A Control Flow Analysis for Beta-Binders With and Without Static Compartments [1]

Chiara Bodei

Dipartimento di Informatica, Università di Pisa,
Via Pontecorvo, 56127 Pisa - Italy
chiara@di.unipi.it

Abstract

We introduce a Control Flow Analysis, that statically approximates the dynamic behaviour of processes, expressed in the Beta-binders calculus and in an extended version of the calculus modelling static compartments. Our analysis of a system is able to describe the essential behaviour of each box, tracking all the possible bindings of variables, all the possible intra- and inter-boxes communications, and, finally, all the possible movements across compartments. The analysis offers a basis for establishing static checks of biological dynamic properties. We apply our analysis to an abstract specification of the interaction between a virus and cells of the immune system and to a model of the $cAMP$-signaling Pathway in Olfactory Sensory Neurons.

Keywords: Control Flow Analysis, Beta-binders, Biological Compartments

1 Introduction

The complexity of biological phenomena calls for a system vision that puts its focus on function rather than on structure, on the behaviour of systems, rather than on the description of single components. This is the idea underlying Systems Biology [10], whose main interest is to study the behaviour emerging from the interaction of components. At the same time, the big amount of raw biological data now available for analysis calls for the development of abstract models able to capture the properties of interest. Furthermore, models should be build incrementally in order to easily integrate new knowledge. The analogy between the study of computer systems and networks and the study of biological systems motivates the convergence of interests of the two communities. Under this regard, *computational thinking* [25], as stated in [18], can be a useful tool for both fields. It is based on the idea of analysing systems at different levels of abstraction and of modelling them through executable formalisms that offer predictions on their dynamic evolution. Among the several

[1] A preliminary version of this paper appeared in [2].

formalisms used to this aim, we recall process calculi, that abstractly describe interactions and communications between independent agents or processes and whose specifications can be incrementally refined. They have been adopted to model biological systems. In particular, π-calculus [11] and Ambient Calculus [5] have been transferred from theoretical computer science setting to the biology setting, where suitable biological versions of them, such as the Biochemical stochastic π-calculus [21,24] and BioAmbients [23] have been introduced. Also a version of CCS, RCCS [7], that addresses biological issues, has been presented. Other calculi have been instead specifically defined for biological modelling, such as κ-calculus [8], Brane calculi [4] and Beta-binders [19]. The underlying idea is that a biological system can be abstractly modelled as a concurrent system. Several approaches – developed to predict the dynamic behaviour of the modelled systems – have introduced the idea of performing *in silico* experiments to establish which *in vitro* experiments are more promising.

The behaviour of a system is usually given in terms of its transition system, whose size can be huge, making its exploration computationally hard. Resorting to static techniques offers the possibility of drastically reducing the computational costs, particular high when modelling complex biological systems. The specification of the system is statically (i.e. at compile time) analysed in order to extract information on the dynamic behaviour and to check the related dynamic properties, without actually running the corresponding program. The price is a loss in precision, because these techniques usually give approximations of the behaviour. Static analysis adds a further level of abstraction, that can be exploited in the modelling phase, to easily tune specifications, in order to capture the properties of interests. As a consequence, it can also be used to perform a sort of preliminary screening of *in silico* experiments. In fact, on the one hand, it offers the possibility of efficiently changing the starting hypotheses. On the other hand, it can predict also rare – but maybe meaningful – events, that can be left out using other approaches, like those based on stochastic simulation.

We present here a Control Flow Analysis for a version of Beta-binders calculus. In this language, processes are enveloped inside boxes, that represent the borders of biological entities. Boxes are equipped with typed interaction sites, through which interactions among boxes can occur. In Beta-binders, nesting of boxes is not explicitly allowed. In order to represent more complex hierarchies useful to model compartments of biological systems, an extended version of Beta-binders has been introduced in [9]. Processes can move across static compartments. The Control Flow Analysis has been extended accordingly. Our Control Flow Analysis safely approximates the behaviour of systems, tracking all the possible bindings of variables and all the possible intra- and inter-boxes communications. In the extended calculus, the analysis also tracks the process movements in and out of compartments. This information offers a basis for studying dynamic properties, by suitably handling the approximation the static analysis introduces. We have indeed an over-approximation of the *exact* behaviour of a system. This means that all those events that the analysis *does not* include will *never* happen, while all the events that the analysis *does* include *can* happen, i.e. they are only possible. Therefore, the analysis offers a basis for establishing static checks of biological dynamic properties.

We can prove some basic facts, that can be immediately exploited to establish simple properties, such as the absence of interaction of two boxes or the isolation of a box. Furthermore, we can deal with the static counterpart of the locality relations introduced in [9].

The paper follows the tradition initiated by [15,14] of applying static techniques and, in particular, Control Flow Analysis to process calculi used in modelling biological phenomena. These analyses are inspired by the analyses previously applied to several process calculi, such as π-calculus (e.g. [3]) and Ambient Calculus (e.g. [13]) to establish security properties. The biological framework requires a suitable tuning and, at the same time, can suggest the introduction of finer or new techniques. Like the above analyses, our is context-insensitive and also flow-insensitive.

The rest of the paper is organised as follows. In Section 2, we present the Beta-binders formalism. We introduce the Control Flow Analysis in Section 3. In Section 4, we propose some possible applications for our analysis. In Section 5, we show Control Flow Analysis at work on an example. The analysis is extended in Section 6 in order to treat the Beta-binders version modelling static biological compartments. In Section 7, the new analysis is applied to a model of the $cAMP$-signaling Pathway in Olfactory Sensory Neurons. Section 8 presents some concluding remarks.

Proofs of theorems and lemmata presented throughout the paper are collected in Appendix A.

2 The Calculus

We briefly introduce the kernel of Beta-binders, that is actually a subset of the calculus and we refer the interested reader to [19,17,22] for more details. In particular, for the sake of simplicity, we do not consider the *join* and the *split* semantic constructs. These constructs make it possible to merge and split boxes. The strategies to do it can be chosen, relying on the definition of (one or more instances of) the computable functions f_{join} and f_{split}. Their operational semantics is parametric with respect to the definition of these functions. Their generality and parametricity would make the Control Flow Analysis too loose and approximate. Alternatively, we could deal with specific definitions of these functions, loosing generality, though. As a consequence, we preferred not to use them, and to resort to a suitable encoding, when needed.

2.1 Syntax

Essentially, processes are the parallel composition of boxes that contain π-calculus like [11] processes. As in the π-calculus, communication can occur when an input and an output synchronize on a particular channel. Boxes represent the borders of biological entities and are equipped with typed interaction sites or binders, which regulate the interactions with the environment. As in the π-calculus, we assume the existence of a countably infinite set $\mathbf{N}$ of names (ranged over by lower-case letters).

Beta binders

A special class of binders, called *beta binders* is introduced. Each binder characterises an interaction site and an associated type.

Definition 2.1 An *elementary beta binder* has either the form $\beta(x : \Gamma)$ (*active binder*) or the form $\beta^h(x : \Gamma)$ (*hidden binder*), where we let $\hat{\beta} \in \{\beta, \beta^h\}$ and:

- the name x is the *subject* of the beta binder,
- Γ is the *type* of x. It is a non-empty set of names such that $x \notin \Gamma$.

Definition 2.2 *Composite beta binders* are generated by the following grammar:

$$\mathbf{B} ::= \beta(x : \Gamma) |\ \beta^h(x : \Gamma) |\ \beta(x : \Gamma)\mathbf{B} |\ \beta^h(x : \Gamma)\mathbf{B}$$

A composite beta binder is said to be *well-formed* when the subjects of its elementary components are all distinct. We let well-formed beta binders be ranged over by $\mathbf{B}, \mathbf{B_1}, \mathbf{B_2}, ...$ Let $sub(\mathbf{B})$ denote the set of the subjects of all the elementary beta binders in $\mathbf{B}$, and $\mathbf{B}^*$ denote either a beta binder or the empty string of elementary beta binders.

Pi-processes

Processes, which are incapsulated into boxes, are a variation of standard π-calculus, that makes handling of beta binders possible from inside boxes. They are called *pi-processes*.

$\mathbf{P} ::=$	$Pi - processes$
nil	inactive process
$(\nu x)P$	restriction
$P\|P$	parallel composition
$!P$	replication
$\overline{x}\langle y\rangle.P$	output
$x(y).P$	input
$\mathsf{expose}(x, \Gamma).P$	addition of a new site
$\mathsf{hide}(x).P$	hiding of a site
$\mathsf{unhide}(x).P$	unhiding of a site

Pi-processes behave just as π-calculus processes. The process nil is inactive. The name x in $(\nu x)P$ acts as a static binder for x in P. The operator $|$ describes parallel composition of processes. Intuitively, P and Q in $P|Q$ act independently and can also communicate when one performs an input and the other an output on the same common link. Replication $!P$ behaves as $P|P|\cdots$ as many times as needed. The output prefix $\overline{x}\langle y\rangle$ sends name y on link x. The input prefix $x(y)$ binds the name y in the prefixed process. Intuitively, some name y is received along the link named x. The additional prefixes $\mathsf{expose}(x, \Gamma)$, $\mathsf{hide}(x)$, $\mathsf{unhide}(x)$ are

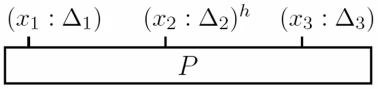
used to change the interaction capabilities of boxes. Prefixes $\mathsf{hide}(x)$ and $\mathsf{unhide}(x)$ make the elementary binder with subject x not available (hidden) and available (not hidden), respectively. A hidden binder cannot be used in a communication. The prefix $\mathsf{expose}(x, \Gamma)$ adds to the box the binder $\beta(x, \Gamma)$. The definitions of *name substitution* and of *free* and *bound names* (defined as $\mathsf{fn}()$ and $\mathsf{bn}()$) are treated as in the π-calculus and are extended to the above syntax, by stipulating that $\mathsf{expose}(x, \Gamma)$ is a binder for x in P. Moreover, the names occurring in the types declared in expose prefixes are free, and therefore can be affected by substitution.

Beta-processes

The set of *beta-processes*, ranged over by $B, B_1, B_2, ...$ is defined as follows. Boxes are nameless entities, but to facilitate our analysis, we annotate boxes as in $\mathbf{B}[P]^\mu$, in order to distinguish different syntactic occurrences of boxes. We refer to $\mu \in \mathbf{Box}$ as the identity of the box, where $\mathbf{Box}$ is the finite set of box identities.

$\mathcal{B}$	::=	$Beta - processes$
	Nil	inactive beta-process
	$\mathbf{B}[P]^\mu$	basic beta-process
	$B \| \| B$	parallel composition

A beta-process is either empty (Nil) or a single box ($\mathbf{B}[P]^\mu$) or the parallel composition of two boxes ($B||B$). When in the above grammar $\mathbf{B}$ is taken to be *well-formed*, the generated process B is said to be *well-formed*. Graphically, each process ($\mathbf{B}[P]^\mu$) can be rendered as a single box, where binders indicate the sites through which the box can interact with the external world. For instance, the process $\beta(x_1 : \Delta_1)\beta^h(x_2 : \Delta_2)\beta(x_3 : \Delta_3)[P]$ is depicted as



Finally, note that nesting of boxes is forbidden (see also Section 4).

2.2 Semantics

Since names can be α-converted, they cannot be used as they are in the Control Flow Analysis in Section 3. To circumvent this problem, we discipline the α-conversion of names. To this aim, we partition all the names used by a process into finitely many equivalence classes and we use the names of the equivalence classes instead of the actual names. The partition works in a way that names from the same equivalence class are assigned a common *canonical name*. Consequently there are only finitely many canonical names in any execution of a given process. This is enforced by assigning the same canonical name to every name generated by the same restriction. For example, consider a process like $!(\nu n)P$, that may generate

infinitely many names, as shown in the following chain of equivalences:

$$!(\nu n)P \equiv (\nu n')P' \mid !(\nu n)P \equiv (\nu n')P' \mid (\nu n'')P'' \mid !(\nu n)P \equiv \ldots$$

All the names that can be generated, e.g. n, n' and n'', have the same canonical name.

The canonical name $\lfloor n \rfloor$ is for a name n. Not to further overload our notation, we simply write n for $\lfloor n \rfloor$, when unambiguous. Furthermore, we demand that two names are α-convertible only when they have the same canonical name. In this way, we statically maintain the identity of values and variables that may be lost by freely applying α-conversions. The α-renaming discipline is included in the structural congruence rules given below. Finally, we assume that all the binders names are distinct and that all the bound names of a process are renamed apart and that they do not clash with the free names. In particular, the names used in the expose prefixes are distinct from the names occurring in their contexts.

The semantics of beta-processes is given in terms of a reduction semantics, that in turn, uses a structural congruence relation. Below we present the standard *structural congruence* $\equiv$ on pi-processes and beta-processes. The symbol $\equiv$ is overloaded and holds in both cases; the context can disambiguate the intended relation. The structural congruence over pi-processes is standard and is the least congruence satisfying the following clauses:

- $P \equiv Q$ if P and Q are disciplined α-equivalent (as explained above);
- $(\mathcal{P}_{\equiv}, |, nil)$ is a commutative monoid;
- $(\nu n)nil \equiv nil$, $(\nu n)(\nu n')P \equiv (\nu n')(\nu n)P$, $(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin \mathsf{fn}(P)$,
- $!P \equiv P \mid !P$

The structural congruence over beta-processes is the least congruence satisfying the following clauses:

- $\mathbf{B}[P]^{\mu} \equiv \mathbf{B}[Q]^{\mu}$ if $P \equiv Q$;
- $(\mathcal{B}/_{\equiv}, ||, Nil)$ is a commutative monoid;
- $\mathbf{B}_1\mathbf{B}_2[P]^{\mu} \equiv \mathbf{B}_2\mathbf{B}_1[P]^{\mu}$,
- $\mathbf{B}^*\hat{\beta}(x:\Gamma)[P]^{\mu} \equiv \mathbf{B}^*\hat{\beta}(x':\Gamma)[P\{x'/x\}]^{\mu}$, provided that $\lfloor x' \rfloor = \lfloor x \rfloor$.

The axioms over beta-processes state, respectively, that: (i) the structural congruence of pi-processes is reflected at the level of boxes; (ii) the parallel composition of beta-processes is a monoidal operation with neutral element; (iii) the actual ordering of elementary beta binders within a composite binder is irrelevant; (iv) the subject of an elementary beta binder is a placeholder that can be changed at any time under the proviso that name clashes are avoided and well-formedness of beta binders is preserved.

The *reduction relation*, $\rightarrow$, is the smallest relation over beta-processes obtained by applying the axioms and rules in Table 1. The semantics preserves the well-formedness of processes. We assume that all the names are initially distinct. Furthermore, we use $\tilde{u}$ as a shorthand for $\{u_1, ..., u_n\}$ and $(\nu\tilde{u})$ for $\{(\nu u_1), ..., (\nu u_n)\}$. The axiom $(Intra)$ lifts to the level of beta-processes any communication between

pi-processes within the same box. The axiom $(Inter)$ models beta-processes interactions between boxes with complementary action (input/output) over complementary sites (with non disjoint types). When the types are non disjoint, we call them *compatible*. The rules $(Expose)$, $(Hide)$, and $(Unhide)$ allow the dynamic modifications of beta binders. The rule $(Expose)$ adds an extra site with the declared type. The name introduced is assumed not to clash with both the subjects of the other binders of the containing box, and with the free names of processes outside the scope of the binding expose prefix. Finally, the rule $(Hide)$, and $(Unhide)$ force the specified site to become hidden and unhidden, respectively.

$$(Intra)\quad \frac{P \equiv (\nu\tilde{u})(x(w).P_1|\ \overline{x}\langle z\rangle.P_2|\ P_3)}{\mathbf{B}[P]^{\mu} \rightarrow \mathbf{B}[(\nu\tilde{u})(P_1\{z/w\}|\ P_2|\ P_3)]^{\mu}}$$

$$(Inter)\quad \frac{P \equiv (\nu\tilde{u})(x(w).P_1|\ P_2) \qquad Q \equiv (\nu\tilde{v})(\overline{y}\langle z\rangle.Q_1|\ Q_2)}{\beta(x:\Gamma)\mathbf{B}^{*}_{\mathbf{1}}[P]^{\mu_P}||\beta(y:\Delta)\mathbf{B}^{*}_{\mathbf{2}}[Q]^{\mu_Q} \rightarrow \beta(x:\Gamma)\mathbf{B}^{*}_{\mathbf{1}}[P']^{\mu_P}||\beta(y:\Delta)\mathbf{B}^{*}_{\mathbf{2}}[Q']^{\mu_Q}}$$

where $P' = (\nu\tilde{u})(P_1\{z/w\}|\ P_2)$ and $Q' = (\nu\tilde{v})(Q_1|\ Q_2)$

provided that $\Gamma \cap \Delta \neq \emptyset$ and $x, z \notin \tilde{u}$ and $y, z \notin \tilde{v}$

$$(Expose)\quad \frac{P \equiv (\nu\tilde{u})(\mathsf{expose}(x,\Gamma).P_1|\ P_2)}{\mathbf{B}[P]^{\mu} \rightarrow \mathbf{B}\beta(x:\Gamma)[(\nu\tilde{u})(P_1|\ P_2)]^{\mu}}$$

provided that $x \notin (\tilde{u} \cup sub(\mathbf{B}) \cup \mathsf{fn}(P_2))$

$$(Hide)\quad \frac{P \equiv (\nu\tilde{u})(\mathsf{hide}(x).P_1|\ P_2)}{\mathbf{B}^{*}\beta(x:\Gamma)[P]^{\mu} \rightarrow \mathbf{B}^{*}\beta^{h}(x:\Gamma)[(\nu\tilde{u})(P_1|\ P_2)]^{\mu}}$$

provided that $x \notin \tilde{u}$

$$(Unhide)\quad \frac{P \equiv (\nu\tilde{u})(\mathsf{unhide}(x).P_1|\ P_2)}{\mathbf{B}^{*}\beta^{h}(x:\Gamma)[P]^{\mu} \rightarrow \mathbf{B}^{*}\beta(x:\Gamma)[(\nu\tilde{u})(P_1|\ P_2)]^{\mu}}$$

provided that $x \notin \tilde{u}$

$$(Par)\quad \frac{B_0 \rightarrow B_0'}{B_0||B_1 \rightarrow B_0'||B_1} \qquad (Struct)\quad \frac{B \equiv B_0\ \wedge\ B_0 \rightarrow B_1\ \wedge\ B_1 \equiv B'}{B \rightarrow B'}$$

Table 1
Reduction Semantics for Beta-binders.

3 Static Analysis

We develop a Control Flow Analysis for analysing beta-processes, based on the analysis of π-calculus [3] and borrowing some ideas from [14]. The aim of the analysis is to safely over-approximate all the possible behaviour. The result of analysing a beta-process B and a pi-process P, is a tuple $(\iota, \epsilon, \rho, \kappa)$, called *estimate* for B (P, respectively), that satisfies the judgements defined by the axioms and rules in the upper (lower, respectively) part of Table 2. The analysis is defined in the flavour of Flow Logic [12]. The first component ι gives information on the beta binders of boxes. The second component ϵ, given a subject of a beta binder, gives information about the set of names it may be associated with. The third component ρ gives information about the set of values to which names can be bound. Finally, the last component, κ gives information about the set of channels that can be sent over given channels.

To *validate* the correctness of a proposed estimate $(\iota, \epsilon, \rho, \kappa)$ we state a set of clauses operating upon judgements for analysing beta-processes $(\iota, \epsilon, \rho, \kappa) \models^* B$ and for analysing pi-processes $(\iota, \epsilon, \rho, \kappa) \models^\mu P$. If $(\iota, \epsilon, \rho, \kappa) \models^* B$, then $(\iota, \epsilon, \rho, \kappa)$ is an *acceptable* estimate of the behaviour of B, i.e. it is valid also for all the states B' passed through a computation of B. More precisely,

- $*$ stands for the universe in which boxes are, while $\mu \in \mathbf{Box}$ annotates $\models$ to keep track of the box in which the process is.
- $\iota : \{*\} \cup \mathbf{Box} \to \wp(\mathbf{Box}) \uplus (\mathbf{N} \cup \mathbf{N^h})$ is the *binder repository*, where $\wp(S)$ stands for the power-set of the set S, $\uplus$ stands for the disjoint union operator, $\mathbf{N^h}$, ranged over by x^hrepresent the set of names in the hidden version, and $\mathbf{N} \cup \mathbf{N^h}$ is ranged over by $\hat{x}$. In $\iota(*)$ are collected all the names μ of the boxes under analysis. In $\iota(\mu)$ are collected instead all the beta binder names x that are declared in the box labelled μ, in the form x or x^h. If $x \in \iota(\mu)$, then the corresponding name can appear in an active binder or in an unhide prefix. In both cases, x is considered a potentially active binder, i.e. it *can* be involved in a communication. If $x^h \in \iota(\mu)$, then the corresponding name can appear in a hidden binder or in a hide prefix.
- $\epsilon : \mathbf{Box} \to (\mathbf{N} \to \wp(\wp(\mathbf{Box})))$ is the *abstract binder environment* that maps a beta binder of a certain box μ to all its possible type sets, i.e. if $G \in \epsilon(\mu)(x)$ then the values in G may be included in the type of the binder x. To simplify the clauses, we will use the boolean predicate $comp(\epsilon(\mu)(a), \epsilon(\mu')(b))$ in place of the condition $\exists G \in \epsilon(\mu)(a), D \in \epsilon(\mu')(b) : G \cap D \neq \emptyset$. The predicate is true when the beta binders are compatible.
- $\rho : \mathbf{N} \to \wp(\mathbf{N})$ is the *abstract environment* that maps a name to the set of beta binders or names it can be bound to, i.e. if $v \in \rho(x)$ then x may take the value v. We assume that for each free name x, we have that $\rho(x) \ni x$. Moreover, we write $\rho(\Gamma)$ as a shorthand for $\rho(v_1) \cup .... \cup \rho(v_n)$, where $\Gamma = \{v_1, ..., v_n\}$.
- $\kappa : \mathbf{Box} \to (\mathbf{N} \to \wp(\mathbf{N}))$ is the *abstract channel environment* that maps a beta binder or a name occurring in a box μ, to the set of values that can be sent over it, i.e. if $v \in \kappa(\mu)(x)$ then the value v can be sent on the channel x in the box labelled μ.

For keeping the analysis component finite, as said above, we have partitioned

$(\iota,\epsilon,\rho,\kappa)$	$\models^* Nil$	iff $true$
$(\iota,\epsilon,\rho,\kappa)$	$\models^* B_0 \| B_1$	iff $(\iota,\epsilon,\rho,\kappa) \models^* B_0 \wedge (\iota,\epsilon,\rho,\kappa) \models^* B_1$
$(\iota,\epsilon,\rho,\kappa)$	$\models^* \beta(x:\Gamma)\mathbf{B}[P]^\mu$	iff $x \in \iota(\mu) \wedge \rho(\Gamma) \in \epsilon(\mu)(x) \wedge (\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}[P]^\mu$
$(\iota,\epsilon,\rho,\kappa)$	$\models^* \beta^h(x:\Gamma)\mathbf{B}[P]^\mu$	iff $x^h \in \iota(\mu) \wedge \rho(\Gamma) \in \epsilon(\mu)(x) \wedge (\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}[P]^\mu$
$(\iota,\epsilon,\rho,\kappa)$	$\models^* [P]^\mu$	iff $\mu \in \iota(*) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu nil$	iff $true$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu (\nu x)P$	iff $x \in \rho(x) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu P_0 \| P_1$	iff $(\iota,\epsilon,\rho,\kappa) \models^\mu P_0 \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P_1$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu !P$	iff $(\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu \overline{x}\langle y\rangle.P$	iff $\forall a \in \rho(x) : \rho(y) \subseteq \kappa(\mu)(a) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu x(y).P$	iff $\forall a \in \rho(x) : \kappa(\mu)(a) \subseteq \rho(y) \wedge$ $\forall a \in \rho(x) : a \in \iota(\mu),\ \forall \mu' \in \iota(*) :\ b \in \iota(\mu')$ $comp(\epsilon(\mu)(a), \epsilon(\mu')(b)) \Rightarrow \kappa(\mu')(b) \subseteq \rho(y)$ $\wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu \mathsf{expose}(x,\Gamma).P$	iff $x \in \iota(\mu) \wedge x \in \rho(x) \wedge \rho(\Gamma) \in \epsilon(\mu)(x) \wedge$ $(\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu \mathsf{hide}(x).P$	iff $\forall a \in \rho(x) :\ a^h \in \iota(\mu) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa)$	$\models^\mu \mathsf{unhide}(x).P$	iff $\forall a \in \rho(x) :\ a \in \iota(\mu) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$

Table 2
Analysis for Beta-processes: $(\iota,\epsilon,\rho,\kappa) \models^* B$ and for Pi-processes: $(\iota,\epsilon,\rho,\kappa) \models^\mu P$.

all the names used by a process into finitely many equivalence classes and we have used the names of the equivalence classes instead of the actual names.

Analysis of Beta-processes

The analysis of beta-processes is in the upper part of Table 2. The clauses follow the structure of beta-processes. The rule for *inactive beta-process* does not restrict the analysis result, while the rule for *parallel composition* || ensures that the analysis also holds for the immediate subprocesses.

The rules for composite beta binders check whether the types Γ of new binders are included in the component ϵ. Furthermore, in the active binder case (hidden binder case, resp.) the inclusion of x (x^h, resp.) in $\iota(\mu)$ is checked. As a consequence, the presence of each beta binder in a process is reflected in the components ϵ and ι. Finally, when the string of beta binders is empty, the analysis proceeds on the pi-process P inside the box, by keeping track of the label μ of the box: $(\iota,\epsilon,\rho,\kappa) \models^\mu P$.

Analysis of Pi-processes

The analysis of pi-processes is in the lower part of Table 2. Similarly to the upper part rules, the rule for *inactive pi-process* does not restrict the analysis result, while the rules for *parallel composition* |, *restriction*, and *replication* ensure that the analysis also holds for the immediate subprocesses. Note that the names x introduced in inputs are variables that are bound as effect of communications and that the analysis keeps track of all the possible bindings in the component ρ. The first condition of the rule for *output* concerns the set of names that can be sent along each element of $\rho(x)$, inside the box μ. This set, recorded in the component κ, has to include the set of values to which y can evaluate. The rule for *input* is more structured, because it takes into account both the possible communication intra- and inter-boxes. The first conjunct (intra-) requires that the set of names that can be sent along a channel with the same name is included in the set of values to which y can evaluate. The second conjunct (inter-) requires that set of values to which y can evaluate includes the set of names that can be sent along a beta binder name b and received along a. This holds, provided that b (a) occurs active in another box μ' (μ, resp.), and under the condition that the types of b and a are compatible ($comp(\epsilon(\mu)(a), \epsilon(\mu')(b))$). To exemplify, suppose to have the process $P = x(y).P_1 | \ \overline{x}\langle z\rangle.P_2$ inside the box μ_P (see the complete example below). Since z can be sent along the channel x, then it should be included in $\kappa(\mu_P)(x)$, and therefore it could be bound to the variable y occurring in the corresponding input on x, i.e. $z \in \rho(y)$ since $\kappa(\mu_P)(x) \subseteq \rho(y)$. Moreover, imagine to have in parallel another process $Q = \overline{u_1}\langle v_1\rangle.Q_1$ inside the box μ_Q, such that the types of x and of u_1 are compatible (i.e. $comp(\epsilon(\mu_P)(x), \epsilon(\mu_Q)(u_1))$) and can both participate in a communication (i.e. $x, \in \iota(\mu_P)$ and $u_1 \in \iota(\mu_Q)$). Since v_1 can be sent on the channel u_1, i.e. $v_1 \in \kappa(\mu_Q)(u_1)$, therefore it could be bound to y, because a communication between the boxes μ_P and μ_Q is possible.

The rule for expose demands that x is included in $\iota(\mu)$ and in $\rho(x)$, and that the set of all the possible values to which the elements of Γ may evaluate to is included in $\epsilon(\mu)(x)$. The rule for hide (unhide, resp.) demands that for all the possible values a of x, a^h (a, resp.) is included in $\iota(\mu)$. Recall that a unhide(x) should follow an hide(x) or a declaration of a hidden binder for x and therefore there is no need to check whether $\epsilon(\mu)(x)$ includes the type of x. Similarly, a hide(x) should follow an unhide(x) or a declaration of an active binder x.

Example 1: Intra- and Inter-Box Communication

The analysis of the simple beta-process B

$$B = B_P||B_Q||B_R = \beta(x : \{c_1, c_2\})[P]^{\mu_P}||\beta(u_1 : \{c_1\})[Q]^{\mu_Q}||\beta(u_2 : \{c_2\})[R]^{\mu_R}$$

$$P = x(y).P_1 | \ \overline{x}\langle z\rangle.P_2 \quad Q = \overline{u_1}\langle v_1\rangle.Q_1 \quad R = \overline{u_2}\langle v_2\rangle.R_1$$

gives rise to the analysis components ι, ϵ, ρ and κ with the following main entries:

$$\begin{array}{lll} & \iota(*) \ni \mu_P, \mu_Q, \mu_R & \\ \iota(\mu_P) \ni x & \iota(\mu_Q) \ni u_1 & \iota(\mu_R) \ni u_2 \\ \epsilon(\mu_P)(x) \ni \{c_1, c_2\}, & \epsilon(\mu_Q)(u_1) \ni \{c_1\}, & \epsilon(\mu_R)(u_2) \ni \{c_2\} \\ \kappa(\mu_P)(x) \ni z, & \kappa(\mu_Q)(u_1) \ni v_1, & \kappa(\mu_R)(u_2) \ni v_2 \\ \rho(y) \ni z, v_1, v_2 & \rho(n) \ni n \text{ for } n \in \mathsf{fn}(B) & \end{array}$$

In fact, we have that,

- $(\iota, \epsilon, \rho, \kappa) \models^{\mu_P} P$, because $(\iota, \epsilon, \rho, \kappa) \models^{\mu_P} \overline{x}\langle z\rangle.P_2$ and $(\iota, \epsilon, \rho, \kappa) \models^{\mu_P} x(y).P_1$;
- $(\iota, \epsilon, \rho, \kappa) \models^{\mu_Q} Q$, because $\rho(u_1) \ni u_1$, $\rho(v_1) \ni v_1$ and $v_1 \in \kappa(\mu_Q)(u_1)$;
- $(\iota, \epsilon, \rho, \kappa) \models^{\mu_R} R$, because $\rho(u_2) \ni u_2$, $\rho(v_2) \ni v_2$ and $v_2 \in \kappa(\mu_R)(u_2)$;
- $(\iota, \epsilon, \rho, \kappa) \models^{\mu_P} \overline{x}\langle z\rangle.P_2$, because $\rho(x) \ni x$, $\rho(z) \ni z$ and $z \in \kappa(\mu_P)(x)$;
- $(\iota, \epsilon, \rho, \kappa) \models^{\mu_P} x(y).P_1$, because

(i) $z \ni \rho(y)$, since $z \in \kappa(\mu_P)(x)$;
(ii) since $x \in \iota(\mu_P)$, $u_1 \in \iota(\mu_Q)$ and $G_P \cap G_Q \ni c_1$, with $G_P = \epsilon(\mu_P)(x)$ and $G_Q = \epsilon(\mu_Q)(u_1)$ then $v_1 \in \rho(y)$, because $v_1 \in \kappa(\mu_Q)(u_1)$;
(iii) since $x \in \iota(\mu_P)$, $u_2 \in \iota(\mu_R)$ and $G_P \cap G_R \ni c_2$, with $G_P = \epsilon(\mu_P)(x)$ and $G_R = \epsilon(\mu_R)(u_2)$ then $v_2 \in \rho(y)$, because $v_2 \in \kappa(\mu_R)(u_2)$;

Note that $\rho(y)$ includes the value z, hence it correctly predicts a possible communication internal to B_P on the channel x, that corresponds to the following transition (where we annotate the transition arrow $\rightarrow$ with the corresponding semantic rule):

$$B \rightarrow_{(Intra)} \beta(x : \{c_1, c_2\})[P_1\{z/y\} \mid P_2]^{\mu_P} || \beta(u_1 : \{c_1\})[Q]^{\mu_Q} || \beta(u_2 : \{c_2\})[R]^{\mu_R}$$

Moreover, $\rho(y)$ includes both v_1 and v_2, therefore it correctly predicts an interaction between B_P and B_Q, accounting for the following transition with B_Q (similarly with B_R):

$$B \rightarrow_{(Inter)} \beta(x : \{c_1, c_2\})[P_1\{v_1/y\} \mid P_2]^{\mu_P} || \beta(u_1 : \{c_1\})[Q_1]^{\mu_Q} || \beta(u_2 : \{c_2\})[R]^{\mu_R}$$

The beta binder x, u_1 and u_2 are included in their active form inside $\iota(\mu_P)$, $\iota(\mu_Q)$ and $\iota(\mu_R)$, respectively. Moreover, the condition $comp(\epsilon(\mu_P)(x), \epsilon(\mu_Q)(u_1))$ shows that the types of x and u_1 are compatible and a similar condition holds for the types of x and u_2.

Example 2: Interface Handling (1)
The analysis of a slightly different beta-process B'

$$B' = B'_P || B_Q || B'_R = \beta(x : \{c_1, c_2\})[P']^{\mu'_P} || \beta(u_1 : \{c_1\})[Q]^{\mu_Q} || \beta(u_2 : \{b_2\})[R]^{\mu'_R}$$

$$P' = x(y).\mathsf{expose}(z, \{y, b_2\}).z(w).P'_1 \quad Q = \overline{u_1}\langle v_1\rangle.Q_1 \quad R' = \overline{u_2}\langle v_2\rangle.R'_1$$

gives rise to:

$$\begin{array}{lll} & \iota(*) \ni \mu'_P, \mu_Q, \mu'_R & \\ \iota(\mu'_P) \ni x, z & \iota(\mu_Q) \ni u_1 & \iota(\mu'_R) \ni u_2 \\ \begin{cases} \epsilon(\mu'_P)(x) \ni \{c_1, c_2\} \\ \epsilon(\mu'_P)(z) \ni \{v_1, b_2\}, \end{cases} & \epsilon(\mu_Q)(u_1) \ni \{a_1\}, & \epsilon(\mu_Q)(u_2) \ni \{b_2\} \\ \kappa(\mu'_P)(x) = \emptyset, & \kappa(\mu'_R)(u_1) \ni v_1, & \kappa(\mu'_R)(u_2) \ni v_2 \\ & \rho(y) \ni v_1 & \rho(w) \ni v_2 \end{array}$$

The prefix expose causes the addition of z to the beta binders of the first box $B_{P'}$, as stated by the inclusion of z in $\iota(\mu_{P'})$. Furthermore, the first box can initially only communicate with the second one. It can communicate with the third one, after the expose, as shown by the following transitions, where B' becomes

$$\beta(x :\{c_1, c_2\})[\mathsf{expose}(z, \{v_1, b_2\}).z(w).P'_1\{v_1/y\}]^{\mu'_P} || \beta(u_1 :\{c_1\})[Q_1]^{\mu_Q} || \beta(u_2 :\{b_2\})[R']^{\mu'_R}$$

and after the expose and the following communication it becomes:

$$\beta(x : \{c_1, c_2\})\beta(z :\{v_1, b_2\})[P'_1\{v_1/y\}\{v_2/w\}]^{\mu'_P} || \beta(u_1 : \{c_1\})[Q_1]^{\mu_Q} || \beta(u_2 : \{b_2\})[R'_1]^{\mu'_R}$$

Note that after the first communication, $\mathsf{expose}(z, \{y, b_2\})$ becomes $\mathsf{expose}(z, \{v_1, b_2\})$, as correctly reported by the analysis, where $\epsilon(\mu'_P)(z) \ni \{v_1, b_2\}$, because $\rho(\{y, b_2\}) = \rho(y) \cup \rho(b_2) \ni v_1, b_2$.

Example 3: Interface Handling (2)
The analysis of another beta-process B''

$$B'' = B''_P || B_Q || B_R = \beta(x : \{c_1, c_2\})[P']^{\mu''_P} || \beta(u_1 : \{c_1\})[Q]^{\mu_Q} || \beta(u_2 : \{c_2\})[R]^{\mu_R}$$
$$P'' = x(y).\mathsf{hide}(x).x(w).P''_1 \quad Q = \overline{u_1}\langle v_1\rangle.Q_1 \quad R = \overline{u_2}\langle v_2\rangle.R_1$$

gives rise to:

$$\begin{array}{lll} & \iota(*) \ni \mu''_P, \mu_Q, \mu_R & \\ \iota(\mu''_P) \ni x, x^h & \iota(\mu_Q) \ni u_1 & \iota(\mu_R) \ni u_2 \\ \epsilon(\mu''_P)(x) \ni \{c_1, c_2\} & \epsilon(\mu_Q)(u_1) \ni \{c_1\}, & \epsilon(\mu_R)(u_2) \ni \{c_2\} \\ \kappa(\mu''_P)(x) = \emptyset, & \kappa(\mu_Q)(u_1) \ni v_1, & \kappa(\mu_R)(u_2) \ni v_2 \\ & \rho(y) \ni v_1, v_2, & \rho(w) \ni v_1, v_2 \end{array}$$

Note that the prefix hide causes the hiding of x and therefore the isolation of the first box B''_P. For instance, if the first transition leads B'' to

$$\beta(x : \{c_1, c_2\})[\mathsf{hide}(x).x(w).P''_1\{v_1/y\}]^{\mu''_P} || \beta(u_1 : \{c_1\})[Q_1]^{\mu_Q} || \beta(u_2 : \{c_2\})[\overline{u_2}\langle v_2\rangle.R_1]^{\mu_R}$$

and the second to

$$\beta^h(x:\{c_1,c_2\})[x(w).P_1''\{v_1/y\}]^{\mu_P''}||\beta(u_1:\{c_1\})[Q_1]^{\mu_Q}||\beta(u_2:\{c_2\})[\overline{u_2}\langle v_2\rangle.R_1]^{\mu_R}$$

then, there is no possible communication between B_P'' and B_R: the sites x and u_2 have non disjoint types, but x is hidden and therefore no communication is possible on x.

The analysis instead considers that communication as possible and also the analogous communication between x and u_1, as shown by the fact that $\rho(w) \ni v_1, v_2$. We are still on the safe side of approximation, because what the analysis includes corresponds to something that *can* happen. Nevertheless, the analysis is not precise. This observation leads us to the following considerations.

3.1 On the Precision of the Analysis

As seen above, the presence of hide and unhide constructs represents a peculiar source of imprecision in Beta-binders. In fact, they can occur in a particular subprocess included in a certain box, but their effect is on the overall process contained in the box, e.g. in the process $\mathbf{B}\beta(x:\Gamma))[\mathsf{hide}(x).P_1|P_2]$ the firing of hide impacts on both the continuation P_1 and on the parallel process P_2. The decision of hiding and of unhiding is unilateral, but affects the whole context. The beta binder is a shared object, whose access is concurrent. This concurrent feature is responsible for the analysis imprecision. Suppose to have the following process, where $\Gamma \cap \Delta \neq \emptyset$.

$$\mathbf{B_P}\beta(x:\Gamma)[!\mathsf{hide}(x).P_1|\overline{x}\langle z\rangle.P_3|!\mathsf{unhide}(x).P_2] \;||\; \mathbf{B_Q}\beta(w:\Delta)[w(y).Q_1]$$

It is impossible (since it is undecidable) to predict at compile time if the communication between $w(y).Q_1$ and $\overline{x}\langle z\rangle.P_3$ will be fired at run time. The beta binder x could be indeed either hidden or unhidden, depending on which is the last interface operation occurred. In our analysis, the communication is predicted as possible, because of the compatibility of types and because x can appear active.

We could obtain more precision in special cases, like in the process $[\mathsf{hide}(x).P_1|P_2]$, where $x \notin \mathsf{fn}(P_2)$. Here the effect of the hide operation is only on the continuation P_1, where x is hidden until an $\mathsf{unhide}(x)$ occurs. More in general, there are cases in which we can decide, at compile time, by a simple syntactic inspection, in which parts of the process a variable x occurs hidden. This is the case of the process B'' above.

In these cases, to reflect the fact that the hidden occurrences of a variable cannot be used for possible communications, (i) we could replace the variable x with a new variable x_h, representing the hidden version of the corresponding beta binder, and, (ii) we could impose then that for each variable x_h, $\epsilon(\mu)(x_h) = \kappa(\mu)(x_h) = \emptyset$.

With this safeguard, the analysis of the process B'' above would correctly predict the absence of any communication on the hidden occurrence of x, since for $i = 1, 2$, $\nexists G_{P'} \in \epsilon(\mu_{P'})(x_h)$ such that $G_{P'} \cap G_R \neq \emptyset$ with $G_R \in \epsilon(\mu_R)(u_i)$ and $\rho(w) = \emptyset$.

3.2 Correctness of the Analysis

Our analysis is semantically correct with respect to the given semantics, as stated by the following subject reduction result: if $(\iota, \epsilon, \rho, \kappa)$ is a valid estimate for a beta-process B, than it is still a valid estimate also for all the states passed through a computation of B.

In order to obtain this, the following lemmata are necessary. The first states that estimates are resistant to substitution of closed terms for variables, and it holds for both pi-processes and beta-processes.

Lemma 3.1 ***(Substitution result)***

(i) $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P$ *and* $v \in \rho(x)$ *imply* $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P\{v/x\}$;

(ii) $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *and* $v \in \rho(x)$ *imply* $(\iota, \epsilon, \rho, \kappa) \models^{*} B\{v/x\}$;

The second lemma says that an estimate for a process P or for a beta-process B is valid for every process congruent to P or B, respectively.

Lemma 3.2 ***(Invariance of Structural Congruence)***

(i) *If* $P \equiv Q$ *and* $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P$ *then* $(\iota, \epsilon, \rho, \kappa) \models^{\mu} Q$

(ii) *If* $B \equiv B'$ *and* $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *then* $(\iota, \epsilon, \rho, \kappa) \models^{*} B'$

We are now ready to state the subject reduction result.

Theorem 3.3 ***(Subject reduction)***
If $B \rightarrow B'$ *and* $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *then also* $(\iota, \epsilon, \rho, \kappa) \models^{*} B'$.

Below $\rightarrow^{*}$ stands for the reflexive and transitive closure of the transition relation $\rightarrow$. The first result shows that the analysis component κ captures all the intra- and inter-boxes communications that a process can engage in, while the second shows a similar result on the component ρ.

Theorem 3.4 ***(Outputs in κ)***

(i) *If* $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *and* $B \rightarrow^{*} B' \rightarrow B''$, *such that the last transition* $B' \rightarrow B''$ *is derived using the rule (Intra) on the output prefix* $\overline{x}\langle z \rangle$ *in the box labelled* μ, *then* $z \in \kappa(\mu)(x)$.

(ii) *If* $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *and* $B \rightarrow^{*} B' \rightarrow B''$, *such that the last transition* $B' \rightarrow B''$ *is derived using the rule (Inter) on the output prefix* $\overline{y}\langle z \rangle$ *in the box labelled* μ, *then* $z \in \kappa(\mu)(y)$.

Theorem 3.5 ***(Values in ρ)***
If $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *and* $B \rightarrow^{*} B' \rightarrow B''$, *such that* B'' *is either in the form* $\mathbf{B}[P\{v/x\}|P']$ *or in the form* $\mathbf{B_1}[P\{v/x\}|P']||\mathbf{B_2}[Q]$ *then* $v \in \rho(x)$.

Finally, we prove that the analysis components ι and ϵ capture the information on all the beta binders that can appear in a process.

Theorem 3.6 ***(Binders in ι and ϵ)***
If $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *and* $B \rightarrow^{*} B' \rightarrow B''$, *such that the last transition is derived using an interface rule (Expose) or (Hide) or (Unhide) on the binder* x *with type* Γ *and* B'' *includes* $\mathbf{B}[P]^{\mu}$ *and if* $(\hat{x} : \Gamma)$ *occurs in* B'' *then* $\hat{x} \in \iota(\mu) \wedge \rho(\Gamma) \in \epsilon(\mu)$.

3.3 Existence of Estimates

In the previous subsection, we have seen a procedure for verifying whether or not a proposed estimate $(\iota, \epsilon, \rho, \kappa)$ is valid. We now show that for any given B there always is a least choice of ι, ϵ, ρ and κ that is acceptable according to the rules in Table 2, i.e. such that $(\iota, \epsilon, \rho, \kappa) \models B$.

Definition 3.7 The set of proposed solutions can be partially ordered by setting $(\iota, \epsilon, \rho, \kappa) \sqsubseteq (\iota', \epsilon', \rho', \kappa')$ iff $\forall \mu : \iota(\mu) \sqsubseteq \iota'(\mu)$, $\forall \mu, x : \epsilon(\mu)(x) \sqsubseteq \epsilon'(\mu)(x)$ $\forall x : \rho(x) \subseteq \rho'(x)$ and $\forall \mu, x : \kappa(\mu)(x) \subseteq \kappa'(\mu)(x)$.

This suffices for making the set of proposed solutions into a complete lattice; using standard notation we write $(\iota, \epsilon, \rho, \kappa) \sqcup (\iota', \epsilon', \rho', \kappa')$ for the binary least upper bound (defined point-wise), $\sqcap \mathcal{I}$ for the greatest lower bound of a set $\mathcal{I}$ of proposed solutions (also defined pointwise), and $(\bot, \bot, \bot, \bot)$ for the least element.

Definition 3.8 A set $\mathcal{I}$ of proposed estimates is a *Moore family* if and only if it contains $\sqcap \mathcal{J}$ for all $\mathcal{J} \subseteq \mathcal{I}$ (in particular $\mathcal{J} = \emptyset$ and $\mathcal{J} = \mathcal{I}$).

When $\mathcal{I}$ is a Moore family it contains a greatest element ($\sqcap \emptyset$) as well as a least element ($\sqcap \mathcal{I}$).

The following theorem then guarantees that there always is an estimate satisfying the specification in Table 2.

Theorem 3.9 ***(Moore Family)***
For any beta-process B the set $\{(\iota, \epsilon, \rho, \kappa) |\ (\iota, \epsilon, \rho, \kappa) \models^* B\}$ is a Moore family.

Theorem 3.10 ***(Existence of Estimates)***
For any beta-process B, there exists an analysis result $(\iota, \epsilon, \rho, \kappa)$ such that $(\iota, \epsilon, \rho, \kappa) \models^* B$.

Finally, the analysis that computes an estimate that satisfies the judgements in Table 2 can be implemented along the lines of the Control Flow Analysis of the π-calculus and that of the BioAmbients [3,16,14].

4 Possible Application of the Analysis

Our analysis of a system statically approximates the essential behaviour of each box, tracking all the possible bindings of variables and all the possible intra- and inter-boxes communications, recording where and between which communications may occur. In particular, we have an over-approximation of the *exact* behaviour of each box. We consider as effective all the communications that might occur through suitable shared channels inside the box and all those that might occur between the box with boxes endowed with compatible beta binders. At run time, only part of these communications can be however viable, due to the dynamic evolution of processes. As a consequence, on the one hand, we can only assess the possibility of certain events, like communications, to happen, when reported in the analysis estimate. On the other hand, the analysis can guarantee that if an event, such as a communication, is not included in the analysis estimate, then it will *never* happen.

Exploiting the soundness of our analysis, we can therefore prove, among others, the following basic facts, that can be immediately used to establish simple properties, without resorting to the exploration of the whole transition system.

(i) **propensity for communication of a beta binder x in $\mathbf{B}[P]^{\mu_P}$**: The binder x can be possibly involved in a communication if $x \in \iota(\mu_P)$, i.e. if it can occur active.

(ii) **no propensity for communication of a beta binder x in $\mathbf{B}[P]^{\mu_P}$**: The binder x cannot be involved in a communication if $x \notin \iota(\mu_P)$, i.e. if it *cannot* occur active.

(iii) **compatibility between $\mathbf{B}[P]^{\mu_P}$ and $\mathbf{B}[Q]^{\mu_Q}$**: $\mathbf{B}[P]^{\mu_P}$ and $\mathbf{B}[Q]^{\mu_Q}$ have compatible types if $\exists \hat{a} \in \iota(\mu_P), \hat{b} \in \iota(\mu_Q)$ such that $comp(\epsilon(\mu_P)(a), \epsilon(\mu_Q)(b))$.

(iv) **no interaction between $\mathbf{B}[P]^{\mu_P}$ and $\mathbf{B}[Q]^{\mu_Q}$**: The process $\mathbf{B}[P]^{\mu_P}$ cannot communicate with $\mathbf{B}[Q]^{\mu_Q}$, if $\forall a \in \iota(\mu_P), \forall b \in \iota(\mu_Q)$: $\neg comp(\epsilon(\mu_P)(a), \epsilon(\mu_Q)(b))$ or $a \notin \iota(\mu_P)$ or $b \notin \iota(\mu_Q)$ i.e. all the possible pairs of beta binders either do not have propensity for communication or are not compatible.

(v) **isolation of $\mathbf{B}[P]^{\mu_P}$**: The process $\mathbf{B}[P]^{\mu_P}$ is isolated, when $\forall \mu_Q \in \iota(*)$, there is no interaction between $\mathbf{B}[P]^{\mu_P}$ and $\mathbf{B}[Q]^{\mu_Q}$.

(vi) **no flow of information from $\mathbf{B}[P]^{\mu_P}$ to $\mathbf{B}[Q]^{\mu_Q}$**: The process $\mathbf{B}[P]^{\mu_P}$ cannot send anything to $\mathbf{B}[Q]^{\mu_Q}$, when $\forall a : a \in \iota(\mu_P)$, and $\forall b : b \in \iota(\mu_Q)$ such that $comp(\epsilon(\mu_P)(a), \epsilon(\mu_Q)(b))$, we have that $\kappa(\mu_P)(a) = \emptyset$. i.e. even in the presence of a pair of beta binders that are compatible, and that show propensity for communication, there is no possible output from box μ_P.

(vii) **virtual nesting of $\mathbf{B}[Q]^{\mu_Q}$ in $\mathbf{B}[P]^{\mu_P}$**: $\mathbf{B}[Q]^{\mu_Q}$ is *virtually nested* in $\mathbf{B}[P]^{\mu_P}$, when (i) $\mathbf{B}[P]^{\mu_P}$ and $\mathbf{B}[Q]^{\mu_Q}$ are such that $\mathbf{B}[Q]^{\mu_Q}$ has only one beta binder b and $\exists \hat{a} \in \iota(\mu_P)$ such that $\epsilon(\mu_Q)(b) = \epsilon(\mu_P)(a)$; moreover (ii) $\forall \mu_R \in \iota(*)$, with $\mu_R \neq \mu_P$, there is no interaction between $\mathbf{B}[P]^{\mu_P}$ and $\mathbf{B}[R]^{\mu_R}$.

As far as the last property is concerned, we must recall that nesting is forbidden to keep the formalism simple. However, (see [20]), the operational semantics of interactions between boxes can express a form of virtual nesting, properly defining the types of sites. This happens when the box virtually nested $\mathbf{B}[Q]^{\mu_Q}$, can perform intra-communications and can be involved in inter-communications only with the nesting box $\mathbf{B}[P]^{\mu_P}$ through a site with exactly the same type of the one in $\mathbf{B}[Q]^{\mu_Q}$.

We chose to have two facts on propensity just to illustrate the way static approximation works. If the condition $x \in \iota(\mu_P)$ holds, we can say that x *may* be used for communication at run time. Nevertheless, it also may not. On the other hand, if the opposite condition $x \notin \iota(\mu_P)$ holds, then we are sure that x *cannot* be used for communication at run time.

5 Example: An Abstract Virus Attack

We illustrate our approach, by using the abstract specification, used in [19], of the interaction between a virus and cells of the immune system. The specification describes a cell C of the immune system that has engulfed the virus V_1 and that has to elaborate it, produce the antigene molecule and display the antigene on its

surface. A specialized lymphocyte L_1 can recognize the antigen a_1 associated with viruses of sort v_1 (the antigen a'_1 associated with viruses of sort v'_1, respectively) and then activate the immune replay.

$$B = B_C||B_L = \beta(x : \{v_1, ..., v_n\})[C]^{\mu_C}||\beta(z : \{a_1, a'_1\})[L_1]^{\mu_{L_1}}$$

$$C = !x(w).\mathsf{expose}(u, \{w\}).\overline{u}\langle r\rangle|\ C_1|\ V_1 \quad V_1 = \overline{x}\langle a_1\rangle.V_1^{res} \quad L_1 = z(y).L_1^{act}$$

The analysis gives rise to:

$$\begin{array}{ll}
\iota(*) \ni \mu_C, \mu_{L_1} & \\
\iota(\mu_C) \ni x, u & \iota(\mu_{L_1}) \ni z \\
\left\{\begin{array}{l}\epsilon(\mu_C)(x) \ni \{v_1, ..., v_n\} \\ \epsilon(\mu_C)(u) \ni \{a_1\},\end{array}\right. & \epsilon(\mu_{L_1})(z) \ni \{a_1, a'_1\} \\
\left\{\begin{array}{l}\kappa(\mu_C)(u) \ni r, \\ \kappa(\mu_C)(x) \ni a_1,\end{array}\right. & \kappa(\mu_{L_1})(z) = \emptyset \\
\rho(w) \ni a_1 & \rho(y) \ni r
\end{array}$$

The following computation is reflected by the above analysis results.

$$\begin{array}{ll}
B \equiv_{[4]} & \beta(x : \{v_1, ..., v_n\})[x(w).\mathsf{expose}(u, \{w\}).\overline{u}\langle r\rangle|\ C_1|\ \overline{x}\langle a_1\rangle.V_1^{res}|C]^{\mu_C}||\beta(z : \{a_1, a'_1\})[L_1]^{\mu_{L_1}} \\
\rightarrow_{Intra} & \beta(x : \{v_1, ..., v_n\})[\mathsf{expose}(u, \{a_1\}).\overline{u}\langle r\rangle|\ C_1|\ V_1^{res}|C]^{\mu_C}||\beta(z : \{a_1, a'_1\})[L_1]^{\mu_{L_1}} \\
\rightarrow_{Expose} & \beta(x : \{v_1, ..., v_n\})\beta(u : \{a_1\})[\overline{u}\langle r\rangle|\ C_1|\ V_1^{res}|C]^{\mu_C}||\beta(z : \{a_1, a'_1\})[z(y).L_1^{act}]^{\mu_{L_1}} \\
\rightarrow_{Inter} & \beta(x : \{v_1, ..., v_n\})\beta(u : \{a_1\})[C_1|\ V_1^{res}|C]^{\mu_C}||\beta(z : \{a_1, a'_1\})[L_1^{act}\{r/y\}]^{\mu_{L_1}}
\end{array}$$

Note that, in particular, at the beginning, the two boxes cannot communicate each other, because their beta binders are not compatible.

Suppose instead that C engulfs a different virus V_2, for which the lymphocyte L_1 cannot activate any immune replay.

$$C = !x(w).\mathsf{expose}(u, \{w\}).\overline{u}\langle r\rangle|\ C_1|\ V_2 \quad V_2 = \overline{x}\langle a_2\rangle.V_2^{res} \quad L_1 = z(y).L_1^{act}$$

In this case, the analysis would be:

$$\begin{array}{ll}
\iota(*) \ni \mu_C, \mu_{L_1} & \\
\iota(\mu_C) \ni x, u & \iota(\mu_{L_1}) \ni z \\
\left\{\begin{array}{l}\epsilon(\mu_C)(x) \ni \{v_1, ..., v_n\} \\ \epsilon(\mu_C)(u) \ni \{a_2\},\end{array}\right. & \epsilon(\mu_{L_1})(z) \ni \{a_1, a'_1\} \\
\left\{\begin{array}{l}\kappa(\mu_C)(u) \ni r, \\ \kappa(\mu_C)(x) \ni a_2,\end{array}\right. & \kappa(\mu_{L_1})(z) = \emptyset \\
\rho(w) \ni a_2 & \rho(y) = \emptyset
\end{array}$$

reflecting the fact that the two boxes *cannot* communicate. Indeed, we have that

$u \in \iota(\mu_C)$, $z \in \iota(\mu_{L_1})$, but $\forall G_C \in \epsilon(\mu_C)(u), G_{L1} \in \epsilon(\mu_{L1})(z)$: $G_C \cap G_{L1} = \emptyset$ and therefore $comp(\epsilon(\mu_C)(u), \epsilon(\mu_{L1})(z))$ is false.

6 Static Biological Compartments

Nesting of boxes is not allowed in Beta-binders, even though, as seen in Section 4, it is possible to model a form of virtual nesting. In fact, the affinity or compatibility between types can be used to implicitly model that boxes can be grouped in compartments: boxes are in the same compartment when their beta binders are compatible. However, virtual nesting can be ambiguous and, in addition, movements across compartments require sequences of suitable interface operations. Nevertheless, representing complex hierarchies could be useful to model compartments of biological systems. For this reason in [9] the calculus has been extended with a notion of *static compartments*. The static hierarchical structure of a system is seen as a tree and the compartments as nodes. The original beta-processes of the language are statically enriched with labels representing the positions of the compartments (in which the beta-processes reside) inside the tree structure. Labels are sequences of natural numbers a là Dewey. For instance, the compartments in Figure 1 are

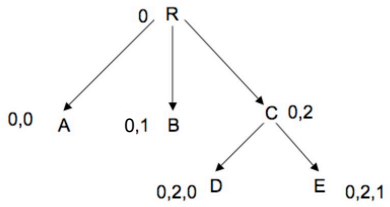


Fig. 1. The Tree Representation of the Hierarchical Structure of a System

identified as follows:

$$R \to 0 \;\; A \to 0,0 \;\; B \to 0,1 \;\; C \to 0,2 \;\; D \to 0,2,0 \;\; R \to 0,2,1$$

Moreover, components can be either internal to compartments or reside on compartment borders. Movement across compartments is requested by internal components and mediated by the border ones. The new syntax, where $n \in I\!N$, is as follows:

$$\mathcal{B} ::= Nil|\ \mathbf{B}[P]^{\mu}_{\lambda}|\ B||B \qquad \text{with} \qquad \begin{cases} \lambda ::= c; s\ (\lambda \in \Lambda) \\ c ::= n|\ c, n \\ s \in \{i, b\} \end{cases}$$

$$\mathbf{P} ::= nil|\ P|P|\ !P|\ \overline{x}\langle y\rangle.P|\ x(y).P|\ \mathsf{expose}(x, \Gamma).P|\ \mathsf{hide}(x).P|\ \mathsf{unhide}(x).P|$$
$$\mathsf{move}(x).P|\ \mathsf{in}(x).P|\ \mathsf{out}(x).P$$

Note that in this version of the calculus there is no restriction. In [9], the notion of compatibility between types is made finer, by introducing a more general notion of affinity, that gives a measure of how much favourable a biological interaction is.

Even though, it is possible to cable this notion inside our analysis, for simplicity we keep the notion used in the previous sections.

Each beta-process is enriched with annotation λ, composed by a label c[2] identifying the corresponding compartment, and a marker s representing the component type (internal i or border one b). In [9] is exploited an extension of Beta-binders' reduction semantics in the style of [6]. In this approach, transitions carry rich labels that allow for retrieving many qualitative aspects of computation. Some locality relations can be introduced, based, in particular, on labels ϑ that uniquely identify the locations of beta-processes. We can use in their place our labels μ, that are uniquely associated with beta-processes. Therefore instead of having $\vartheta\mathbf{B}[P]^c_s$ we have $\mathbf{B}[P]^{\mu}_{c;s}$.

An i-component can move across a compartment border only if mediated by a b-component residing on that border. Movements are rendered, exploiting affinity, by pairs of the following new complementary prefixes, where $x \in \mathcal{N}$. A beta-process can move into a sub-compartment, by firing a move synchronising with the in action, performed by the component residing on the border. A beta-process can move out of a sub-compartment, by firing a move synchronising with the out action, performed by the component residing on the border. Pi-processes are extended with these new movement prefixes. Each transition is labelled by a pair $\phi = \langle \theta, c \rangle$, where c is the compartment identifier, and θ is defined as

$$\theta ::= \langle \mu\, \overline{x}\langle z\rangle, \mu\, x(w)\rangle |\ \langle \mu\, \underline{\overline{y}\langle z\rangle}, \mu \underline{x(w)}\rangle |\ \mu\, a |\ \langle \mu\, \mathsf{in}(x), \mu'\, \mathsf{move}(y)\rangle |\ \langle \mu\, \mathsf{out}(x), \mu'\, \mathsf{move}(y)\rangle$$

where $a ::= \mathsf{expose}(x, \Gamma) |\ \mathsf{hide}(x) |\ \mathsf{unhide}(x)$. Labels annotate transitions in the expected way, recording beta-process identifiers, action prefixes and compartments. Prefixes of intra-communications $(\overline{x}\langle z\rangle, x(w))$ are syntactically distinguished from the inter-communication ones $(\underline{\overline{y}\langle z\rangle}, \underline{x(w)})$. Two beta-processes that reside in different compartments can interact (inter-communication or movement) if one is in a sub-compartment of the other: the label c included in the corresponding ϕ is the one identifying the sub-compartment. For the sake of brevity, we just focus on the

[2] we use c instead of κ, as in [9], because κ is one of our analysis components.

rules for boxes and movements.

$$(In)$$
$$\frac{P \equiv \mathsf{in}(x).P_1|\ P_2 \qquad Q \equiv \mathsf{move}(y).Q_1|\ Q_2}{X \xrightarrow{\langle \mu_P\ \mathsf{in}(x), \mu_Q\ \mathsf{move}(y); c\rangle} Y}$$
$$X = \beta(x:\Gamma)\mathbf{B}^*_{\mathbf{1}}[P]^{\mu_P}_{c,n;b} || \beta(y:\Delta)\mathbf{B}^*_{\mathbf{2}}[Q]^{\mu_Q}_{c;i}$$
$$Y = \beta(x:\Gamma)\mathbf{B}^*_{\mathbf{1}}[P_1|\ P_2]^{\mu_P}_{c,n;b} || \beta(y:\Delta)\mathbf{B}^*_{\mathbf{2}}[Q_1|\ Q_2]^{\mu_Q}_{c,n;i}$$
$$\text{provided that } \Gamma \cap \Delta \neq \emptyset$$

$$(Out)$$
$$\frac{P \equiv \mathsf{out}(x).P_1|\ P_2 \qquad Q \equiv \mathsf{move}(y).Q_1|\ Q_2}{X \xrightarrow{\langle \mu_P\ \mathsf{out}(x), \mu_Q\ \mathsf{move}(y); c, n\rangle} Y}$$
$$X = \beta(x:\Gamma)\mathbf{B}^*_{\mathbf{1}}[P]^{\mu_P}_{c,n;b} || \beta(y:\Delta)\mathbf{B}^*_{\mathbf{2}}[Q]^{\mu_Q}_{c,n;i}$$
$$Y = \beta(x:\Gamma)\mathbf{B}^*_{\mathbf{1}}[P_1|\ P_2]^{\mu_P}_{c,n;b} || \beta(y:\Delta)\mathbf{B}^*_{\mathbf{2}}[Q_1|\ Q_2]^{\mu_Q}_{c;i}$$
$$\text{provided that } \Gamma \cap \Delta \neq \emptyset$$

Our Control Flow Analysis can be extended accordingly. We need a further component $\sigma : \mathbf{Box} \to \Lambda$, the *compartment repository*, that tracks for each beta-process the associated annotation: if $(c; s) \in \sigma(\mu)$ then the beta-process identified by μ can be included in the static compartment c and is an internal component (a border one), if $s = i$ ($s = b$, resp.). This component is checked in analysing $[P]^{\mu}_{c;s}$, and in analysing the move action. Furthermore, we enlarge the co-domain of ι: $\iota : \{*\} \cup \mathbf{Box} \to \wp(\mathbf{Box}) \uplus (\mathbf{N} \cup \mathbf{N^h}) \uplus BindOp$, where $BindOp = \{\mathsf{expose}(x), \mathsf{hide}(x), \mathsf{unhide}(x), \mathsf{in}(x), \mathsf{out}(x), \mathsf{move}(x) |\ x \in \mathbf{N}\}$. If $in(a) \in \iota(\mu)$ then an occurrence of $\mathsf{in}(a)$ may occur in the beta-process labelled μ.

$$(\iota, \sigma, \epsilon, \rho, \kappa) \models^* [P]^{\mu}_{c;s} \quad \text{iff } \mu \in \iota(*) \ \wedge\ (c; s) \in \sigma(\mu) \ \wedge\ (\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} P$$

$$(\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} \mathsf{in}(x).P \quad \text{iff } \forall a \in \rho(x) : in(a) \in \iota(\mu) \ \wedge\ (\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} P$$

$$(\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} \mathsf{out}(x).P \quad \text{iff } \forall a \in \rho(x) : out(a) \in \iota(\mu) \ \wedge\ (\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} P$$

$$\begin{aligned}(\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} \mathsf{move}(x).P \ \text{iff } & \forall a \in \rho(x) : a \in \iota(\mu), \forall \mu' \in \iota(*) : b \in \iota(\mu'), \\ & [((c', m; i) \in \sigma(\mu) \ \wedge\ (c', m, n; i) \in \sigma(\mu') \wedge \\ & in(b) \in \iota(\mu') \ \wedge\ comp(\epsilon(\mu)(a), \epsilon(\mu')(b)) \Rightarrow \\ & (c', m, n; i) \in \sigma(\mu)] \wedge \\ & [((c', m; i) \in \sigma(\mu) \ \wedge\ (c', m; i) \in \sigma(\mu') \wedge \\ & out(b) \in \iota(\mu') \ \wedge\ (\epsilon(\mu)(a) \cap \epsilon(\mu')(b) \neq \emptyset)) \Rightarrow \\ & (c'; i) \in \sigma(\mu)] \ \wedge\ (\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu} P\end{aligned}$$

The clause for in (out) (see above) demands that for all the possible values a of x,

$in(a)$ ($out(a)$, resp.) belongs to $\iota(\mu)$. The rule for move is more structured, because it takes into account both the synchronisation with in and with out actions. The first conjunct requires that if exists a beta-process residing in a sub-compartment of the compartment of the analysed process in which a in action may occur, if the corresponding beta binders are compatible and possibly active, then the process can move inside and change its label accordingly. Similarly, the second conjunct requires that if exists a beta-process residing in a compartment surrounding the compartment of the analysed process in which a out action may occur, if the corresponding binders are compatible and possibly active, then the process can move outside and change its label accordingly. Suppose to have $S = \mathbf{B_P}\beta(x : \Gamma)[\mathsf{move}(x).P]^{\mu_P}_{0;i} || \mathbf{B_Q}\beta(y : \Delta)[\mathsf{in}(y).Q]^{\mu_Q}_{0,0;b}$. Dynamically, the following transition is possible, provided that $\Gamma \cap \Delta \neq \emptyset$: $S \rightarrow \mathbf{B_P}\beta(x : \Gamma)[P]^{\mu_P}_{0,0;i} || \mathbf{B_Q}\beta(y : \Delta)[Q]^{\mu_Q}_{0,0;b}$. Statically, we have that $x \in \iota(\mu_P)$, $in(y) \in \iota(\mu_Q)$, $comp(\epsilon(\mu_P)(x), \epsilon(\mu_Q)(y))$, $(0; i) \in \sigma(\mu_P)$, $(0, 0; b) \in \sigma(\mu_Q)$. Therefore, we also have that $(0, 0; i) \in \sigma(\mu_P)$.

Compartments are seen as *static* localities, because they represent the sites at which events occur and because they not change during the dynamic execution. Here, we only consider some of the locality relations reported in [9], where the function *compart* applied to a label ϕ, returns the compartment identifier, i.e. $compart(\langle \theta, c \rangle) = c$.

Definition 6.1 Given a computation $B_0 \stackrel{\phi_0}{\rightarrow} B_1 \stackrel{\phi_1}{\rightarrow} B_2 ... \stackrel{\phi_n}{\rightarrow} B_n$, ϕ_n has a

- *same compartment dependency* on ϕ_h if $h < n \wedge compart(\phi_h) = compart(\phi_n) = c$.
- *father-son dependency* on ϕ_h if $h < n \wedge compart(\phi_h), m = compart(\phi_n)$.
- *son-father dependency* on ϕ_h if $h < n \wedge compart(\phi_n), m = compart(\phi_h)$.

It is not difficult to imagine the generalisations of the last two relations, obtained by their relative transitive closures.

In order to obtain their static counterparts, we need to enrich our analysis, reported in Table 3 (where only the interesting clauses are included), with a further component ψ decorating the $\models$ symbol in the new judgement $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ and $(\iota, \sigma, \epsilon, \rho, \kappa) \models^\mu_\psi P$. This new component records for each possible communication or interface operation the corresponding static labels that include the prefixes used, the beta-process identifier and the set of all the possible compartments in which the beta-process can reside, according to the analysis. They represent a sort of static approximation of the corresponding dynamic labels $\phi = \langle \theta, c \rangle$. The new component called *label repository* is defined as $\psi \in \wp(\mathcal{C})$, where $C \in \mathcal{C}$ is defined as follows, where $a ::= \mathsf{expose}(x, \Gamma) | \ \mathsf{hide}(x) | \ \mathsf{unhide}(x)$.

$$
\begin{aligned}
C ::= \ & \langle \mu \ \overline{a}\langle d \rangle @ \sigma(\mu), \mu \ a(y) @ \sigma(\mu) \rangle | \ \langle \mu \ \underline{\overline{b}\langle d \rangle} @ \sigma(\mu), \mu' \ \underline{a(y)} @ \sigma(\mu') \rangle | \\
& \langle \mu \ a @ \sigma(\mu) \rangle | \ \langle \mu \ \mathsf{in}(b) @ (c', m, n; i), \mu' \ \mathsf{move}(a) @ (c', m; i) \rangle | \\
& \langle \mu \ \mathsf{out}(b) @ (c', m; i), \mu' \ \mathsf{move}(a) @ (c', m; i) \rangle
\end{aligned}
$$

Note that in order to obtain the static counterpart of the relations presented above, we could also have dealt with simpler labels, both dynamic and static, carrying only

$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{*}_{\psi} [P]^{\mu}_{c;s}$	iff $\mu \in \iota(*) \ \wedge \ (c;s) \in \sigma(\mu) \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \overline{x}\langle y\rangle.P$	iff $\forall a \in \rho(x) : \rho(y) \subseteq \kappa(\mu)(a) \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} x(y).P$	iff $\forall a \in \rho(x) : \forall d \in \kappa(\mu)(a) : d \in \rho(y) \wedge$ $\langle \mu \ \overline{a}\langle d\rangle@\sigma(\mu), \mu \ a(y)@\sigma(\mu)\rangle \in \psi \wedge$ $\forall a \in \rho(x) : a \in \iota(\mu), \forall \mu' \in \iota(*) : \ b \in \iota(\mu'),$ $comp(\epsilon(\mu)(a), \epsilon(\mu')(b)) \Rightarrow \forall d \in \kappa(\mu')(b) : \ d \in \rho(y) \wedge$ $\langle \mu' \ \underline{\overline{b}\langle d\rangle}@\sigma(\mu'), \mu \ \underline{a(y)}@\sigma(\mu)\rangle \in \psi \wedge$ $(\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \mathsf{expose}(x,\Gamma).P$	iff $x, expose(x) \in \iota(\mu) \ \wedge \ x \in \rho(x) \ \wedge \ \rho(\Gamma) \in \epsilon(\mu)(x) \wedge$ $\langle \mu \ \mathsf{expose}(x,\Gamma)@\sigma(\mu)\rangle \in \psi \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \mathsf{hide}(x).P$	iff $\forall a \in \rho(x) : \ a^h, hide(a) \in \iota(\mu) \wedge$ $\langle \mu \ \mathsf{hide}(a)@\sigma(\mu)\rangle \in \psi \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \mathsf{unhide}(x).P$	iff $\forall a \in \rho(x) : \ a, unhide(a) \in \iota(\mu) \wedge$ $\langle \mu \ \mathsf{unhide}(a)@\sigma(\mu)\rangle \in \psi \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \mathsf{in}(x).P$	iff $\forall a \in \rho(x) : in(a) \in \iota(\mu) \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \mathsf{out}(x).P$	iff $\forall a \in \rho(x) : out(a) \in \iota(\mu) \ \wedge \ (\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$
$(\iota,\sigma,\epsilon,\rho,\kappa)$	$\models^{\mu}_{\psi} \mathsf{move}(x).P$	iff $\forall a \in \rho(x) : a \in \iota(\mu), \forall \mu' \in \iota(*) : b \in \iota(\mu'),$ $[((c',m;i) \in \sigma(\mu) \ \wedge \ (c',m,n;i) \in \sigma(\mu') \wedge$ $in(b) \in \iota(\mu') \ \wedge \ (\epsilon(\mu)(a) \cap \epsilon(\mu')(b) \neq \emptyset)) \Rightarrow$ $(c',m,n;i) \in \sigma(\mu)] \wedge$ $\langle \mu' \ \mathsf{in}(b)@(c',m,n;i), \mu \ \mathsf{move}(a)@(c',m;i)\rangle \in \psi \wedge$ $[((c',m;i) \in \sigma(\mu) \ \wedge \ (c',m;i) \in \sigma(\mu') \wedge$ $out(b) \in \iota(\mu') \ \wedge \ comp(\epsilon(\mu)(a), \epsilon(\mu')(b)) \Rightarrow$ $(c';i) \in \sigma(\mu)] \wedge$ $\langle \mu' \ \mathsf{out}(b)@(c',m;i), \mu \ \mathsf{move}(a)@(c',m;i)\rangle \in \psi \wedge$ $(\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$

Table 3
Analysis for Extended Beta-processes: $(\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{*}_{\psi} B$ and for Pi-processes: $(\iota,\sigma,\epsilon,\rho,\kappa) \ \models^{\mu}_{\psi} P$.

information on the compartment. Nevertheless, the complete labels allows for an immediate extension to other relations, such as the ones in [9] that are not reported here, or the ones based on causality, like the ones in [6].

The correspondence with the dynamic labels is easy to establish as stated by the following theorem, that states that for each transition labelled ϕ, there is a corresponding label c in ψ.

Theorem 6.2 ***(Label Correspondence)***
If $B \xrightarrow{\phi} B'$ *and* $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ *then*

- *if* $\phi = \langle\langle \mu\ \overline{x}\langle z\rangle, \mu\ x(w)\rangle; c\rangle$ *then* $\langle \mu\ \overline{a}\langle d\rangle @\sigma(\mu), \mu\ a(y)@\sigma(\mu)\rangle \in \psi$, *with* $(c; s) \in \sigma(\mu)$;
- *if* $\phi = \langle\langle \mu_0\ \underline{\overline{y}\langle z\rangle}, \mu_1\ \underline{x(w)}\rangle; c\rangle$ *then* $\langle \mu_0\ \underline{\overline{b}\langle d\rangle}@\sigma(\mu_0), \mu_1\ \underline{a(y)}@\sigma(\mu_1)\rangle \in \psi$, *with* $(c_0; s) \in \sigma(\mu_0), (c_1; s) \in \sigma(\mu_1)$, *and* $c_l = c_{1-l}$ *or* $c_l = c_{1-l}, n$ *and* $c = c_l$;
- *if* $\phi = \langle \mu\ a; c\rangle$ *then* $\langle \mu\ a@\sigma(\mu)\rangle \in \psi$, *with* $(c; s) \in \sigma(\mu)$;
- *if* $\phi = \langle\langle \mu\ \mathsf{in}(x), \mu'\ \mathsf{move}(y)\rangle; c, n\rangle$ *then* $\langle \mu\ \mathsf{in}(b)@(c, n; i), \mu'\ \mathsf{move}(a)@(c; i)\rangle \in \psi$;
- *if* $\phi = \langle\langle \mu\ \mathsf{out}(x), \mu'\ \mathsf{move}(y)\rangle c, n\rangle$ *then* $\langle \mu\ \mathsf{out}(b)@(c, n; i), \mu'\ \mathsf{move}(a)@(c, n; i)\rangle \in \psi$.

The subject reduction result still holds in the new framework, i.e. our extended analysis is semantically correct with respect to the extended semantics.

Theorem 6.3 ***(Subject reduction)***
If $B \xrightarrow{\phi} B'$ *and* $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ *then also* $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B'$.

7 Example: the $cAMP$-signaling Pathway in Olfactory Sensory Neurons

We use here the example presented in [9], i.e. the $cAMP$-signaling pathway in olfactory sensory neurons ($OSNs$). The pathway [1] represents the way the G protein-coupled receptors indirectly modulate the activity of ion channels via the action of second messengers. An odorant ligand O can bind an odorant receptor OR, activating it. The active OR stimulates the G-protein $GDP\alpha\beta\gamma$, causing the dissociation of the trimer in two active subunits $GTP\alpha$ and $GTP\beta\gamma$. Afterwards, $GTP\alpha$ can either hydrolyse, returning $GDP\alpha$, or activate the adenylyl cyclase (AC), his target protein. In the first case, the subunit $GDP\alpha$ associates again with the subunit $GDP\beta\gamma$. In the second case, instead, the activation of AC produces, through the synthesis of ATP, an increase in concentration of the second messenger $cAMP$. A $cAMP$ molecule can open, through a reversible binding, the ion channel IC, allowing Na^+ and Ca^{2+} molecules to enter. The hydrolysis of GTP to GDP causes $GTP\alpha$ to dissociate from AC and associate again with $GTP\beta\gamma$. The Beta-binder specification of the presented pathway is showed in the upper part of Table 4. It is slightly different from the one given in [9], because we do not use the *join* and *split* operations. In order to synchronise the hide and unhide operations in the beta-processes B_G, B_{AC}, and B_{ATP} we resort to a communication protocol, based on I/O actions on the beta binders o_G, c_G in B_G and o_{AC}, c_{AC} in B_{AC}. The most meaningful analysis results are reported in the lower part of Table 4. They show that the analysis gives a correct approximation of the communications and movements of the

$B = B_O||B_{OR}||B_G||B_{AC}||B_{ATP}||B_{IC}||B_{Na^+}||B_{Ca^{2+}}$

$$
\begin{array}{lcl}
B_O & = & \beta(x_0 : \Delta_0)[O]^{\mu_O}_{0;i} \\
B_{OR} & = & \beta(x'_0 : \Delta_0)\beta^h(y_1 : \Delta_1)[OR|\ A]^{\mu_{OR}}_{0,0;b} \\
B_G & = & \beta(x_1 : \Delta_1)\beta^h(y''_2 : \Delta_2) \\
 & & \beta(o_G : \Delta_7)\beta(c_G : \Delta_8)[GDP|P_\alpha|P_{\beta\gamma}]^{\mu_G}_{0,0;b} \\
 & & \\
B_{AC} & = & \beta^h(y'_2 : \Delta_2)\beta(o_{AC} : \Delta_7)\beta(c_{AC} : \Delta_8)[AC]^{\mu_{AC}}_{0,0;b} \\
B_{ATP} & = & \beta(x_3 : \Delta_3)\beta(y''_2 : \Delta_2)[\overline{y''_2}\langle z_{ATP}\rangle.cAMP]^{\mu_{ATP}}_{0,0;i} \\
B_{IC} & = & \beta(x'_3 : \Delta_3)\beta^h(y_4 : \Delta_4)[IC|M]^{\mu_{IC}}_{0,0;b} \\
B_{Na^+} & = & \beta(x_5 : \Delta_5)[Na^+]^{\mu_{Na^+}}_{0;i} \\
B_{Ca^{2+}} & = & \beta(x_6 : \Delta_6)[Ca^{2+}]^{\mu_{Ca^{2+}}}_{0;i}
\end{array}
$$

$$
\begin{array}{lcl}
O & = & \overline{x_0}\langle z_O\rangle.x_0(w_O).O \\
OR & = & x'_0(w_{OR}).\mathsf{unhide}(y_1).\mathsf{hide}(y_1).\overline{x'_0}\langle w_{OR}\rangle.OR \\
A & = & \overline{y_1}\langle z_A\rangle.A \\
GDP & = & x_1(w_{GDP}).\mathsf{hide}(x_1).GTP \\
GTP & = & \mathsf{unhide}(y''_2).\overline{o_G}\langle j_{AC}\rangle.y_2(w_{GTP}).\mathsf{unhide}(x_1). \\
 & & \overline{c_G}\langle s_{AC}\rangle.\mathsf{hide}(y'_2).GDP \\
P_\alpha & = & \overline{y_2}\langle z_\alpha\rangle.P_\alpha \\
AC & = & o_{AC}(z_{AC}).\mathsf{unhide}(y'_2).y'_2(w_{AC}). \\
 & & c_{AC}(y_{AC}).\mathsf{hide}(y'_2).AC \\
cAMP & = & \overline{x_3}\langle z_{cAMP}\rangle.x_3(w_{cAMP}).cAMP \\
IC & = & x'_3(w_{IC}).\mathsf{unhide}(y_4). \\
 & & \mathsf{hide}(y_4).\overline{x'_3}\langle z_{IC}\rangle.IC \\
M & = & \mathsf{in}(y_4).M \\
Na^+ & = & \mathsf{move}(x_5).Na^+ \\
Ca^{2+} & = & \mathsf{move}(x_6).Ca^{2+}
\end{array}
$$

$\Delta_i \cap \Delta_j \neq \emptyset$ when $i = j$ or $i = 4$ and $j = 5, 6$

$\iota(*) \ni \mu_O, \mu_{OR}, \mu_G, \mu_{AC}, \mu_{ATP}, \mu_{IC}, \mu_{Na^+}, \mu_{Ca^{2+}}$
$\iota(\mu_O) \ni x_0; \iota(\mu_{OR}) \ni x'_0, y^h_1, y^u_1; \iota(\mu_G) \ni x_1; \iota(\mu_{AC}) \ni {y'_2}^h; \iota(\mu_{ATP}) \ni x_3, y''_3;$
$\iota(\mu_{IC}) \ni x'_3, y^h_4, y_4, in(y); \iota(\mu_{Na^+}) \ni x_5; \iota(\mu_{Ca^{2+}}) \ni x_6;$
$\iota(\mu_{G1}) \ni x^h_1; \iota(\mu_{G2}) \ni x^h_1; \iota(\mu_{GAC}) \ni x^h_1, y'_2;$
$(0; i) \in \sigma(\mu_O), \sigma(\mu_{Na^+}), \sigma(\mu_{Ca^{2+}}), (0, 0; i) \in \sigma(B_{ATP});$
$(0, 0; b) \in \sigma(B_{OR}), \sigma(B_G), \sigma(B_{AC}), \sigma(B_{IC}), \sigma(B_{G1}), \sigma(B_{G2}), \sigma(B_{GAC}), \sigma(\mu_{Na^+}), \sigma(\mu_{Ca^{2+}});$
$\kappa(\mu_O)(x_0) \ni z_O; \rho(w_0) \ni z_{OR};$
$\kappa(\mu_{OR})(x'_0) \ni z_{OR}; \rho(w_{OR}) \ni z_O; \kappa(\mu_{OR})(y_1) \ni z_A;$
$\rho(w_{GDP}) \ni z_A; \kappa(\mu_G)(y_2) \ni z_\alpha; \rho(w_{GTP}) \ni z_\alpha; \rho(w_{AC}) = \emptyset;$
$\kappa(o_G) \ni j_{AC}, \kappa(c_G) \ni s_{AC}, \rho(o_{AC}) \ni j_{AC}, \rho(c_{AC}) \ni s_{AC};$
$\kappa(\mu_{ATP})(y''_2) \ni z_{ATP}; \kappa(\mu_{ATP})(x_3) \ni z_{cAMP}; \rho(w_{cAMP}) \ni z_{IC};$
$\kappa(\mu_{IC})(x'_3) \ni z_{IC}; \rho(w_{IC}) \ni z_{cAMP} \rho(w_{GTP}) \ni z_\alpha; \rho(w_{GDP}) = \emptyset; \kappa(\mu_{G1})(y_2) \ni z_\alpha;$
$\rho(w_{GTP}) \ni z_\alpha, z_{ATP}; \rho(w_{GDP}) = \emptyset; \kappa(\mu_{G1})(y_2) \ni z_\alpha; \rho(w_{AC}) \ni z_\alpha, z_{ATP}$

Table 4
Specification of the $cAMP$-Signaling Pathway in OSNs (upper part) and some entries of the analysis (lower part)

specified system. Note, for instance, that $(0, 0; b) \in \sigma(\mu_{Na^+}), \sigma(\mu_{Ca^{2+}})$ correspond to the possible movements of B_{Na^+} and $B_{Ca^{2+}}$ in the compartment identified by $0, 0$. Let us analyse part of the computation, also presented in [9], where the ion-channel IC is activated and causes the entrance of a Ca^{2+} molecule. For the sake of simplicity, below we only report the transitions labels ϕ_i, in the left-hand side together with the corresponding static labels C_i recorded in ψ, in the right-hand side.

...

$\phi_1 = \langle \mu_{ATP}\ \underline{\overline{y''_2}\langle z_{ATP}\rangle}, \mu_{AC}\ \underline{y'_2(w_{AC})}; \underline{0, 0}\rangle$ $C_1 = \langle \mu_{ATP}\ \underline{\overline{y''_2}\langle z_{ATP}\rangle}@\{(\underline{0, 0}; i)\}, \mu_{AC}\ \underline{y'_2(w_{AC})}@\{(\underline{0, 0}; b)\}\rangle$

$\phi_2 = \langle \mu_{ATP}\ \underline{\overline{x_3}\langle z_{cAMP}\rangle}, \mu_{IC}\ \underline{x'_3(w_{IC})}; 0, 0\rangle$ $C_2 = \langle \mu_{ATP}\ \underline{\overline{x_3}\langle z_{cAMP}\rangle}@\{(0, 0; i)\}, \mu_{IC}\ \underline{x'_3(w_{IC})}@\{(0, 0; b)\}\rangle$

$\phi_3 = \langle \mu_{IC}\ \mathsf{unhide}(y_4); 0, 0\rangle$ $C_3 = \langle \mu_{IC}\ \mathsf{unhide}(y_4)@\{(0, 0; b)\}\rangle$

$\phi_4 = \langle \mu_{IC}\ \mathsf{in}(y_4), \mu_{Ca^{2+}}\ \mathsf{move}(x_6); \underline{0, 0}\rangle$ $C_4 = \langle \mu_{IC}\ \mathsf{in}(y_4)@(\underline{0, 0}; b), \mu_{Ca^{2+}}\ \mathsf{move}(x_6)@(0; i)\rangle$

By analysing this computation, we can observe that there is a *same compartment dependency* between the transition labelled ϕ_1 and the one labelled ϕ_4, because $1 < 4$ and $compart(\phi_1) = compart(\phi_4) = 0, 0$. We can observe that the corre-

sponding static labels present the same dependency. Biologically, this dependency corresponds to the relation between the entrance of a Ca^{2+} molecule and the activation of the target protein AC.

8 Conclusions and Future Work

We have presented a Control Flow Analysis for Beta-binders, able to describe the essential behaviour of each of its boxes, in terms of possible interactions. When applied to the version of Beta-binders, modelling biological compartments, the analysis has been extended accordingly. To this aim, it is also able to approximate the possible movements across compartments. Furthermore, the analysis can offer a set of approximations of all the possible actions the beta-process under analysis can perform, with indication on the possible boxes and the compartments involved.

Simply exploiting the soundness of our analysis, in the first part of the work, we have proved some basic facts, that can be immediately used to establish simple properties, such as the absence of interaction of two boxes or the isolation of a box. In the second part, we have addressed, for brevity, only part of the locality relations reported in [9]. Our approach can be easily extended to the other relations in [9] and also to the causality-based relations, like the ones in [6] for the $\pi-$calculus.

Both results are useful for illustrating how suitable static techniques can be adapted to model biological systems, giving some insights on their behaviour.

In classical process algebras, communications are modelled in a *key-lock* style, by requiring that an input and an output can communicate only if they synchronize on the same channel. In Beta-binders the *key-lock* model for interaction is partially relaxed. Interactions between boxes are allowed when the types of binders show some form of compatibility. This is a nice feature, when formalising biological behaviour, maybe collecting part of specifications in different databases, because it allows to easily put together the needed components. It is sufficient to put them in different boxes, just establishing the proper binders. Under this regard, code re-use is easier than in the π-calculus. It would be interesting to exploit this is feature also from a static analysis point of view, looking for suitable ways of composing independent analyses of parts of systems.

In perspective, static analysis can be fruitfully exploited to study dynamic properties of large biological systems, by keeping the computational costs low. Furthermore, it can be used to reason about the model chosen for describing the biological system under consideration, by checking the properties on the model and by comparing the obtained results with the experimental ones reported in the literature.

Acknowledgments. We are grateful to Pierpaolo Degano and Linda Brodo for their helpful discussions and comments.

References

[1] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts and Peter Walter. Molecular Biology of the Cell. Garland Science, 2002.

[2] Chiara Bodei. A static analysis for Beta-Binders. *Proceedings of From Biology To Concurrency and back Workshop (FBTC'07).* Electronic Notes in Theoretical Computer Science 194(3): 69-85, 2008.

[3] Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static analysis for the π-calculus with their application to security. *Information and Computation* (165): 68-92, 2001.

[4] Luca Cardelli. Brane Calculi - Interactions of Biological Membranes. *Proceedings of Computational Methods in Systems Biology (CMSB'04)*. Lecture Notes in Computer Science 3082, pp. 257-278, Springer, 2005.

[5] Luca Cardelli and Andrew D. Gordon. Mobile Ambients. *Theoretical Computer Science* 240(1): 177-213 (2000).

[6] Michele Curti, Pierpaolo Degano, Corrado Priami, Cosima Baldari. Modeling biochemical pathways through enhanced pi-calculus. *Theoretical Computer Science*325:111–140, 2004,

[7] Vincent Danos, Jean Krivine. Transactions in RCCS. *Proceedings of Conference on Concurrency Theory (CONCUR'05)*. Lecture Notes in Computer Science 3653, pp. 398-412, Springer 2005.

[8] Vincent Danos, Cosimo Laneve. Graphs for Core Molecular Biology. *Proceedings of Computational Methods in Systems Biology (CMSB'03)*. Lecture Notes in Computer Science 2602, pp. 34-46, Springer 2003.

[9] Maria Luisa Guerriero, Corrado Priami and Alessandro Romanel. Static Biological Compartments with Beta-binders. *Proceedings of Algebraic Biology, Second International Conference (AB'07)*, Lecture Notes in Computer Science 4545, pp. 247–261. Springer, 2007.

[10] Hiroaki Kitano Systems Biology: a brief overview. *Science* 2002, 295(5560):1662–1664.

[11] Robin Milner. Communicating and mobile systems: the π-calculus. Cambridge University Press, 1999.

[12] Hanne Riis Nielson and Flemming Nielson. Flow Logic: a multi-paradigmatic approach to static analysis. *The Essence of Computation: Complexity, Analysis, Transformation*, Lecture Notes in Computer Science 2566, pp. 223-244, Springer, 2002.

[13] Flemming Nielson, Hanne Riis Nielson, and Ren Rydhof Hansen. Validating firewalls using flow logics. *Theoretical Computer Science* 283(2): 381-418 (2002).

[14] Flemming Nielson, Hanne Riis Nielson, Corrado Priami, and Debora Schuch da Rosa. Control Flow Analysis for BioAmbients. *Electronic Notes in Theoretical Computer Science* 180(3): 65-79. 2007.

[15] Flemming Nielson, Hanne Riis Nielson, Debora Schuch da Rosa, and Corrado Priami. *Static analysis for systems biology. Proceedings of workshop on Systeomatics - dynamic biological systems informatics.* Computer Science Press, Trinity College Dublin, 2004.

[16] Flemming Nielson and Helmut Seidl. Control Flow Analysis in cubic time. *Proceedings of Proceedings of European Symposium on Programming (ESOP'01)*, Lecture Notes in Computer Science 2028, pp. 252-268. Springer Verlag, 2001.

[17] Davide Prandi, Corrado Priami and Paola Quaglia. Communicating by compatibility. *Journal of Logic and Algebraic Programming* 75 (2): 167-181, 2008.

[18] Corrado Priami. Computational thinking in Biology. TR-10-2007. Centre for Computational and Systems Biology, University of Trento, 2007.

[19] Corrado Priami and Paola Quaglia. Beta-binders for Biological Interactions. *Proceedings of Proceedings of Computational Methods in Systems Biology (CMSB'04)*, Lecture Notes in Computer Science 3380, pp. 20–33. Springer, 2005.

[20] Corrado Priami and Paola Quaglia. Operational patterns in Beta-binders. *Proceedings of Transactions on Computational Systems Biology*, Lecture Notes in Computer Science 3380, pp. 50-65, Springer, 2005.

[21] Corrado Priami, Aviv Regev, Ehud Y. Shapiro, William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.* 80(1): 25-31 (2001).

[22] Paola Quaglia. On Beta-binders Communications. *Proceedings of Montanari Festschrift.* Lecture Notes in Computer Science 5065, pp. 457-472, Springer, 2008.

[23] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Y. Shapiro. BioAmbients: An abstraction for biological compartments. *Theoretical Computer Science* 325(1): 141-167. 2004, Elsevier.

[24] Aviv Regev, William Silverman, Ehud Y. Shapiro. Representation and Simulation of Biochemical Processes Using the pi-Calculus Process Algebra. *Pacific Symposium on Biocomputing*, pp. 459-470, 2001.

[25] Jeannette M.Wing. Computational thinking. *Communications of the ACM*, 49(3):33-35, 2006.

A Proofs

In this appendix restates the lemmata and theorems presented earlier in the paper and gives the proofs of their correctness.

A.1 Correctness of the Analysis

Lemma A.1 ***(Substitution result)***

(i) $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P$ *and* $v \in \rho(x)$ *imply* $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P\{v/x\}$;

(ii) $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *and* $v \in \rho(x)$ *imply* $(\iota, \epsilon, \rho, \kappa) \models^{*} B\{v/x\}$;

Proof. Both proofs proceed by structural induction and uses the following fact:

$$\forall y : \rho(y(\{v/x\})) \subseteq \rho(y)$$

If indeed $y \neq x$ then $\rho(y(\{v/x\})) = \rho(y)$, while if $y = x$, then $\rho(y(\{v/x\})) = \rho(v)$. Furthermore, $v \in \rho(y)$, because $v \in \rho(x)$ and $x = y$. Therefore $\rho(v) \subseteq \rho(y)$.
[(i)] The proof is by structural induction on P. We consider here only the two most interesting cases.
Case $P \equiv z(w).P'$. We may assume that $w \neq v, x$. Now, $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P$ amounts to checking that $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P'$ and that (1) $\forall a \in \rho(z) : \kappa(\mu)(a) \subseteq \rho(w)$; (2) $\forall a \in \rho(z) : a \in \iota(\mu)$ and $\forall \mu' \in \iota(*) : \ b \in \iota(\mu')$ then $comp(\epsilon(\mu)(a), \epsilon(\mu')(b)) \Rightarrow \kappa(\mu')(b) \subseteq \rho(y)$. By the induction hypothesis and the fact stated above, we have that $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P'\{v/x\}$. Furthermore, since $\rho(z\{v/x\}) \subseteq \rho(z)$, we have that a certain condition holds for $\forall a \in \rho(z)$ then *a fortiori* it holds also for $\forall a \in \rho(z\{v/x\})$. As a consequence: (1) $\forall a \in \rho(z\{v/x\}) : \kappa(\mu)(a) \subseteq \rho(w)$, and (2) $\forall a \in \rho(z\{v/x\}) : a \in \iota(\mu)$ and $\forall \mu' \in \iota(*) : \ b \in \iota(\mu')$ then $comp(\epsilon(\mu)(a), \epsilon(\mu')(b)) \Rightarrow \kappa(\mu')(b) \subseteq \rho(y)$. This is equivalent to the required $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P\{v/x\}$.
Case $P = \mathsf{hide}(z).P'$. Now, $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P$ amounts to checking that $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P'$ and that $\forall a \in \rho(z) : \ a^h \in \iota(\mu)$. As above, since $\rho(z\{v/x\}) \subseteq \rho(z)$, we have that $\forall a \in \rho(z\{v/x\}) : \ a^h \in \iota(\mu)$. For this reason and by the induction hypothesis, we obtain the required $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P\{v/x\}$.
[(ii)] The proof is by structural induction on B. We consider here only one case.
Case $B = \beta(z : \Gamma)\mathbf{B}[P]^{\mu}$. Now $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ is equivalent to $(\iota, \epsilon, \rho, \kappa) \models^{*} B\{v/x\}$, because $z\{v/x\} = z$.

□

Lemma A.2 ***(Invariance of Structural Congruence)***

(i) *If* $P \equiv Q$ *and* $(\iota, \epsilon, \rho, \kappa) \models^{\mu} P$ *then* $(\iota, \epsilon, \rho, \kappa) \models^{\mu} Q$

(ii) *If* $B \equiv B'$ *and* $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *then* $(\iota, \epsilon, \rho, \kappa) \models^{*} B'$

Proof. The proof amounts to a straightforward inspection of each of the clauses defining the structural congruence clauses. □

Theorem A.3 ***(Subject reduction)***
If $B \rightarrow B'$ *and* $(\iota, \epsilon, \rho, \kappa) \models^{*} B$ *then also* $(\iota, \epsilon, \rho, \kappa) \models^{*} B'$.

Proof. By induction on the inference of $\rightarrow$.
Case (Intra). Let B be $\mathbf{B}[P]^{\mu}$, where $P \equiv (\nu\tilde{u})(x(w).P_1 | \ \overline{x}\langle z\rangle.P_2 | \ P_3)$, and

$B' = \mathbf{B}[(\nu\tilde{u})(P_1\{z/w\}|\ P_2|\ P_3)]^\mu$. We have to prove that $(\iota,\epsilon,\rho,\kappa) \models^* B'$. $(\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}[P]^\mu$, amounts to $\mu \in \iota(*)$, $\forall\hat{\beta}(x:\Gamma)$ in $\mathbf{B}$: $x \in \epsilon(\mu)$ and $\rho(\Gamma) \in \epsilon(\mu)(x)$, and – furthermore – to: (1) $\forall a \in \rho(x) : \rho(z) \subseteq \kappa(\mu)(a) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P_1$, (2$a$) $\forall a \in \rho(x) : \kappa(\mu)(a) \subseteq \rho(w) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P_2$ and (2b) $\forall a \in \rho(z) : (a \in \iota(\mu)$ and $\forall \mu' \in \iota(*) : b \in \iota(\mu')$ then $comp(\epsilon(\mu)(a), \epsilon(\mu')(b))$ implies that $\kappa(\mu')(b) \subseteq \rho(y)$, (3) $\wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P_3$. Since $\rho(x) = \{x\}$ and $\rho(z) = \{z\}$, (1) is equivalent to $z \in \rho(x) \subseteq \kappa(\mu)(x)$ and (2a) to $\kappa(\mu)(x) \subseteq \rho(w)$, and therefore we have that $z \in \rho(w)$. By Lemma A.1 and (1), we have that $(\iota,\epsilon,\rho,\kappa) \models^\mu P_1\{z/w\}$, by (2a) and (3) we have that $(\iota,\epsilon,\rho,\kappa) \models^\mu P_1\{z/w\}|\ P_2|\ P_3$. We can obtain the required $(\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}[(\nu\tilde{u})(P_1\{z/w\}|\ P_2|\ P_3)]^\mu$.

Case (Inter). Let B be $\beta(x:\Gamma)\mathbf{B}^*_\mathbf{1}[P]^{\mu_P}||\beta(y:\Delta)\mathbf{B}^*_\mathbf{2}[Q]^{\mu_Q}$, where $P \equiv (\nu\tilde{u})(x(w).P_1|\ P_2)$, $Q \equiv (\nu\tilde{v})(\overline{y}\langle z\rangle.Q_1|\ Q_2)$, and B' be $\mathbf{B}^*_\mathbf{1}[(\nu\tilde{u})(P_1|\ P_2)]^{\mu_P}||B^*_2[(\nu\tilde{v})(Q_1|\ Q_2)]^{\mu_Q}$, with $\Gamma \cap \Delta \neq \emptyset$ and $x, z \notin \tilde{u}$ and $y, z \notin \tilde{v}$. We have to prove that $(\iota,\epsilon,\rho,\kappa) \models^* B'$. $(\iota,\epsilon,\rho,\kappa) \models^* B$ in particular, implies that $\mu_P, \mu_Q \in \iota(*)$, (1) $x \in \iota(\mu_P)$, $G_P = \rho(\Gamma) \in \epsilon(\mu_P)(x)$ and $(\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}^*_\mathbf{1}[P]^{\mu_P}$; (2) $y \in \iota(\mu_Q)$, $G_Q = \rho(\Delta) \in \epsilon(\mu_Q)(y)$ and $(\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}^*_\mathbf{2}[Q]^{\mu_Q}$; (3) $\forall a \in \rho(x) : a \in \iota(\mu)$ and $\forall \mu' \in \iota(*) : b \in \iota(\mu')$ then since $G_P \cap G_Q \neq \emptyset$, $comp(\epsilon(\mu)(a), \epsilon(\mu)(b))$ implies that $\kappa(\mu')(b) \subseteq \rho(y)$, and $(\iota,\epsilon,\rho,\kappa) \models^\mu P_1$ (4) $\forall a \in \rho(y) : \rho(z) \subseteq \kappa(\mu)(a) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu Q_1$. Since $\rho(x) = \{x\}$ and $\rho(z) = \{z\}$, we have that $a = x \in \iota(\mu)$ and that $z \subseteq \kappa(\mu)(y)$. Condition (3) is verified with $b = y$ and then $\{z\} = \kappa(\mu_Q)(y) \subseteq \rho(w)$, i.e. $z \in \rho(w)$. By Lemma A.1 and the fact that $z \in \rho(w)$, we have that $(\iota,\epsilon,\rho,\kappa) \models^{\mu_P} P_1\{z/w\}$, and also that $(\iota,\epsilon,\rho,\kappa) \models^{\mu_P} (\nu\tilde{u})(P_1|\ P_2)$ and $(\iota,\epsilon,\rho,\kappa) \models^{\mu_Q} (\nu\tilde{v})(Q_1|\ Q_2)$ and finally the required $(\iota,\epsilon,\rho,\kappa) \models^* \mathbf{B}^*_\mathbf{1}[(\nu\tilde{u})(P_1|\ P_2)]^{\mu_P}||B^*_2[(\nu\tilde{v})(Q_1|\ Q_2)]^{\mu_Q}$.

Case (Hide). Let B be $\mathbf{B}^*\beta(x:\Gamma)[P]^\mu$, with $P \equiv (\nu\tilde{u})(\mathsf{hide}(x).P_1|\ P_2)$ and B' be $\mathbf{B}^*\beta^h(x:\Gamma)[(\nu\tilde{u})(P_1|\ P_2)]^\mu$. We have to prove that $(\iota,\epsilon,\rho,\kappa) \models^* B'$. $(\iota,\epsilon,\rho,\kappa) \models^* B$ in particular, implies that $\mu \in \iota(*)$, $x \in \iota(\mu)$, $\rho(\Gamma) \in \epsilon(\mu)(x)$ and that $\forall a \in \rho(x) : a^h \in \iota(\mu) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$. Since x is a value, then $\rho(x) = \{x\}$ and the above amounts to $x^h \in \iota(\mu)$. From this, from $\rho(\Gamma) \in \epsilon(\mu)(x)$ and $(\iota,\epsilon,\rho,\kappa) \models^\mu P$, we can conclude that $(\iota,\epsilon,\rho,\kappa) \models^\mu \mathbf{B}^*\beta^h(x:\Gamma)[(\nu\tilde{u})(P_1|\ P_2)]^\mu$.

Case (Par) follows directly from the induction hypothesis.

Case (Struct) uses Lemma A.2.

□

Theorem A.4 ***(Outputs in κ)***

(i) *If $(\iota,\epsilon,\rho,\kappa) \models^* B$ and $B \to^* B' \to B''$, such that the last transition $B' \to B''$ is derived using the rule (Intra) on the output prefix $\overline{x}\langle z\rangle$ in the box labelled μ, then $z \in \kappa(\mu)(x)$.*

(ii) *If $(\iota,\epsilon,\rho,\kappa) \models^* B$ and $B \to^* B' \to B''$, such that the last transition $B' \to B''$ is derived using the rule (Inter) on the output prefix $\overline{y}\langle z\rangle$ in the box labelled μ, then $z \in \kappa(\mu)(y)$.*

Proof. By induction on the length of the computation. By Theorem A.3, we have that $(\iota,\epsilon,\rho,\kappa) \models^* B'$. Therefore the proof proceeds by induction on the transition rules used to derive $B' \to B''$.

[(i)]

Case (Intra). If this rule is applied, than B' is in the form

$\mathbf{B}'[(\nu\tilde{u})(x(w).P_1|\ \overline{x}\langle z\rangle.P_2|\ P_3)]^{\mu}$. Since $(\iota,\epsilon,\rho,\kappa)\ \models^* B'$, we have that, in particular, $(\iota,\epsilon,\rho,\kappa)\ \models^{\mu}\overline{x}\langle z\rangle.P_2$ and also the required $z\in\kappa(\mu)(x)$, because $\rho(x)=x$ and $\rho(z)=z$.
Cases (Inter), (Expose), (Hide), (Unhide). Transitions that use any of these rules will not use (Intra) and are therefore disregarded.
Cases (Par), (Struct) are straightforward, by applying the induction hypotheses.
[(ii)]
Case (Inter). If this rule is applied, than B' is in the form $\beta(x : \Gamma)\mathbf{B}^*_{\mathbf{1}}[(\nu\tilde{u})(x(w).P_1|\ P_2)]^{\mu_P}||\beta(y : \Delta)\mathbf{B}^*_{\mathbf{2}}[(\nu\tilde{v})(\overline{y}\langle z\rangle.Q_1|\ Q_2)]^{\mu_Q}$. Since $(\iota,\epsilon,\rho,\kappa)\ \models^* B'$, we have that, in particular, $(\iota,\epsilon,\rho,\kappa)\ \models^{\mu_Q}\overline{y}\langle z\rangle.Q_1$ and also the required $z\in\kappa(\mu)(x)$, because $\rho(y)=y$ and $\rho(z)=z$.
Cases (Intra), (Expose), (Hide), (Unhide). Transitions that use any of these rules will not use (Intra) and are therefore disregarded.
Cases (Par), (Struct) are straightforward, by applying the induction hypotheses. □

In a completely similar way can be proved the following theorem, whose proof we skip.

Theorem A.5 ***(Values in** ρ**)*** *If* $(\iota,\epsilon,\rho,\kappa)\ \models^* B$ *and* $B\rightarrow^* B'\rightarrow B''$*, such that* B'' *is either in the form* $\mathbf{B}[P\{v/x\}|P']$ *or in the form* $\mathbf{B_1}[P\{v/x\}|P']||\mathbf{B_2}[Q]$ *then* $v\in\rho(x)$*.*

Theorem A.6 ***(Binders in** ι **and** ϵ**)***
If $(\iota,\epsilon,\rho,\kappa)\ \models^* B$ *and* $B\rightarrow^* B'\rightarrow B''$*, such that the last transition is derived using an interface rule (Expose) or (Hide) or (Unhide) on the binder* x *with type* Γ *and* B'' *includes* $\mathbf{B}[P]^{\mu}$ *and if* $(\hat{x}:\Gamma)$ *occurs in* B'' *then* $\hat{x}\in\iota(\mu)\wedge\rho(\Gamma)\in\epsilon(\mu)$*.*

Proof. By induction on the length of the computation. By Theorem A.3, we have that $(\iota,\epsilon,\rho,\kappa)\ \models^* B'$. Therefore the proof proceeds by induction on the transition rules used to derive $B'\rightarrow B''$.
Cases (Expose), (Hide) and (Unhide) are similar.
Case (Expose). If this rule is applied, than B' is in the form $\mathbf{B}^*[(\nu\tilde{u})(expose(x,\Gamma).P_1|\ P_2)]^{\mu}$. Since $(\iota,\epsilon,\rho,\kappa)\ \models^* B'$ and therefore $(\iota,\epsilon,\rho,\kappa)\ \models^{\mu}(\nu\tilde{u})(expose(x,\Gamma).P_1|\ P_2)$ then, in particular, we have that $x\in\iota(\mu)$ and $\rho(\Gamma)\in\epsilon(\mu)$.

Cases (Expose), (Hide) and (Unhide) are similar.
Cases (Intra) and (Inter). Transitions that use any of these rules will not change the interface and are therefore disregarded.
Cases (Par), (Struct) are straightforward, by applying the induction hypotheses.
□

A.2 Existence of Estimates

Theorem A.7 ***(Moore Family)***
For any beta-process B *the set* $\{(\iota,\epsilon,\rho,\kappa)|\ (\iota,\epsilon,\rho,\kappa)\ \models^* B\}$ *is a Moore family.*

Proof. We proceed by structural induction on B. Let $\mathcal{I}$ a set of proposed estimates

and let J and $(\iota_j, \epsilon_j, \rho_j, \kappa_j)$ such that $\mathcal{J} \subseteq \mathcal{I} = \{(\iota_j, \epsilon_j, \rho_j, \kappa_j) |\ j \in J\}$. Next, define

$$(\iota', \epsilon', \rho', \kappa') = \sqcap \mathcal{J}$$

We have to check that $(\iota', \epsilon', \rho', \kappa') \models^* B$. We just consider one case. The others are similar.
Case $(\beta(x : \Gamma)\mathbf{B}[P]^\mu)$. Since $\forall j \in J : (\iota_j, \epsilon_j, \rho_j, \kappa_j) \models^* \beta(x : \Gamma)\mathbf{B}[P]^\mu$, then

$$\forall j \in J : x \in \iota_j(\mu) \text{ and } \rho_j(\Gamma) \in \epsilon_j(\mu)(x)\ \wedge\ (\iota_j, \epsilon_j, \rho_j, \kappa_j) \models^* \mathbf{B}[P]^\mu$$

Using the induction hypothesis and the fact that ι' and ϵ' are obtained in a pointwise way, we then obtain that

$$x \in \iota'(\mu), \rho(\Gamma) \in \epsilon'(\mu)(x) \text{ and that } (\iota', \epsilon', \rho', \kappa') \models^* \mathbf{B}[P]^\mu$$

thus establishing the required $(\iota', \epsilon', \rho', \kappa') \models^* \beta(x : \Gamma)\mathbf{B}[P]^\mu$. □

Theorem A.8 ***(Existence of Estimates)***
For any beta-process B, there exists an analysis result $(\iota, \epsilon, \rho, \kappa)$ such that $(\iota, \epsilon, \rho, \kappa) \models^* B$.

Proof. The thesis follows from Theorem A.7, because a Moore family is never empty, then there always exists an estimate satisfying the rules in Table 2. □

A.3 Correctness of the Extended Analysis

Theorem A.9 ***(Label Correspondence)***
If $B \xrightarrow{\phi} B'$ and $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ then

- *if $\phi = \langle\langle \mu\ \overline{x}\langle z\rangle, \mu\ x(w)\rangle; c\rangle$ then $\langle \mu\ \overline{a}\langle d\rangle @\sigma(\mu), \mu\ a(y)@\sigma(\mu)\rangle \in \psi$, with $(c; s) \in \sigma(\mu)$;*
- *if $\phi = \langle\langle \mu_0\ \overline{y}\langle z\rangle, \mu_1\ \underline{x(w)}\rangle; c\rangle$ then $\langle \mu_0\ \overline{b}\langle d\rangle @\sigma(\mu_0), \mu_1\ \underline{a(y)}@\sigma(\mu_1)\rangle \in \psi$, with $(c_0; s) \in \sigma(\mu_0), \overline{(c_1; s) \in \sigma(\mu_1)}$, and $c_l = c_{1-l}$ or $c_l = c_{1-l}, n$ and $c = c_l$;*
- *if $\phi = \langle \mu\ a; c\rangle$ then $\langle \mu\ a@\sigma(\mu)\rangle \in \psi$, with $(c; s) \in \sigma(\mu)$;*
- *if $\phi = \langle\langle \mu\ \mathsf{in}(x), \mu'\ \mathsf{move}(y)\rangle; c, n\rangle$ then $\langle \mu\ \mathsf{in}(b)@(c, n; i), \mu'\ \mathsf{move}(a)@(c; i)\rangle \in \psi$;*
- *if $\phi = \langle\langle \mu\ \mathsf{out}(x), \mu'\ \mathsf{move}(y)\rangle c, n\rangle$ then $\langle \mu\ \mathsf{out}(b)@(c, n; i), \mu'\ \mathsf{move}(a)@(c, n; i)\rangle \in \psi$.*

The proof coincides with that of Theorem A.10. In fact, ψ just collects information recorded in the remaining components of the analysis.

Theorem A.10 ***(Subject reduction)***
If $B \xrightarrow{\phi} B'$ and $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ then also $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B'$.

Proof. [Case (In)]. Let B be $\beta(x : \Gamma)\mathbf{B}^*_\mathbf{1}[P]^{\mu_P}_{c,n;b} || \beta(y : \Delta)\mathbf{B}^*_\mathbf{2}[Q]^{\mu_Q}_{c;i}$, with $P = \mathsf{in}(x).P_1 |\ P_2$ and $Q = \mathsf{move}(y).Q_1 |\ Q_2$ and with $\Delta \cap \Gamma = \emptyset$. Let B' be $\beta(x : \Gamma)\mathbf{B}^*_\mathbf{1}[P_1 |\ P'_2]^{\mu_P}_{c,n;b} || \beta(y : \Delta)\mathbf{B}^*_\mathbf{2}[Q_1 |\ Q_2]^{\mu_Q}_{c,n;i}$ and $\phi = \langle \mu_P\ \mathsf{in}(x), \mu_Q\ \mathsf{move}(y); c, n\rangle$. We have to prove that if $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ then $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B'$. $(\iota, \sigma, \epsilon, \rho, \kappa) \models^*_\psi B$ implies that:

(i) $x \in \iota(\mu_P)$, $(c, n; b) \in \sigma(\mu_P)$, and $(\iota, \sigma, \epsilon, \rho, \kappa) \models^\mu_\psi P_i$, with $i = 1, 2$;

(ii) $in(x) \in \iota(\mu_P)$;

(iii) $y \in \iota(\mu_Q)$, and $(\iota, \sigma, \epsilon, \rho, \kappa) \models^{\mu}_{\psi} Q_i$, with $i = 1, 2$;

(iv) $(c; s) \in \sigma(\mu_Q)$ and $\epsilon(\mu_P)(x) \cap \epsilon(\mu_Q)(y) \neq \emptyset$ (by Therorem A.6)

(v) $(c, n; s) \in \sigma(\mu_Q)$ and $\langle \mu_P\ \mathsf{in}(x)@(c, n; b), \mu_Q\ \mathsf{move}(y)@(c; i)\rangle \in \psi$

Now, by (i) we have that $(\iota, \sigma, \epsilon, \rho, \kappa) \models^{*}_{\psi} \beta(x : \Gamma)\mathbf{B}^{*}_{\mathbf{1}}[P_1|\ P'_2]^{\mu_P}_{c,n;b}$. By (iii) and (v) we have that $(\iota, \sigma, \epsilon, \rho, \kappa) \models^{*}_{\psi} \beta(y : \Delta)\mathbf{B}^{*}_{\mathbf{2}}[Q_1|\ Q_2]^{\mu_Q}_{c,n;i}$, and therefore the thesis.

Furthermore, note that $\langle \mu_P\ \mathsf{in}(x)@(c, n; b), \mu_Q\ \mathsf{move}(y)@(c; i)\rangle \in \psi$ corresponds to $\phi = \langle \mu_P\ \mathsf{in}(x), \mu_Q\ \mathsf{move}(y); c, n\rangle$. We recall that in the dynamic label, the compartment identifier is the one annotating the sub-comparment.

[Case (Out)]. The proof is analogous to the above one.

The other cases are similar to the ones proved in Theorem A.9. □