements of this set.

Programs are processes $P_\star$ that satisfy the predicate $\mathsf{PRG}_{\mathbf{C}}(P_\star)$ defined as the conjunction of the following conditions (explained further below):

- $P_\star$ has no free process identifiers: $\mathsf{fpi}(P_\star) = \emptyset$.
- $P_\star$ has free names only from $\mathbf{C}$: $\lfloor \mathsf{fn}(P_\star) \rfloor \subseteq \mathbf{C}$
- $P_\star$ is well-formed with respect to $\mathbf{C}$: $\mathbf{C} \vdash_\emptyset P_\star$.

Here we write $\mathsf{fn}(P)$ for the *free names* of P and $\mathsf{fpi}(P)$ for the *free process identifiers* of P. As for names we associate each process identifier X with a corresponding *canonical process identifier* $\lfloor X \rfloor$ that is invariant under α-renaming. For convenience the canonicalisation operation $\lfloor \cdot \rfloor$ is extended in a point-wise manner to capabilities, sets of names and sets of process identifiers.

The associated *well-formedness predicate* $\mathbf{C} \vdash P$, defined in Fig. 2, enforces the implicit typing requirements imposed by the division of **Name**; here we write $\mathsf{bn}(M)$ for the *bound names* of capability M, e.g. $\mathsf{bn}(n?\{p\}) = \{p\}$ whereas $\mathsf{bn}(n!\{m\}) = \{\}$. The predicate ensures that each process identifier is actually used recursively in the process that it defines, but never inside a sub-ambient as this has no biological interpretation; this simplifies the technical development.

We shall write $P[m/n]$ and $P[Q/X]$ for substitution of names and process identifiers respectively - both cases subject to α-renaming of bound names and process identifiers. We require that α-renaming be disciplined such that canonical identities are preserved when the syntactical representations changes.

Example 2.1 Our running example is the following program P_{eat}; as we shall explain later it models how food particles may either be ignored, digested, or secreted by the cell:

$$\begin{array}{l}
[\ \mathsf{rec}\, Z.\ \mathsf{expel}\ rj^{\ell_1}.\ Z \\
\quad \mid [\ \mathsf{rec}\, Y.\ \mathsf{enter}\ ac^{\ell_2}.\ Y + \mathsf{exit}\ rj^{\ell_3}.\ Y + rea\hat{\ }?\{rl\}^{\ell_4}.\ \mathsf{expel}\ rl^{\ell_5}.\ Y \\
\qquad \mid [\mathsf{exit}\ RL^{\ell_6}.\ \mathsf{0}\,]^{nutrient}\,]^{food} \\
\quad \mid [\ \mathsf{rec}\, S.\ \mathsf{accept}\ ac^{\ell_7}.\ S + \mathsf{expel}\ rj^{\ell_8}.\ S + rea\_!\{RL\}^{\ell_9}.\ S\,]^{cell}\,]^{system}
\end{array}$$

The well-formedness condition is obviously satisfied for $\mathbf{C} = \{\lfloor rj \rfloor, \lfloor ac \rfloor, \lfloor rea \rfloor, \lfloor RL \rfloor\}$.

2.2 Semantics

The semantics is a standard reaction semantics. The structural congruence relation, $\equiv$, is defined for processes in general. It is the least congruence induced by the axioms of Fig. 3. The reaction relation, $\rightarrow$, is defined only for programs. It is shown in Fig. 4. Note that the semantics of the recursion construct is given as a congruence.

In the sequel we shall write $P \longrightarrow^\star P' \rightarrow Q$ if $P \rightarrow P_1 \rightarrow \cdots \rightarrow P_k$ for some $P_1, \ldots, P_k$ satisfying $P_{k-1} = P'$ and $P_k = Q$.

Example 2.2 The semantics of the example program P_{eat} is illustrated in Fig. 5. The initial configuration in frame 1 is pointed out by the fat arrow and here the tree structure reflects a scenario where *cell* and *food* are siblings inside *system* and *nutrient* is a sub-ambient of *food*. In this configuration (ℓ_3, ℓ_1) can fire to move *food* out of *system* and obtain the stuck configuration of frame 2. Alternatively, (ℓ_2, ℓ_7) can fire to move *food* into

Alpha-renaming of bound names and process identifiers:

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

Reordering of parallel processes:

$$P \mid P' \equiv P' \mid P$$
$$(P \mid P') \mid P'' \equiv P \mid (P' \mid P'')$$
$$P \mid \mathsf{0} \equiv P$$

Scope rules for name bindings:

$$(n)\mathsf{0} \equiv \mathsf{0}$$
$$(n_1)(n_2)P \equiv (n_2)(n_1)P$$
$$(n)(P \mid P') \equiv ((n)P) \mid P' \quad \text{if } n \notin \mathsf{fn}(P')$$
$$(n)([P]^\mu) \equiv [(n)P]^\mu$$

Recursion:

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

Reordering of sum processes:

$$P_1 + \ldots + P_i + P_{i+1} + \ldots + P_n \equiv P_1 + \ldots + P_{i+1} + P_i + \ldots + P_n \qquad \text{if } n > 1$$

Fig. 3. Axioms for structural congruence $P \equiv Q$.

Movement of ambients:

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

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:

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

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

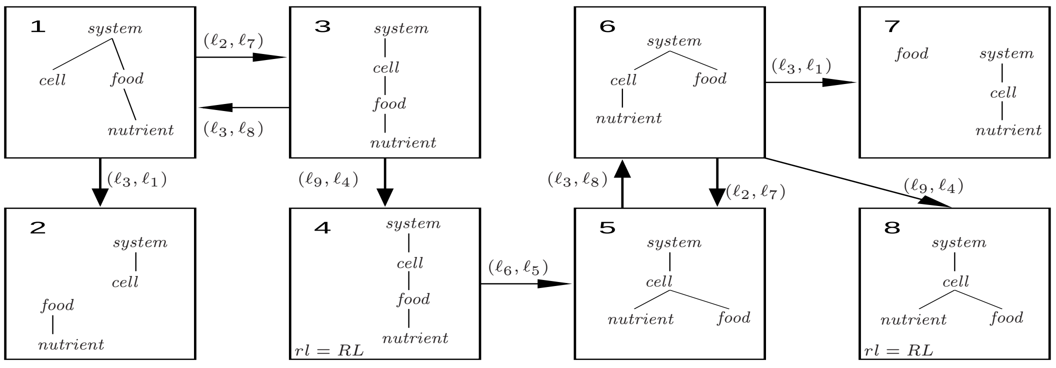
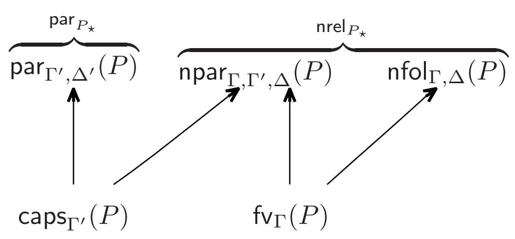


Fig. 5. The semantics of the example P_{eat}.

cell (frame 3). Then (ℓ_3, ℓ_8) can fire to move *food* back out of *cell* (frame 1 again), or (ℓ_1, ℓ_4) can fire to bind the variable *rl* to the constant *RL* (frame 4). After that only (ℓ_6, ℓ_5) can fire to move *nutrient* out of *food* (frame 5). Then only (ℓ_3, ℓ_8) can fire to move *food* out of *cell* (frame 6). From here (ℓ_2, ℓ_7) may fire to move *food* back into *cell* (frame 5). Alternatively,



$$\begin{tikzcd}
\mathsf{par}_{P_\star} & \arrow[r, phantom, "\mathsf{nrel}_{P_\star}"] & \\
\mathsf{par}_{\Gamma',\Delta'}(P) & \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P) & \mathsf{nfol}_{\Gamma,\Delta}(P) \\
\mathsf{caps}_{\Gamma'}(P) \arrow[u] \arrow[ur] & \mathsf{fv}_{\Gamma}(P) \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Fig. 6. Induced hierarchy of notions.

either (ℓ_3, ℓ_1) can move *food* out of *system* (frame 7) or (ℓ_1, ℓ_4) can fire to bind *rl* to *RL* (frame 8). Both of the resulting configurations are stuck.

3 Establishing Causality Information

The context dependent analysis to be defined in Section 4 gains much precision using the information from three auxiliary analyses.

We use a hierarchy of notions, illustrated in Fig. 6, to define these analyses. When defining the functions of this hierarchy (as well as the remainder of the analysis) we take the approach that is inherent in Flow Logic based program analysis [7]: In order to study properties of a process P we analyse the static snapshot given by the direct syntactical representation of P. However, we carefully define the analyses such that the properties are "$\supseteq$-preserved" under reduction and "$=$-preserved" under congruence and use this to establish the semantical soundness. This ensures that the information learned from studying P in $P_\star \longrightarrow^\star P \rightarrow Q$ is also valid for Q. Generally the subject of analysis is a top-level program $P_\star$. The approach outlined then ensures that the information learned from studying $P_\star$ is valid for any process P or Q related to $P_\star$ through a reduction path as shown above.

The indices Γ, Γ', Δ of the functions are then necessary because for the purposes of the analysis our focus no longer is on a given process P, but rather on all processes congruent to it; in particular, all processes obtained from P by unfolding one or more recursions. Because some of the sub-processes that we analyse *will* contain free process identifiers it is essential that only appropriately indexed functions are applied to sub-expressions.

3.1 Parallel Capabilities Analysis

Given a process P the aim of the parallel capabilities analysis is to identify the pairs of labelled capabilities that, from a syntactic point of view, may have the possibility of engaging in a reaction. As in [15,3] we take advantage of the fact that two capabilities only have the potential of engaging in reactions if they occur in parallel processes.

The analysis will ignore the actual names occurring in the capabilities (because even the canonical names are not preserved under reduction) and to

$$
\begin{array}{lcl}
\mathsf{caps}_{\Gamma'}((n)P) & = & \mathsf{caps}_{\Gamma'}(P)\\
\mathsf{caps}_{\Gamma'}([P]^{\mu}) & = & \mathsf{caps}_{\Gamma'}(P)\\
\mathsf{caps}_{\Gamma'}(P_1 \mid P_2) & = & \mathsf{caps}_{\Gamma'}(P_1) \cup \mathsf{caps}_{\Gamma'}(P_2)\\
\mathsf{caps}_{\Gamma'}(\sum_{i\in I} M_i^{\ell_i}.P_i) & = & \bigcup_{i\in I}(\{\lceil M_i\rceil^{\ell_i}\} \cup \mathsf{caps}_{\Gamma'}(P_i))\\
\mathsf{caps}_{\Gamma'}(\mathsf{rec}\ X.P) & = & \mathsf{caps}_{\Gamma'[X\mapsto\emptyset]}(P)\\
\mathsf{caps}_{\Gamma'}(X) & = & \Gamma'(X)
\end{array}
$$

Fig. 7. Capabilities $\mathsf{caps}_{\Gamma'}(P)$ of processes P.

$$
\begin{array}{lcl}
\mathsf{par}_{\Gamma',\Delta'}((n)P) & = & \mathsf{par}_{\Gamma',\Delta'}(P)\\
\mathsf{par}_{\Gamma',\Delta'}([P]^{\mu}) & = & \mathsf{par}_{\Gamma',\Delta'}(P)\\
\mathsf{par}_{\Gamma',\Delta'}(P_1 \mid P_2) & = & \mathsf{par}_{\Gamma',\Delta'}(P_1) \cup \mathsf{par}_{\Gamma',\Delta'}(P_2) \cup \mathsf{cross}_{\Gamma'}(P_1, P_2)\\
\mathsf{par}_{\Gamma',\Delta'}(\sum_{i\in I} M_i^{\ell_i}.P_i) & = & \bigcup_{i\in I} \mathsf{par}_{\Gamma',\Delta'}(P_i)\\
\mathsf{par}_{\Gamma',\Delta'}(\mathsf{rec}\ X.P) & = & \mathsf{par}_{\Gamma'[X\mapsto\mathsf{caps}_{\Gamma'[X\mapsto\emptyset]}(P)],\Delta'[X\mapsto\emptyset]}(P)\\
\mathsf{par}_{\Gamma',\Delta'}(X) & = & \Delta'(X)\\
\\
\mathsf{cross}_{\Gamma'}(P, P') & = & (\mathsf{caps}_{\Gamma'}(P) \times \mathsf{caps}_{\Gamma'}(P')) \cup (\mathsf{caps}_{\Gamma'}(P') \times \mathsf{caps}_{\Gamma'}(P))
\end{array}
$$

Fig. 8. Definition of $\mathsf{par}_{\Gamma',\Delta'}(P)$.

reflect this we shall introduce the notion of a *skeleton capability*: $\lceil M\rceil \in \mathbf{SCap}$ is obtained from M simply by replacing all names in M with the token "·".

To obtain the parallel capabilities information we extract the set of *labelled skeleton capabilities* occurring within a process. For this we use the function

$$\mathsf{caps}_{\Gamma'} : \mathbf{Proc} \to \mathcal{P}(\mathbf{SCap} \times \mathbf{Lab})$$

defined in Fig. 7. The index Γ' constitutes a mapping that associates each process identifier with a set of labelled skeleton capabilities.

The set of *parallel capabilities* can now be obtained using the function

$$\mathsf{par}_{\Gamma',\Delta'} : \mathbf{Proc} \to \mathcal{P}((\mathbf{SCap} \times \mathbf{Lab}) \times (\mathbf{SCap} \times \mathbf{Lab}))$$

defined in Fig. 8, which establishes a safe estimate of the set of pairs of labelled skeleton capabilities that may have a chance of being active at the same time. Here the index Γ' is as above and the index Δ' is a mapping that associates each process identifier with a set of pairs of labelled skeleton capabilities. We explain the interesting cases:

- For parallel composition the auxiliary operation cross is employed to record as interaction candidates all members of the direct product of the sets of capabilities from the two branches (obtained using caps).
- For non-deterministic choice no interaction candidates are recorded.
- For recursion constructs Γ' is updated to associate the process identifier used with the set of labelled capabilities of the process that it defines.

Lemma 3.1 (Correctness of the parallel capabilities analysis)

- *If* $P \equiv P'$ *then* $\mathsf{par}_{\Gamma',\Delta'}(P) \;=\; \mathsf{par}_{\Gamma',\Delta'}(P')$
- *If* $\mathsf{PRG}_{\mathbf{C}}(P)$ *and* $P \to^{\star} P'$ *then* $\mathsf{par}_{[\,],[\,]}(P) \supseteq \mathsf{par}_{[\,],[\,]}(P')$

$\lceil M\rceil^{\ell}$	$\{\lceil M'\rceil^{\ell'} \mid \mathsf{par}_{P_{\mathsf{eat}}}(\lceil M\rceil^{\ell}, \lceil M'\rceil^{\ell'})\}$
$\cdot\hat{}?\{\cdot\}^{\ell_4}$	$\mathsf{expel}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_7}, \cdot\_!\{\cdot\}^{\ell_9}, \mathsf{exit}\ \cdot^{\ell_6}, \mathsf{expel}\ \cdot^{\ell_8}$
$\mathsf{expel}\ \cdot^{\ell_5}$	$\mathsf{expel}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_7}, \cdot\_!\{\cdot\}^{\ell_9}, \mathsf{exit}\ \cdot^{\ell_6}, \mathsf{expel}\ \cdot^{\ell_8}$
$\mathsf{exit}\ \cdot^{\ell_3}$	$\mathsf{expel}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_7}, \cdot\_!\{\cdot\}^{\ell_9}, \mathsf{exit}\ \cdot^{\ell_6}, \mathsf{expel}\ \cdot^{\ell_8}$
$\mathsf{enter}\ \cdot^{\ell_2}$	$\mathsf{expel}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_7}, \cdot\_!\{\cdot\}^{\ell_9}, \mathsf{exit}\ \cdot^{\ell_6}, \mathsf{expel}\ \cdot^{\ell_8}$
$\mathsf{exit}\ \cdot^{\ell_6}$	$\mathsf{expel}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_7}, \cdot\_!\{\cdot\}^{\ell_9}, \cdot\hat{}?\{\cdot\}^{\ell_4}, \mathsf{expel}\ \cdot^{\ell_5}, \mathsf{exit}\ \cdot^{\ell_3}, \mathsf{enter}\ \cdot^{\ell_2}, \mathsf{expel}\ \cdot^{\ell_8}$
$\mathsf{expel}\ \cdot^{\ell_1}$	$\mathsf{exit}\ \cdot^{\ell_6}, \mathsf{enter}\ \cdot^{\ell_2}, \mathsf{exit}\ \cdot^{\ell_3}, \mathsf{expel}\ \cdot^{\ell_5}, \cdot\hat{}?\{\cdot\}^{\ell_4}, \mathsf{expel}\ \cdot^{\ell_8}, \mathsf{accept}\ \cdot^{\ell_7}, \cdot\_!\{\cdot\}^{\ell_9}$
$\mathsf{accept}\ \cdot^{\ell_7}$	$\mathsf{exit}\ \cdot^{\ell_6}, \mathsf{enter}\ \cdot^{\ell_2}, \mathsf{exit}\ \cdot^{\ell_3}, \mathsf{expel}\ \cdot^{\ell_5}, \cdot\hat{}?\{\cdot\}^{\ell_4}, \mathsf{expel}\ \cdot^{\ell_1}$
$\cdot\_!\{\cdot\}^{\ell_9}$	$\mathsf{exit}\ \cdot^{\ell_6}, \mathsf{enter}\ \cdot^{\ell_2}, \mathsf{exit}\ \cdot^{\ell_3}, \mathsf{expel}\ \cdot^{\ell_5}, \cdot\hat{}?\{\cdot\}^{\ell_4}, \mathsf{expel}\ \cdot^{\ell_1}$
$\mathsf{expel}\ \cdot^{\ell_8}$	$\mathsf{exit}\ \cdot^{\ell_6}, \mathsf{enter}\ \cdot^{\ell_2}, \mathsf{exit}\ \cdot^{\ell_3}, \mathsf{expel}\ \cdot^{\ell_5}, \cdot\hat{}?\{\cdot\}^{\ell_4}, \mathsf{expel}\ \cdot^{\ell_1}$

Fig. 9. The relation $\mathsf{par}_{P_{\mathsf{eat}}}$.

$$
\begin{array}{ll}
\mathsf{fv}_{\Gamma}((n)P) & = \mathsf{fv}_{\Gamma}(P) \\
\mathsf{fv}_{\Gamma}([P]^{\mu}) & = \mathsf{fv}_{\Gamma}(P) \\
\mathsf{fv}_{\Gamma}(P_1 \mid P_2) & = \mathsf{fv}_{\Gamma}(P_1) \cup \mathsf{fv}_{\Gamma}(P_2) \\
\mathsf{fv}_{\Gamma}(\sum_{i\in I} M_i^{\ell_i}.P_i) & = \bigcup_{i\in I}((\lfloor \mathsf{fn}(M_i^{\ell_i})\rfloor \setminus \mathbf{C}) \cup (\lfloor \mathsf{fv}_{\Gamma}(P_i)\rfloor \setminus \lfloor \mathsf{bn}(M_i^{\ell_i})\rfloor)) \\
\mathsf{fv}_{\Gamma}(\mathsf{rec}\ X.P) & = \mathsf{fv}_{\Gamma[X\mapsto\emptyset]}(P) \\
\mathsf{fv}_{\Gamma}(X) & = \Gamma(X)
\end{array}
$$

Fig. 10. Free variables $\mathsf{fv}_{\Gamma}(P)$ of process P.

As we always analyse a program $P_\star$ we will define the abbreviation $\mathsf{par}_{P_\star} \triangleq \mathsf{par}_{[],[]}(P_\star)$, to denote the result of the parallel capabilities analysis. The result then shows the required "$\supseteq$-preservation" property in the sense that if $\mathsf{PRG}_{\mathbf{C}}(P_\star)$ and $P_\star \longrightarrow^\star P$ then $\mathsf{par}_{P_\star} \supseteq \mathsf{par}_{[],[]}(P)$ meaning that the relation $\mathsf{par}_{P_\star}$ correctly contains all potential redex-pairs occurring in P.

Example 3.2 For the running example P_{eat} we obtain the relation $\mathsf{par}_{P_{\mathsf{eat}}}$ shown in Fig. 9. As must be expected, this constitutes a crude over-approximation of the interactions that may take place.

3.2 Relevant Variables and Scope Analyses

Given a process, P, the aim of the relevant variables and scope analyses is to establish a safe estimate of the variables whose binding contexts must be updated when movement capabilities are executed and of the variable bindings that must be propagated across the boundaries of ambients.

Rather than the simple free names function fn employed by the semantics to determine when to apply the scope rules we shall use an appropriately indexed function fv_{Γ}, defined in Fig. 10, that collects free (canonical) variables rather than free names[2]. The index Γ is a mapping that to each process identifier associates a set of (free) canonical variables.

[2] The latter is essential for Lemmas 5 and 6 to hold; actually it will be the case that $\mathsf{fv}_{[]}(P) = \lfloor \mathsf{fn}(P)\rfloor \cap \mathbf{V}$.

$$\begin{array}{llll}
\mathsf{npar}_{\Gamma,\Gamma',\Delta}(X) & = \Delta(X) & \mathsf{npar}_{\Gamma,\Gamma',\Delta}(\sum_{i\in I} M_i^{\ell_i}.P_i) & = \bigcup_{i\in I} \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P_i) \\
\mathsf{npar}_{\Gamma,\Gamma',\Delta}((n)P) & = \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P) & \mathsf{npar}_{\Gamma,\Gamma',\Delta}([P]^{\mu}) & = \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P)
\end{array}$$

$$\begin{array}{rcl}
\mathsf{npar}_{\Gamma,\Gamma',\Delta}(\mathsf{rec}\ X.P) & = & \mathsf{npar}_{\Gamma[X\mapsto \mathsf{fv}_{\Gamma[X\mapsto\emptyset]}(P)],\Gamma'[X\mapsto \mathsf{caps}_{\Gamma'[X\mapsto\emptyset]}(P)],\Delta[X\mapsto\emptyset]}(P) \\
\mathsf{npar}_{\Gamma,\Gamma',\Delta}(P_1 \mid P_2) & = & \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P_1) \cup \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P_2) \cup \\
 & & \{(\lceil M\rceil^{\ell},\nu) \mid (\lceil M\rceil^{\ell} \in \mathsf{caps}_{\Gamma'}(P_1) \wedge \nu \in \lfloor \mathsf{fv}_{\Gamma}(P_2)\rfloor)\vee \\
 & & \qquad (\lceil M\rceil^{\ell} \in \mathsf{caps}_{\Gamma'}(P_2) \wedge \nu \in \lfloor \mathsf{fv}_{\Gamma}(P_1)\rfloor)\}
\end{array}$$

Fig. 11. Definition of $\mathsf{npar}_{\Gamma,\Gamma',\Delta}(P)$.

We then define the required information as the combination of two analyses - one dealing with variables occurring in branches parallel to capabilities and one dealing with variables whose scope extends over capabilities or ambients.

The relevant variables analysis is motivated by the observation that variables bound in a branch parallel to the execution of a particular movement capability M remain active when M executes as e.g. p does when $\mathsf{enter}\ m^{\ell'}$ executes in $[n\#?\{p\}^{\ell}.Q \mid \mathsf{enter}\ m^{\ell'}.P]^{\mu}$. This requires us to extract information that relates each of the occurring labelled skeleton capabilities to the set of free variables occurring in *parallel* with it. This is done using the function

$$\mathsf{npar}_{\Gamma,\Gamma',\Delta} : \mathbf{Proc} \to \mathcal{P}((\mathbf{SCap} \times \mathbf{Lab}) \times \mathbf{V})$$

that extracts the required information as shown in Fig. 11. Here Γ is the index used in fv, Γ' is that used in caps, and Δ is a mapping that associates each process identifier with a set of pairs of labelled skeleton capabilities and canonical variables. We explain the interesting cases:

- For parallel composition the union of two direct products is recorded; the set of all capabilities from the first sub-process (obtained using caps) paired with the set of all free canonical variables from the second sub-process (obtained using fv) and vice versa.
- Nothing is recorded for non-deterministic choice.
- For recursion constructs Γ' and Γ are updated in order to correctly associate the used process identifier with, respectively, the set of labelled capabilities and the set of free variables of the process that it defines.

Lemma 3.3 (Correctness of the relevant variables analysis)

- *If* $P \equiv P'$ *then* $\mathsf{npar}_{\Gamma,\Gamma',\Delta}(P) = \mathsf{npar}_{\Gamma,\Gamma',\Delta}(P')$
- *If* $\mathsf{PRG}_{\mathbf{C}}(P)$ *and* $P \to^{\star} P'$ *then* $\mathsf{npar}_{[\,],[\,],[\,]}(P) \supseteq \mathsf{npar}_{[\,],[\,],[\,]}(P')$

The scope analysis is motivated by the observation that variables, whose scope extend over a movement capability, may be required for the correct execution of the continuation remaining once the movement capability has been executed as e.g. p is when $\mathsf{enter}\ m^{\ell'}$ executes in $[n\#?\{p\}^{\ell}.\mathsf{enter}\ m^{\ell'}.\mathsf{enter}\ p^{\ell''}.P]^{\mu}$. As the variables are of course only required if they are actually used we can restrict our attention to the set of variables that occur free in the continuation itself. So to account for this we extract information that relates the skeleton

$$
\begin{array}{ll}
\mathsf{nfol}_{\Gamma,\Delta}((n)P) & = \mathsf{nfol}_{\Gamma,\Delta}(P) \\
\mathsf{nfol}_{\Gamma,\Delta}([P]^{\mu}) & = (\{\mu\} \times \lfloor \mathsf{fv}_{\Gamma}(P) \rfloor) \cup \mathsf{nfol}_{\Gamma,\Delta}(P) \\
\mathsf{nfol}_{\Gamma,\Delta}(P_1|P_2) & = \mathsf{nfol}_{\Gamma,\Delta}(P_1) \cup \mathsf{nfol}_{\Gamma,\Delta}(P_2) \\
\mathsf{nfol}_{\Gamma,\Delta}(\sum_{i\in I} M_i^{\ell_i}.P_i) & = \bigcup_{i\in I}((\{\lceil M_i \rceil^{\ell_i}\} \times \lfloor \mathsf{fv}_{\Gamma}(P_i) \rfloor) \cup \mathsf{nfol}_{\Gamma,\Delta}(P_i)) \\
\mathsf{nfol}_{\Gamma,\Delta}(\mathsf{rec}\ X.P) & = \mathsf{nfol}_{\Gamma[X\mapsto \mathsf{fv}_{\Gamma[X\mapsto\emptyset]}(P)],\Delta[X\mapsto\emptyset]}(P) \\
\mathsf{nfol}_{\Gamma,\Delta}(X) & = \Delta(X)
\end{array}
$$

Fig. 12. Definition of $\mathsf{nfol}_{\Gamma,\Delta}(P)$.

of each occurring labelled capability to the set of free variables that occur sequentially after it.

If the occurrence of such a free variable is nested inside an ambient then the ambient is in the scope of the variable. In order to correctly treat variable bindings it is necessary to take these hierarchical scopes into account. Fortunately, as variables have static scope, this can be done simply by extracting information that relates each ambient to the set of variables in scope. We do all of this using the function

$$\mathsf{nfol}_{\Gamma,\Delta} : \mathbf{Proc} \to \mathcal{P}(((\mathbf{SCap} \times \mathbf{Lab}) \cup \mathbf{Role}) \times \mathbf{V})$$

that given a process extracts the required information as shown in Fig. 12. Here Γ is the index used in fv and Δ is a mapping that associates each process identifier with a set of pairs of either labelled skeleton capabilities or ambient roles and canonical names.

Lemma 3.4 (Correctness of the scope analysis)

- *If* $P \equiv P'$ *then* $\mathsf{nfol}_{\Gamma,\Delta}(P) = \mathsf{nfol}_{\Gamma,\Delta}(P')$
- *If* $\mathsf{PRG}_{\mathbf{C}}(P)$ *and* $P \to^{\star} P'$ *then* $\mathsf{nfol}_{[\,],[\,]}(P) \supseteq \mathsf{nfol}_{[\,],[\,]}(P')$

As we always analyse a program $P_\star$ we will define the abbreviation $\mathsf{nrel}_{P_\star} \triangleq \mathsf{npar}_{[],[],[]}(P_\star) \cup \mathsf{nfol}_{[],[]}(P_\star)$, to denote the combined result of the relevant names and scope analyses. The two lemmas then show the "$\supseteq$-preservation" property for the relevant variables and scope analysis as a whole. Thus, if $\mathsf{PRG}_{\mathbf{C}}(P_\star)$ and $P_\star \longrightarrow^{\star} P$ then $\mathsf{nrel}_{P_\star} \supseteq \mathsf{npar}_{[],[],[]}(P) \cup \mathsf{nfol}_{[],[]}(P)$ meaning that the relation $\mathsf{nrel}_{P_\star}$

(i) correctly relates each capability occurring in P to all names in P that may be relevant to it, i.e. names that do not disappear because of non-deterministic choice, and

(ii) correctly relates each ambient occurring in P to all names in scope.

Example 3.5 For the running example P_{eat} we get

$$\mathsf{nrel}_{P_{\mathsf{eat}}} = \{(\cdot\hat{}?\{\cdot\}^{\ell_4}, \lfloor rl \rfloor)\}$$

which in fact indicates that no names are relevant to any ambient movement.

$$\begin{array}{ll}
(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} (n)P & \text{iff } (\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P \\
(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} [P]^{\mu_c} & \text{iff } \mu_c \in \mathcal{I}(\mu_{gp},\mu_p,\mu) \wedge (\mathcal{I},\mathcal{R}) \models^{\mu_p\mu\mu_c} P \wedge \mathsf{closure}_{\mathsf{scp}} \\
(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P \mid P' & \text{iff } (\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P \wedge (\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P' \\
(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} \sum_{i\in I} M_i^{\ell_i}.P_i & \text{iff } \forall i \in I : (\lfloor M_i \rfloor^{\ell_i} \in \mathcal{I}(\mu_{gp},\mu_p,\mu) \wedge (\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P_i \wedge \mathsf{closure}_{\lceil M \rceil_i}) \\
(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} \mathsf{rec}\, X.\, P & \text{iff } (\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P \\
(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} X & \text{iff } \mathsf{true}
\end{array}$$

Fig. 13. Analysis of processes: $(\mathcal{I},\mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P$.

4 Control Flow Analysis

Given the auxiliary analysis information we are now ready to define the actual context dependent spatial analysis. The aim of this analysis is to determine which ambients may turn up inside which other ambients during the execution of a program $P_\star$.

In order to specify the required information the analysis establishes an over-approximation of the potential ambient containments. Two types of information are extracted:

- A localised approximation of the contents of ambients:
$$\mathcal{I} \subseteq \mathbf{Role} \times \mathbf{Role} \times \mathbf{Role} \times (\mathbf{Role} \cup (\mathbf{Cap} \times \mathbf{Lab}))$$
Here $u \in \mathcal{I}(\mu_{gp},\mu_p,\mu)$ (standing for $(\mu_{gp},\mu_p,\mu,u) \in \mathcal{I}$) means that μ may contain u in the context of $\mu_{gp}\mu_p$.
- A localised approximation of the relevant name bindings:
$$\mathcal{R} \subseteq \mathbf{Role} \times \mathbf{Role} \times \mathbf{Role} \times \mathbf{V} \times \mathbf{C}$$
Here $\nu' \in \mathcal{R}(\mu_{gp},\mu_p,\mu,\nu)$ (i.e. $(\mu_{gp},\mu_p,\mu,\nu,\nu') \in \mathcal{R}$) means that the canonical variable ν may bind the canonical constant ν' in the context $\mu_{gp}\mu_p\mu$.

4.1 Judgements

The judgements of the analysis take the form

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

where $\mu_{gp}\mu_p\mu$ denotes the initial context in which the syntactic construct P occurs. These judgements express that when a sub-process P of $P_\star$ is enclosed within an ambient identified by $\mu \in \mathbf{Role}$ in the context $\mu_{gp}\mu_p \in \mathbf{Role} \times \mathbf{Role}$ then $\mathcal{I}$ and $\mathcal{R}$ correctly capture the behaviour of P.

The Flow Logic specification of the analysis is given in Fig. 13. The specification refers to Figs. 14 and 16 for the specification of closure conditions $\mathsf{closure}_{\lceil M \rceil}$ that mimics the semantics by modelling, within the limited precision of the analysis, the actual semantic prerequisites and consequences of communication and ambient movement respectively, and finally to Fig. 17 for the closure condition $\mathsf{closure}_{\mathsf{scp}}$ that propagates variable scopes into ambients as appropriate. The clauses of the analysis specify a simple syntax directed traversal of processes. Below we comment on these clauses.

The clause for name binding $(n)P$ simply ensures that the sub-expression P is traversed while ignoring n. Constants matter for the analysis only when bound and are therefore handled only in the closure conditions.

The clause for ambient expressions $[P]^{\mu_c}$ checks that whenever a child ambient μ_c is encountered during the traversal the $\mathcal{I}$ component records the hierarchy position of μ_c appropriately ($\mu_c \in \mathcal{I}(\mu_{gp}, \mu_p, \mu)$). The clause also ensures that sub-expressions of μ_c are traversed in the updated context $\mu_p\mu\mu_c$ ($(\mathcal{I}, \mathcal{R}) \models^{\mu_p\mu\mu_c} P$). Finally, in order to take appropriate care of the variables in scope, the clause introduces the closure condition $\mathsf{closure}_{\mathsf{scp}}$.

The clauses for parallel $P_1|P_2$ composition ensures the traversal of the sub-expressions P_1 and P_2. Issues regarding scopes etc. matter only for the closure conditions and are handled there using information from $\mathsf{par}_{P_\star}$ and $\mathsf{nrel}_{P_\star}$.

The clause for non-deterministic choice $\sum_{i \in I} M_i^{\ell_i}.P_i$ ensures that the hierarchy position of all labelled capabilities $M_i^{\ell_i}$ encountered during traversal are recorded with a canonical entry in $\mathcal{I}$ ($\lfloor M_i \rfloor^{\ell_i} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu)$). Also, the clause introduces a closure condition $\mathsf{closure}_{\lceil M_i \rceil}$ corresponding to the skeleton of each capability M_i. Finally, traversal of all sub-expressions P_i is ensured.

The clause for recursion constructs $\mathsf{rec}\, X.\, P$ only ensures traversal of the subexpression P. Due to the well-formedness condition it is safe to assume that it suffices to analyse P in the context where it is first defined. For the same reason we can ignore process identifiers X.

4.2 Closure Conditions

The closure conditions (Figs. 14, 16, 17) mimic the semantics to ensure that $\mathcal{I}$ and $\mathcal{R}$ reflect the dynamic behaviour of the analysed program, $P_\star$. There is one closure condition for each type of capability, but the ones corresponding to co-capabilities are trivial and included only for completeness.

We shall define and explain the closure conditions in the following. Before we go on to do this, however, another useful relation

$$\langle \mathcal{R} \rangle \subseteq \mathbf{Role} \times \mathbf{Role} \times \mathbf{Role} \times \mathbf{Name} \times \mathbf{C}$$

needs to be defined:

$$\begin{aligned} &\forall \mu_{gp}, \mu_p, \mu : \mathcal{R}(\mu_{gp}, \mu_p, \mu) \subseteq \langle \mathcal{R} \rangle(\mu_{gp}, \mu_p, \mu) \land \\ &\qquad \forall \mu_{gp}, \mu_p, \mu, \mu_c, \nu : (\mu_{gp}, \mu_p, \mu, \mu_c) \in \mathcal{I} \land \nu \in \mathbf{C} \Rightarrow \langle \mathcal{R} \rangle(\mu_p, \mu, \mu_c, \nu, \nu) \end{aligned}$$

It binds constants to themselves in *all* realisable contexts but binds variables to values only in the contexts where the bindings may actually be in scope.

The closure conditions for communication govern the dynamics of variable bindings. These conditions are very similar; hence we shall explain only the one for local communication and define the the others in Fig. 14

The essence of the closure condition is shown in Fig. 15a where the shape of the tree corresponds directly to the requirements that have to be matched by the preconditions of the clause. The dotted arrow, which signifies that all

$$\begin{array}{l}
\mathsf{closure}_{\cdot!\{\cdot\}} = \forall \mu_{gp}, \mu_p, \mu, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
\quad \nu_1!\{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \nu_2?\{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \mathsf{par}_{P_\star}(\cdot!\{\cdot\}^{\ell_1}, \cdot?\{\cdot\}^{\ell_2}) \wedge \\
\qquad \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_2) \neq \emptyset \Rightarrow \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_m) \subseteq \mathcal{R}(\mu_{gp}, \mu_p, \mu, \nu_p) \\
\mathsf{closure}_{\cdot?\{\cdot\}} = \mathsf{true} \\
\mathsf{closure}_{\cdot\_!\{\cdot\}} = \forall \mu_{gp}, \mu_p, \mu, \mu_1, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
\quad \nu_1\_!\{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \nu_2\hat{}?\{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_p, \mu, \mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \\
\qquad \mathsf{par}_{P_\star}(\cdot\_!\{\cdot\}^{\ell_1}, \cdot\hat{}?\{\cdot\}^{\ell_2}) \wedge \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_2) \neq \emptyset \Rightarrow \\
\hfill \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_m) \subseteq \mathcal{R}(\mu_p, \mu, \mu_1, \nu_p) \\
\mathsf{closure}_{\cdot\hat{}?\{\cdot\}} = \mathsf{true} \\
\mathsf{closure}_{\cdot\hat{}!\{\cdot\}} = \forall \mu_{gp}, \mu_p, \mu, \mu_1, \nu_m, \nu_p, \nu_1, \nu_2, \ell_1, \ell_2 : \\
\quad \nu_1\hat{}!\{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu_p, \mu, \mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \nu_2\_?\{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \\
\qquad \mathsf{par}_{P_\star}(\cdot\hat{}!\{\cdot\}^{\ell_1}, \cdot\_?\{\cdot\}^{\ell_2}) \wedge \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_2) \cap \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_1) \neq \emptyset \Rightarrow \\
\hfill \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_m) \subseteq \mathcal{R}(\mu_{gp}, \mu_p, \mu, \nu_p) \\
\mathsf{closure}_{\cdot\_?\{\cdot\}} = \mathsf{true} \\
\mathsf{closure}_{\cdot\#!\{\cdot\}} = \forall \mu_{gp}, \mu_p, \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_p, \mu, \mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \\
\quad\; \nu_2\#?\{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mu_2 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \mathsf{par}_{P_\star}(\cdot\#!\{\cdot\}^{\ell_1}, \cdot\#?\{\cdot\}^{\ell_2}) \wedge \\
\qquad \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_2, \nu_2) \neq \emptyset \Rightarrow \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_m) \subseteq \mathcal{R}(\mu_p, \mu, \mu_2, \nu_p) \\
\mathsf{closure}_{\cdot\#?\{\cdot\}} = \mathsf{true}
\end{array}$$

Fig. 14. Closure conditions for communication capabilities.

names that may possibly be bound to ν_m may also bind to ν_p, corresponds to the conclusion of the closure condition. In the following we describe this modelling in more detail.

The preconditions make the following requirements:

- The output and input capabilities (redexes) must be in the same location:

$$\nu_1!\{\nu_m\}^{\ell_1} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \nu_2?\{\nu_p\}^{\ell_2} \in \mathcal{I}(\mu_{gp}, \mu_p, \mu)$$

- The redexes must have the possibility of occurring in parallel:

$$\mathsf{par}_{P_\star}(\cdot!\{\cdot\}^{\ell_1}, \cdot?\{\cdot\}^{\ell_2})$$

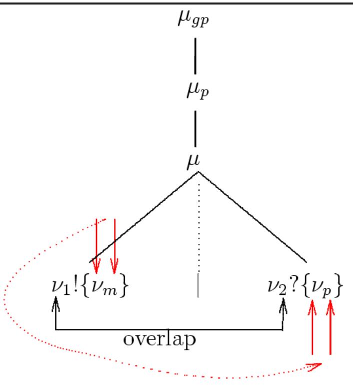
- The redexes must agree on a shared name:

$$\langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_2) \neq \emptyset$$

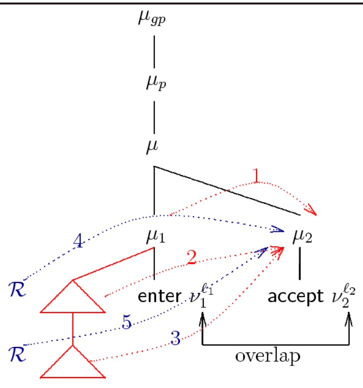
  The capabilities may reference the shared name directly, by using a constant, or indirectly, by using a variable. We allow both by expressing the condition as a requirement of non-empty intersection in the $\langle\mathcal{R}\rangle$ relation.

With the preconditions satisfied the conclusion of the clause expresses the consequence of completing the communication. Any value denoted by the output name might be communicated, which the rule reflects by updating $\mathcal{R}$ to record that the input variable in context $\mu_{gp}\mu_p\mu$ may be bound to any of the values possibly bound to the output name in context $\mu_{gp}\mu_p\mu$:

$$\langle\mathcal{R}\rangle(\mu_{gp}, \mu_p, \mu, \nu_m) \subseteq \mathcal{R}(\mu_{gp}, \mu_p, \mu, \nu_p)$$



a. Local Communication.



b. Enter Movement.

Fig. 15. Capability actions.

Because a name *sent* over a channel may be referenced through either a constant or a variable we obtain the set of candidates from the $\langle\mathcal{R}\rangle$ relation. A name *received*, however, is always referenced through a variable and we therefore update the $\mathcal{R}$ relation.

These closure conditions are our only means of populating the $\mathcal{R}$ relation; this ensures that only bindings of variables to constants are tracked dynamically by $\mathcal{R}$.

The closure conditions for movement govern the dynamics of the containment hierarchy. We shall describe only the closure condition of the *enter* kind and defer the definition of the others to Fig. 16. Again, the essence of the closure condition is shown in Fig. 15b.

The preconditions make the following requirements:

- The enter and accept capabilities (redexes) must be in sibling ambients:
$$\mathsf{enter}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu_p,\mu,\mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_{gp},\mu_p,\mu)\wedge$$
$$\mathsf{accept}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_p,\mu,\mu_2) \wedge \mu_2 \in \mathcal{I}(\mu_{gp},\mu_p,\mu)$$
- The redexes must have the possibility of occurring in parallel:
$$\mathsf{par}_{P_\star}(\mathsf{enter}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_2})$$
- The redexes must agree on a shared name:
$$\langle\mathcal{R}\rangle(\mu_p,\mu,\mu_1,\nu_1) \cap \langle\mathcal{R}\rangle(\mu_p,\mu,\mu_2,\nu_2) \neq \emptyset$$
Similar to before, we impose the condition as non-empty intersection in the $\langle\mathcal{R}\rangle$ relation.

With the preconditions satisfied the conclusions of the clause express the consequences of completing the movement:

- The $\mathcal{I}$ relation must be updated to reflect that μ_1 may be inside μ_2 and that the contents of μ_1 is also inside μ_1 in this new context of μ_2:
$$\mu_1 \in \mathcal{I}(\mu_p,\mu,\mu_2) \wedge \mathcal{I}(\mu_p,\mu,\mu_1) \subseteq \mathcal{I}(\mu,\mu_2,\mu_1) \wedge \mathcal{I}(\mu,\mu_1) \subseteq \mathcal{I}(\mu_2,\mu_1)$$
Note that we have three contributions depending on whether we consider

$$
\begin{array}{l}
\mathsf{closure}_{\mathsf{enter}}\ \cdot = \forall \mu_{gp}, \mu_p, \mu, \mu_1, \mu_2, \nu_1, \nu_2, \ell_1, \ell_2 : \\
\quad \mathsf{enter}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu_p, \mu, \mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \mathsf{accept}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mu_2 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \\
\quad \mathsf{par}_{P_\star}(\mathsf{enter}\ \cdot^{\ell_1}, \mathsf{accept}\ \cdot^{\ell_2}) \wedge \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_2, \nu_2) \neq \emptyset \Rightarrow \\
\qquad \mu_1 \in \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mathcal{I}(\mu_p, \mu, \mu_1) \subseteq \mathcal{I}(\mu, \mu_2, \mu_1) \wedge \mathcal{I}(\mu, \mu_1) \subseteq \mathcal{I}(\mu_2, \mu_1) \wedge \\
\qquad (\forall \nu : \mathsf{nrel}_{P_\star}(\mathsf{enter}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu_p, \mu, \mu_1, \nu) \subseteq \mathcal{R}(\mu, \mu_2, \mu_1, \nu)) \wedge \\
\qquad (\forall \nu, \mu' : \mathsf{nrel}_{P_\star}(\mathsf{enter}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu, \mu_1, \mu', \nu) \subseteq \mathcal{R}(\mu_2, \mu_1, \mu', \nu)) \\
\mathsf{closure}_{\mathsf{accept}}\ \cdot = \mathsf{true} \\
\mathsf{closure}_{\mathsf{exit}}\ \cdot = \forall \mu_{gp}, \mu_p, \mu, \mu_1, \mu_2, \nu_1, \nu_2, \ell_1, \ell_2 : \\
\quad \mathsf{exit}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu, \mu_2, \mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mathsf{expel}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mu_2 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \\
\quad \mathsf{par}_{P_\star}(\mathsf{exit}\ \cdot^{\ell_1}, \mathsf{expel}\ \cdot^{\ell_2}) \wedge \langle\mathcal{R}\rangle(\mu, \mu_2, \mu_1, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_2, \nu_2) \neq \emptyset \Rightarrow \\
\qquad \mu_1 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \mathcal{I}(\mu, \mu_2, \mu_1) \subseteq \mathcal{I}(\mu_p, \mu, \mu_1) \wedge \mathcal{I}(\mu_2, \mu_1) \subseteq \mathcal{I}(\mu, \mu_1) \wedge \\
\qquad (\forall \nu : \mathsf{nrel}_{P_\star}(\mathsf{exit}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu, \mu_2, \mu_1, \nu) \subseteq \mathcal{R}(\mu_p, \mu, \mu_1, \nu)) \wedge \\
\qquad (\forall \nu, \mu' : \mathsf{nrel}_{P_\star}(\mathsf{exit}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu_2, \mu_1, \mu', \nu) \subseteq \mathcal{R}(\mu, \mu_1, \mu', \nu)) \\
\mathsf{closure}_{\mathsf{expel}}\ \cdot = \mathsf{true} \\
\mathsf{closure}_{\mathsf{merge-}}\ \cdot = \forall \mu_{gp}, \mu_p, \mu, \mu_1, \mu_2, \nu_1, \nu_2, \ell_1, \ell_2 : \\
\quad \mathsf{merge-}\ \nu_1^{\ell_1} \in \mathcal{I}(\mu_p, \mu, \mu_1) \wedge \mu_1 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \mathsf{merge+}\ \nu_2^{\ell_2} \in \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mu_2 \in \mathcal{I}(\mu_{gp}, \mu_p, \mu) \wedge \\
\quad \mathsf{par}_{P_\star}(\mathsf{merge-}\ \cdot^{\ell_1}, \mathsf{merge+}\ \cdot^{\ell_2}) \wedge \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_1, \nu_1) \cap \langle\mathcal{R}\rangle(\mu_p, \mu, \mu_2, \nu_2) \neq \emptyset \Rightarrow \\
\qquad \mathcal{I}(\mu_p, \mu, \mu_1) \subseteq \mathcal{I}(\mu_p, \mu, \mu_2) \wedge \mathcal{I}(\mu, \mu_1) \subseteq \mathcal{I}(\mu, \mu_2) \wedge \mathcal{I}(\mu_1) \subseteq \mathcal{I}(\mu_2) \wedge \\
\qquad (\forall \nu : \mathsf{nrel}_{P_\star}(\mathsf{merge-}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu_p, \mu, \mu_1, \nu) \subseteq \mathcal{R}(\mu_p, \mu, \mu_2, \nu)) \wedge \\
\qquad (\forall \nu, \mu' : \mathsf{nrel}_{P_\star}(\mathsf{merge-}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu, \mu_1, \mu', \nu) \subseteq \mathcal{R}(\mu, \mu_2, \mu', \nu)) \wedge \\
\qquad (\forall \nu, \mu', \mu'' : \mathsf{nrel}_{P_\star}(\mathsf{merge-}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu_1, \mu', \mu'', \nu) \subseteq \mathcal{R}(\mu_2, \mu', \mu'', \nu)) \\
\mathsf{closure}_{\mathsf{merge+}}\ \cdot = \mathsf{true}
\end{array}
$$

Fig. 16. Closure conditions for movement capabilities.

μ_1 itself, the children of μ_1, or the grandchildren of μ_1.

- The $\mathcal{R}$ relation must be updated to reflect that the *relevant variables* known in μ_1 are also known in μ_1 in the new context of μ_2. Note that the relevant variables are those in μ_1 that occur free either in parallel with the enter capability or sequentially after it. In other words we only exclude variables in threads that "die" due to non-deterministic choices. A similar update is needed for the relevant variables in a child μ' of μ_1:

$$
\begin{array}{l}
(\forall \nu : \mathsf{nrel}_{P_\star}(\mathsf{enter}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu_p, \mu, \mu_1, \nu) \subseteq \mathcal{R}(\mu, \mu_2, \mu_1, \nu)) \wedge \\
(\forall \nu, \mu' : \mathsf{nrel}_{P_\star}(\mathsf{enter}\ \cdot^{\ell_1}, \nu) \Rightarrow \mathcal{R}(\mu, \mu_1, \mu', \nu) \subseteq \mathcal{R}(\mu_2, \mu_1, \mu', \nu))
\end{array}
$$

  As only the ambient hosting the enter capability moves, the corresponding accept capability does not cause similar updates.

Note that by copying name bindings only for the $\mathcal{R}$ relation and not for $\langle\mathcal{R}\rangle$ we achieve an elegant restriction such that only variable bindings are copied; in the preconditions, however, we duly take all names into account.

The closure condition for static scope ensures that all variable bindings are recorded *throughout scope*, i.e. the binding information is propagated to all contexts where the variable is in scope - as recorded by $\mathsf{nrel}_{P_\star}$.

The structure of the clause is similar to that of the previous closure con-

$$\begin{aligned}\mathsf{closure}_{\mathsf{scp}} = \ & \forall \mu_{gp}, \mu_p, \mu, \mu_c : \mathcal{I}(\mu_{gp}, \mu_p, \mu, \mu_c) \Rightarrow \\ & \quad \forall \nu : \nu \in \mathsf{nrel}_{P_\star}(\mu_c) \Rightarrow \mathcal{R}(\mu_{gp}, \mu_p, \mu, \nu) \subseteq \mathcal{R}(\mu_p, \mu, \mu_c, \nu)\end{aligned}$$

Fig. 17. Closure condition for the propagation of name scopes.

μ_{gp}	μ_p	μ	$\mathcal{I}(\mu_{gp}, \mu_p, \mu)$
$\top$	$\top$	$\top$	$food, system$
$\top$	$\top$	$system$	$cell, food,$ expel rj^{ℓ_1}
$\top$	$\top$	$food$	enter ac^{ℓ_2}, exit rj^{ℓ_3}, $rea\hat{}?\{rl\}^{\ell_4}$, expel rl^{ℓ_5}, $nutrient$
$\top$	$system$	$food$	$nutrient$, expel rl^{ℓ_5}, $rea\hat{}?\{rl\}^{\ell_4}$, exit rj^{ℓ_3}, enter ac^{ℓ_2}
$\top$	$system$	$cell$	$nutrient, food, rea\_!\{RL\}^{\ell_9}$, expel rj^{ℓ_8}, accept ac^{ℓ_7}
$\top$	$food$	$nutrient$	exit RL^{ℓ_6}
$system$	$food$	$nutrient$	exit RL^{ℓ_6}
$system$	$cell$	$food$	enter ac^{ℓ_2}, exit rj^{ℓ_3}, $rea\hat{}?\{rl\}^{\ell_4}$, expel rl^{ℓ_5}, $nutrient$
$system$	$cell$	$nutrient$	exit RL^{ℓ_6}
$cell$	$food$	$nutrient$	exit RL^{ℓ_6}

μ_{gp}	μ_p	μ	ν	$\mathcal{R}(\mu_{gp}, \mu_p, \mu, \nu)$
$system$	$cell$	$food$	rl	RL

Fig. 18. Analysis result.

ditions but is much simpler:

- The first precondition is used to fix ambients μ_c and their containment contexts μ_{gp}, μ_p, μ:
$$\mathcal{I}(\mu_{gp}, \mu_p, \mu, \mu_c)$$
- The second precondition is used to fix the variables ν whose scopes extend over the ambients:
$$\nu \in \mathsf{nrel}_{P_\star}(\mu_c)$$
- As scoping is static this information is sufficiently precise for determining a correct propagation of the localised environment from 'outside μ_c' into 'inside μ_c':
$$\mathcal{R}(\mu_{gp}, \mu_p, \mu, \nu) \subseteq \mathcal{R}(\mu_p, \mu, \mu_c, \nu)$$

Note that both the $\mathcal{I}$ relation and the $\mathsf{nrel}_{P_\star}$ relation may mistakenly provide labelled capabilities in place of the expected μ_c ambient - potentially polluting the $\mathcal{R}$ relation. This is not a problem in practice as $\mathcal{I}$ provides canonical capabilities whereas $\mathsf{nrel}_{P_\star}$ provides skeleton capabilities, i.e. the two relations cannot *agree* on a capability to use instead of an ambient.

Example 4.1 The least fixed point analysis of the running example P_{eat} gives rise to the $\mathcal{I}$ and $\mathcal{R}$ components shown in Fig. 18. Fig. 19 gives a graphical relation of the ambient part of the containment relation $\mathcal{I}$. The triple bordered node represents the super-environment, the double bordered nodes connected by bold lines represent the initial configuration (Fig. 13), and the remaining nodes represent the system dynamics (Figs. 14, 16, and 17). The trees of the individual frames of Fig. 5 are all sub-trees of this figure. Note,

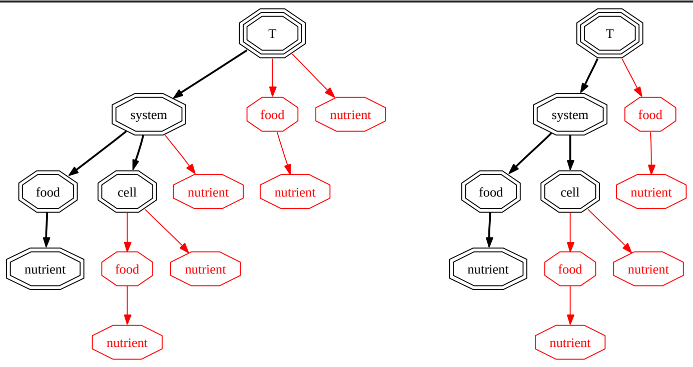


Fig. 19. Spatial Analysis of P_{eat}. Left: Previous 0CFA. Right: Present 2CFA.

that although the analysis is formally an over-approximation the result is rather precise.

4.3 Correctness

The correctness of the analysis is established by a number of invariance results. First of all we have:

Lemma 4.2 (Invariance under congruence)

If $P \equiv P'$ *and* $\mathbf{C} \vdash P$ *then* $(\mathcal{I}, \mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P$ *iff* $(\mathcal{I}, \mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P'$

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

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

The correctness of the analysis then follows from:

Theorem 4.3 (Invariance under reduction)

If $\mathsf{PRG}_{\mathbf{C}}(P)$, $(\mathcal{I}, \mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P$, $\mathsf{par}_{[],[]}(P) \subseteq \mathsf{par}_{P_\star}$, $(\mathsf{npar}_{[],[],[]}(P) \cup \mathsf{nfol}_{[],[]}(P)) \subseteq \mathsf{nrel}_{P_\star}$, *and* $P \to^\star P'$ *then* $(\mathcal{I}@\mathcal{R}, \mathcal{R}) \models^{\mu_{gp}\mu_p\mu} P'$

5 Conclusion

We have adapted the Flow Logic approach to static analysis to apply to Bio-Ambients performing several extensions on the way. In particular the treatment of non-deterministic choice and general recursion have posed some challenges. Precision has been increased by using three syntactic auxiliary analyses and by incorporating context in the manner of 2CFA. Our Succinct Solver implementation strategy favours sparse relations and we achieve this by using representatives only in the containment relation; for formulating semantic correctness it then needs to be "closed" using the environment (i.e. $\mathcal{I}@\mathcal{R}$).

Applying the development to an overly simplistic model of nutrient extraction we have illustrated how the analysis provides interesting information about biological systems and shown (Fig. 19) that it is more precise than previous contributions. The more elaborate example reported in [16] further illustrates the usefulness of the analysis.

Acknowledgements. This work has been supported by the Danish Natural Science Research Council project LoST (21-02-0507).

References

[1] Cardelli, L., *Bioware languages*, in: *Computer Systems - Papers for Roger Needham*, Springer, pp. 59–65, 2003.

[2] Cardelli, L. and A. D. Gordon, *Mobile ambients*, Theoretical Computer Science **240** (2000), pp. 177–213.

[3] Bodei, C. et al, *An enhanced CFA for security policies.*, in: *Proc. of WITS'03*, pp. 131–145, 2003.

[4] Levi, F. and S. Maffeis, *An abstract interpretation framework for analysing Mobile Ambients*, in: *Proc. of SAS'2001*, LNCS **2126** (2001), pp. 395–411.

[5] Levi, F. and D. Sangiorgi, *Controlling interference in ambients*, in: *Proc. of POPL'2000* (2000), pp. 352–364.

[6] Milner, R., "Communicating and Mobile Systems: The π-Calculus," Cambridge University Press, 1999.

[7] Nielson, F. and H. R. Nielson, *Flow Logic: a multi-paradigmatic approach to static analysis*, in: *The Essense of Computation: Complexity, Analysis, Transformation*, LNCS **2566** (2002), pp. 223–244.

[8] Nielson, F., H. R. Nielson and C. Hankin, "Principles of Program Analysis," Springer, 1999.

[9] Nielson, F., H. R. Nielson and R. R. Hansen, *Validating firewalls using Flow Logics*, Theoretical Computer Science **283** (2002), pp. 381–418.

[10] Nielson, F. et al, *Control flow analysis for BioAmbients*, in: *Proc. of BioConcur'03*, 2003.

[11] Nielson, F., H. R. Nielson and H. Seidl, *A succinct solver for ALFP*, Nordic Journal of Computing **9** (2002), pp. 335–372.

[12] Nielson, F. et al, *The succinct solver suite*, in: *Proc. of TACAS'04*, LNCS **2988** (2004), pp. 251–265.

[13] Nielson, H. R. and F. Nielson, *Shape analysis for mobile ambients*, in: *Proc. of POPL'00* (2000), pp. 142–154.

[14] Nielson, H. R., F. Nielson and M. Buchholtz, *Security for mobility*, in: *FOSAD 2001/2002 Tutorial Lecture Notes*, LNCS **2946** (2004), pp. 207–265.

[15] Nielson, H. R., F. Nielson, H. Pilegaard, *Spatial Analysis of BioAmbients*, Proc. of SAS'04, LNCS **3148** (2004), pp. 69–83.

[16] Pilegaard, H., F. Nielson and H. R. Nielson, *Static analysis of a model of the LDL degradation pathway*, in: G. Plotkin, editor, *Proc. of CMSB'05*, 2005.

[17] Regev, A., "Computational Systems Biology: A Calculus for Biomolecular Knowledge," Ph.D. thesis, Tel Aviv University (2003).

[18] Regev, A. et al, *BioAmbients: An abstraction for biological compartments*, Theoretical Computer Science **325** (2004), pp. 141–167.