Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 194 (2008) 69–85
www.elsevier.com/locate/entcs

A Static Analysis for Beta-Binders

Chiara Bodei

Dipartimento di Informatica, Università di Pisa,
Via Pontecorvo, 56127 Pisa - Italia
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. Our analysis of a system is able to describe the essential behaviour of each box, tracking all the possible bindings of variables and all the possible intra- e inter-boxes communications. The analysis offers a basis for establishing static checks of biological dynamic properties. We finally apply our analysis to an example, based on an abstract specification of the interaction between a virus and cells of the immune system.

Keywords: Control Flow Analysis, Beta Binders

1 Introduction

A recent research line exploits well established theories and techniques of Formal Methods for Systems Biology, by using them to support the interpretation of the big amount of raw biological data now available for analysis. Among the several formalisms used to this aim, we recall process calculi, that abstractly describe interactions and communications between independent agents or processes. In particular, π-calculus [6] and Ambient Calculus [3] 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 [15,17] and BioAmbients [16] have been introduced. Also a version of CCS, RCCS [4], that addresses biological issues, has been presented. Other calculi have been instead specifically defined for biological modelling, such as κ-calculus [5], Brane calculi [2] and Beta Binders [13].

The underlying idea is that a biological system can be abstractly modelled as a concurrent system. The behaviour 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

doi:10.1016/j.entcs.2007.12.006

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.

We present here a control flow analysis for a subset 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 inter-boxes can happen. Our control flow analysis safely approximates the behaviour of systems, tracking all the possible bindings of variables and all the possible intra- e inter-boxes communications. 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.

The paper follows the tradition initiated by [10,9] of applying static techniques and, in particular, control flow analysis to process calculi used in modelling biological phenomena. These analyses are and can be inspired by the analyses previously applied to several process calculi, such as π-calculus (e.g. [1]) and Ambient calculus (e.g. [8]) 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.

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 how the Control Flow Analysis works on an example. We conclude in Section 6 with an assessment of our approach.

2 The Calculus

We briefly introduce the kernel of Beta Binders and we refer the interested reader to [13,14] for more details. In particular, for the sake of simplicity, we do not consider the *join* and the *split* semantic constructs.

2.1 The syntax

Essentially, processes are the parallel composition of boxes that contain π-calculus like [6] 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).

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 $\mathbf{B}, \mathbf{B_1}, \mathbf{B_2}, \ldots$. 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
$(\nu y)P$	restriction
$P\|P$	parallel composition
$!P$	replication
$\overline{x}y$	output
$x(y)$	input
$expose(x, \Gamma)$	addition of a new site
$hide(x)$	hiding of a site
$unhide(x)$	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 $\overline{x}y$ 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 $expose(x, \Gamma)$, $hide(x)$, $unhide(x)$ are used to manipulate beta binders. Prefixes $hide(x)$ and $unhide(x)$ make the elementary binder with subject x not available (hidden) and available (not hidden), respectively. A hidden binder cannot be used in a communication. Finally, the prefix $expose(x, \Gamma)$ adds the elementary beta binder $\beta(x : \Gamma)$ to the box and acts as a binder for x in the process it prefixes. 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 in the obvious way. 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 $\mathbf{B}[P]^{\mu}$, to distinguish different syntactic occurrences of boxes. We refer to $\mu \in \mathbf{Box}$ as the identity of the box, where **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 **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 rendered as a single box, where binders indicate the sites through which the boxes can interact with the external world. Finally, note that nesting of boxes is forbidden (see also Section 4).

2.2 Semantics

To simplify the definition of our control flow analysis in Section 3, 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. This partition works in a way that names from the same equivalence class are assigned a common *canonical name* and consequently there are only finitely many canonical names in any execution of a given process. 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. Therefore, we discipline the α-renaming of bound values and variables, allowed by the structural

congruence given below. Finally, we assume that the bound names of a process are renamed apart and that they do not clash with the free names.

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 (disjoint, respectively), we call them *compatible* (*incompatible*, respectively). The rules $Expose, Hide, Unhide$ allow the dynamic modifications of beta binders. The rule $Expose$ adds an extra site with the declared type: the name introduced is a placeholder which can be renamed to avoid clashes with the subjects of the other binders of the containing box. Finally, the rule $Hide, Unhide$ force the specified site to become hidden and unhidden, respectively.

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

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

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 $y, z \notin \tilde{v}$

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

provided that $\lfloor x' \rfloor = \lfloor x \rfloor$

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

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

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

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

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

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 [1]. 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 [7]. The first component ι gives information on the binders of boxes. The second component ϵ,

given a subject of a 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. The judgements for analysing beta processes and pi processes are $(\iota, \epsilon, \rho, \kappa) \models^* B$ and $(\iota, \epsilon, \rho, \kappa) \models^\mu P$, expressing respectively. 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.
- $\iota : \{*\} \cup \mathbf{Box} \to \wp(\mathbf{Box}) \uplus (\mathbf{N} \cup \mathbf{N^h} \cup \mathbf{N^u})$ is the *binder repository*, where $\mathbf{N^h}$ ($\mathbf{N^u}$, resp.), ranged over by x^h (x^u, resp.) represent the set of names in the hidden (unhidden) version, and where $\mathbf{N} \cup \mathbf{N^h} \cup \mathbf{N^u}$ is ranged over by $\hat{x}$. In $\iota(*)$ are collected all the names μ of the boxes under analysis and in $\iota(\mu)$ all the binder names x that are declared in the box labelled μ, and that may appear in the form x, x^h and x^u. These last two keep track of the fact that the corresponding name can appear as hidden or unhiddent, respectively.
- $\epsilon : \mathbf{Box} \to (\mathbf{N} \to \wp(\mathbf{N}))$ is the *abstract binder environment* that maps a binder of a certain box μ to its possible type set.
- $\rho : \mathbf{N} \to \wp(\mathbf{N})$ is the *abstract environment* that maps a name to the set of binders or names it can be bound to. We assume that for each name x, we have that $\rho(x) \supseteq \{x\}$. Moreover, we write $\rho(\{v_1, ..., v_n\})$ as a shorthand for $\rho(v_1) \times .... \times \rho(v_n)$.
- $\kappa : \mathbf{Box} \to (\mathbf{N} \to \wp(\mathbf{N}))$ is the *abstract channel environment* that maps a binder or a name occurring in a box μ, to the set of values that can be sent over it.

For keeping the analysis component finite, as said above, we have partitioned 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 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 checks 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. Finally, when the string of beta binders is empty, the analysis keeps track of the label of the box and the pi process in the box is passed to the process analysis.

Analysis of Pi Processes

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. The first condition of the rule for *output* requires that the set of names – recorded in

$(\iota,\epsilon,\rho,\kappa) \models^* Nil$	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 $(\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa) \models^\mu \overline{x}y.P$	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 $\forall a \in \rho(x) : \kappa(\mu)(a) \subseteq \rho(y) \wedge$ $\forall a \in \rho(x) : ((a \in \iota(\mu)) \cup (a^u \in \iota(\mu)), \wedge$ $\forall \mu' \in \iota(*) : ((b \in \iota(\mu')) \cup (b^u \in \iota(\mu')),$ $(\epsilon(\mu)(a) \cap \epsilon(\mu')(b) \neq \emptyset) \Rightarrow \kappa(\mu')(b) \subseteq \rho(y)$ $\wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa) \models^\mu expose(x,\Gamma).P$	iff $x \in \iota(\mu) \wedge \rho(\Gamma) \in \epsilon(\mu)(x) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa) \models^\mu hide(x).P$	iff $\forall a \in \rho(x) : a^h \in \iota(\mu) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$
$(\iota,\epsilon,\rho,\kappa) \models^\mu unhide(x).P$	iff $\forall a \in \rho(x) : a^u \in \iota(\mu) \wedge (\iota,\epsilon,\rho,\kappa) \models^\mu P$

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

the component κ – that can be sent along each element of $\rho(x)$, inside the box μ, includes the set of values to which y can evaluate. The rule for *input* is more structured, because it takes into account both the 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 binder name b – provided that b is a binder that occurs active or unhidden in another box μ' – and received along a – provided that a is a binder that occurs active or unhidden in μ, under the condition that the type of b is compatible with the one of a . To exemplify, suppose to have the process $P = x(y).P_1 | \overline{x}z.P_2$ 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 could be bound to the variable y occurring in the corresponding input on the shared channel x, i.e. $\{z\} \subseteq \kappa(\mu_P)(x) \subseteq \rho(x)$. Moreover, imagine to have in parallel another process $Q = \overline{u_1}v_1.Q_1$ inside the box μ_Q, such that the types of x and of u_1 are compatible (i.e. $\epsilon(\mu_P)(x) \cap \epsilon(\mu_Q)(u_1) \neq \emptyset$) and can both participate in a communication (i.e. $\{x, x^u\} \cap \iota(\mu_P) \neq \emptyset$ and $\{u_1, u_1^u\} \cap \iota(\mu_Q) \neq \emptyset$). Since v_1 can

be sent on the channel u_1, i.e. it should be included 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 for all the possible values a of x, a is included in $\iota(\mu)$ and all the possible values to which the elements of Γ may evaluate to are included in $\epsilon(\mu)(a)$. The rule for *hide* (*unhide*, resp.) demands that for all the possible values a of x, a^h (a^u, resp.) is included in $\iota(\mu)$. Recall that an *unhide* prefix $unhide(x)$ can only follow an *hide* or a declaration of a hidden binder and therefore there is no need to check whether $\epsilon(\mu)(x)$ includes the type of x.

3.1 Example 1: Intra- and Inter-Box Communication

The analysis of the simple beta process B

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

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

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

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

In fact, we have that,

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

(i) $\{z\} \subseteq \rho(y)$, since $\{z\} \subseteq \kappa(\mu_P)(x)$;
(ii) since $x \in \iota(\mu_P)$ and $u_1 \in \iota(\mu_Q)$ and $\epsilon(\mu_P)(x) \cap \epsilon(\mu_Q)(u_1) \supseteq \{c_1\}$ then $\{v_1\} \subseteq \rho(y)$, because $\{v_1\} \subseteq \kappa(\mu_Q)(u_1)$;
(iii) since $x \in \iota(\mu_P)$ and $u_2 \in \iota(\mu_R)$ and $\epsilon(\mu_P)(x) \cap \epsilon(\mu_R)(u_2) \supseteq \{c_2\}$ then $\{v_2\} \subseteq \rho(y)$, because $\{v_2\} \subseteq \kappa(\mu_R)(u_2)$;

Note that $\rho(y)$ includes the value z, hence correctly predicting the possibility of a 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):

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

Moreover, it includes both v_1 and v_2, therefore correctly predicting that an interaction between B_P and either B_Q, accounting for the following transition with B_Q (similarly with B_R):

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

The 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 $\epsilon(\mu_P)(x) \cap \epsilon(\mu_Q)(u_1) = \{c_1\}$ shows that the types of x and u_1 are compatible and a similar condition holds for the types of x and u_2.

3.2 Example 2: Interface Handling (1)

The analysis of a slightly different beta process B'

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

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

gives rise to:

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

The prefix *expose* causes the addition of z to the 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, while it can communicate with the third one, after the *expose*, as shown by the following transitions, where B' becomes

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

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, $expose(z, \{y, b_2\})$ becomes $expose(z, \{v_1, b_2\})$, as correctly reported by the analysis, where $\epsilon(\mu'_P)(z) = \{v_1, b_2\}$, because $\rho(\{y, b_2\}) = \rho(y) \times \rho(b_2) = \{v_1, b_2\}$.

3.3 Example 3: Interface Handling (2)

The analysis of another beta process B'

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

$$\text{where } P'' = x(y).hide(x).x(w).P''_1 \quad Q = \overline{u_1}v_1.Q_1 \quad R = \overline{u_2}v_2.R_1$$

gives rise to:

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

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

$$\begin{array}{l} \beta(x:\{c_1,c_2\})[hide(x).x^h(w).P''_1\{v_1/y\}]^{\mu''_P} \,||\, \beta(u_1:\{c_1\})[Q_1]^{\mu_Q} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ||\, \beta(u_2:\{c_2\})[\overline{u_2}v_2.R_1]^{\mu_R} \end{array}$$

and the second to

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

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 possible and also the analogous communication between x and u_1, as shown by the fact that $\rho(w) = \{v_1, v_2\}$. 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.

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 $\mathbf{B}\beta(x:\Gamma))[hide(x).P_1|P_2]$ 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 binder is a shared object, whose access is concurrent. This concurrent feature is responsible for the analysis imprecision. Suppose to have the following process:

$$\mathbf{B}\beta(x:\Gamma)[!hide(x).P_1|x(y).Q_1|\overline{x}z.Q_2|Q_3|!unhide(x).P_2]$$

It is impossible (since it is undecidable) to predict at compile time if the communication between $x(y).Q_1$ and $\overline{x}z.Q_2$ will be fired at run time. The 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.

We could obtain more precision in special cases, like in the process $[hide(x).P_1|P_2]$, when $x \notin \mathsf{fn}(P_2)$. Here the effect of the *hide* operation is only on the continuation P_1, where x is hidden until an $unhide(x)$ 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 the hidden occurrences of a variable cannot be used for possible communications, we could (i) replace the variable x with a new variable x_h, representing the hidden version of the corresponding 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 would correctly predict the absence of any communication on the hidden occurrence of x, since for $i = 1, 2$, $\epsilon(\mu_{P'})(x_h) \cap \epsilon(\mu_R)(u_i) = \emptyset$ and $\rho(w) = \emptyset$.

3.4 Validation

Our analysis is semantically correct. More precisely, we prove a subject reduction result: if (ϵ, ρ, κ) 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.

To obtain this result the following lemmata are necessary. The first states that estimates are resistant to substitution of closed terms for variables, and it holds for both 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* $(\iota, \epsilon, \rho, \kappa) \models^{*} B\{v/x\}$;

Proof. [Sketch] Both proofs proceed by structural induction. □

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 ***(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. [Sketch] The proof amounts to a straightforward inspection of each of the clauses defining the structural congruence clauses. □

We are now ready to state the subject reduction result. It expresses that our analysis is semantically correct with respect to the given semantics.

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

Proof. [sketch] By induction on the inference of $\rightarrow$. □

One can show that for any given B there always is a least choice of ϵ, ρ and κ. Moreover, the analysis that computes the least 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 [1,11,9].

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- e 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, i.e. 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 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.

- **propensity for communication of a binder** x: The binder x can be possibly involved in a communication if $\{x, x^u\} \cap \iota(\mu_P) \neq \emptyset$, i.e. if it can occur active or unhidden.
- **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 $\exists a : \{a, a^h, a^u\} \in \iota(\mu_P)$ and $\exists b : \{b, b^h, b^u\} \in \iota(\mu_Q)$ such that $\epsilon(\mu_P)(a) \cap \epsilon(\mu_Q)(b) \neq \emptyset$.
- **no interaction between** $\mathbf{B}[P]^{\mu_P}$ **and** $\mathbf{B}[Q]^{\mu_Q}$: The process $\mathbf{B}[P]^{\mu_P}$ cannot communicate with $\mathbf{B}[Q]^{\mu_Q}$, $\forall a : \{a, a^u\} \cap \iota(\mu_P) \neq \emptyset$, and $\forall b : \{b, b^u\} \cap \iota(\mu_Q) \neq \emptyset$ then $\epsilon(\mu_P)(a) \cap \epsilon(\mu_Q)(b) = \emptyset$, i.e. all the possible pairs of binders that show propensity for communication are incompatible.
- **isolation of** $\mathbf{B}[P]^{\mu_P}$: The process $\mathbf{B}[P]^{\mu_P}$ is isolated, when $\forall \mu_Q \in \iota(*)$, $\mathbf{B}[P]^{\mu_P}$ cannot interact with $\mathbf{B}[Q]^{\mu_Q}$.
- **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 $\forall a : \{a, a^u\} \cap \iota(\mu_P) \neq \emptyset$, and $\forall b : \{b, b^u\} \cap \iota(\mu_Q) \neq$

$\emptyset$ such that $\epsilon(\mu_P)(a) \cap \epsilon(\mu_Q)(b) \neq \emptyset$, we have that $\kappa(\mu_P)(a) = \emptyset$. i.e. even in the presence of a pair of binders that are compatible, and that show propensity for communication, there is no possible output from box μ_P.

- **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 binder b and $\exists a : \{a, a^h, a^u\} \in \iota(\mu_P)$ such that $\epsilon(\mu_Q)(b) = \epsilon(\mu_P)(a)$; moreover (ii) $\forall \mu_R \in \iota(*)$, with $\mu_R \neq \mu_P$ $\mathbf{B}[P]^{\mu_P}$ cannot communicate with $\mathbf{B}[R]^{\mu_R}$.

As far as the last property is concerned, we must recall that nesting is forbidden to keep the formalism simple. However, (see [14]), 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 only perform intra-communications and can be involved in inter-communications with the nesting box $\mathbf{B}[P]^{\mu_P}$ through a site with exactly the same type of the one in $\mathbf{B}[Q]^{\mu_Q}$.

5 Example: An Abstract Virus Attack

We illustrate our approach, by using an abstract specification, found in [13] 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.

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

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

The analysis gives rise to:

$$\iota(*) = \{\mu_C, \mu_{L_1}\}$$

$$\iota(\mu_C) = \{x, u\} \qquad \iota(\mu_{L_1}) = \{z\}$$

$$\begin{cases} \epsilon(\mu_C)(x) = \{v_1, ..., v_n\} \\ \epsilon(\mu_C)(u) = \{a_1\}, \end{cases} \quad \epsilon(\mu_{L_1})(z) = \{a_1, a_1'\}$$

$$\begin{cases} \kappa(\mu_C)(u) = \{r\}, \\ \kappa(\mu_C)(x) = \{a_1\}, \end{cases} \quad \kappa(\mu_{L_1})(z) = \emptyset$$

$$\rho(w) = \{a_1\} \qquad \rho(y) = \{r\}$$

that reflects one of the possible dynamic evolutions of the process:

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

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

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

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

In this case, the analysis would be:

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

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 $\epsilon(\mu_C)(u) \cap \epsilon(\mu_{L_1})(z) = \emptyset$.

6 Conclusions and Future Work

The paper presents a control flow analysis for Beta Binders, able to describe the essential behaviour of each of its boxes, in terms of possible interactions. Mainly, the analysis gives an over-approximations of (i) the binders of a box; (ii) the set of names the subject of a binder may be associated with; (iii) the set of values to which names can be bound, and of (iv) the set of names that can be sent over given channels.

Simply exploiting the soundness of our analysis, we can prove 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. These properties are quite simple, but are useful for illustrating how suitable static techniques can be adapted to model biological systems, giving some insights on their behaviour.

The study of Beta Binders suggests at least two possible future directions. In fact, 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 share some common value. 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. This is a feature that can be explored also from the static analysis point of view, looking for suitable ways of composing independent analyses of parts of systems.

It would also be interesting to understand how to analyse some more complex forms of interactions, that are currently studied in Beta Binders and that seem to be closer to biological interactions. Indeed, the requirement that the intersection of the types are not empty to allow an inter-boxes communication can result too abstract. In [12], for instance, this notion is made finer, by introducing a more general notion of *affinity*, based, in turn, on a larger notion of compatibility between types. Compatibility can give a measure of how much favourable a biological interaction is and could be cabled inside our analysis.

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.

Acknowledgement

We are grateful to Pierpaolo Degano for his helpful discussions and comments.

References

[1] 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.

[2] Luca Cardelli. *Brane Calculi - Interactions of Biological Membranes.* Proc. of CMSB 2004. LNCS 3082, Springer, 2005.

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

[4] Vincent Danos, Jean Krivine. *Transactions in RCCS.* In Proc. of CONCUR 2005, pp. 398-412.

[5] Vincent Danos, Cosimo Laneve. *Graphs for Core Molecular Biology.* In Proc. of CMSB 2003, pp. 34-46

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

[7] Hanne Riis Nielson and Flemming Nielson. Flow Logic: a multi-paradigmatic approach to static analysis. *The Essence of Computation: Complexity, Analysis, Transformation* LNCS 2566, pp. 223-244, Springer Verlag, 2002.

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

[9] Flemming Nielson, Hanne Riis Nielson, Corrado Priami, and Debora Schuch da Rosa. *Control flow analysis for BioAmbients*. Electronic Notes in Theoretical Computer Science, 2003.

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

[11] Flemming Nielson and Helmut Seidl. *Control-flow analysis in cubic time*. In Proc. of European Symposium on Programming (ESOP'01), LNCS 2028, pp. 252-268. Springer Verlag, 2001.

[12] Davide Prandi, Corrado Priami and Paola Quaglia. *Shape spaces in formal interactions*, ComPlexUs, 2(3-4):128139, 2006.

[13] Corrado Priami and Paola Quaglia. *Beta Binders for Biological Interactions*, CMSB '04, LNBI 3082, Springer (2005).

[14] Corrado Priami and Paola Quaglia. *Operational patterns in beta-binders, Transactions on Computational Systems Biology*, LNCS 3380, Springer (2005).

[15] 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).

[16] 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.

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