A Core Calculus for a Comparative Analysis of Bio-inspired Calculi

Cristian Versari

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
versari@cs.unibo.it

Abstract. The application of process calculi theory to the modeling and the analysis of biological phenomena has recently attracted the interests of the scientific community. To this aim several specialized, bio-inspired process calculi have been proposed, but a formal comparison of their expressivity is still lacking. In this paper we present $\pi@$, an extension of the π-Calculus with priorities and polyadic synchronisation that turns out to be suitable to act as a core platform for the comparison of other calculi. Here we show $\pi@$ at work by providing "reasonable" encodings of the two most popular calculi for modeling membrane interactions, namely, BioAmbients and Brane Calculi.

Keywords: pi-calculus, priority, polyadic synchronisation, BioAmbients, Brane Calculi.

1 Introduction

After the first use of π-Calculus for the modeling of biological processes [22], the applications of process calculi to Systems Biology attracted increasing research efforts. The direct employment of π-Calculus allowed the formalisation of several biological mechanism, its variants and extension [20,23,8] permitted the representation or analysis *in silico* of cellular processes [13,7]. To obtain higher abstraction level and biological faithfulness, more complex calculi have been proposed [4,24,21,10,11,12] which are based on or get inspiration from π-Calculus. Even if they present many common features, each calculus focuses its attention on particular biological entities or mechanisms. Their similarity induces the interest for a parallel analysis, but their specialisation does not allow a direct comparison.

The $\pi@$ language was designed to this aim: its simple but powerful extensions to π-Calculus – polyadic synchronisation and prioritised communication – allow to express the ideas shared by all these formalisms and flexibly adapt to represent the peculiarities of each one. Moreover, its simple syntax and semantics, very close to π-Calculus, allow a natural extension of many properties and results already stated for standard π-Calculus, thus facilitating $\pi@$ theoretical analysis.

In this paper we show $\pi@$ at work by encoding two of these formalisms: Brane Calculi and BioAmbients. Their straightforward embedding in the same language

R. De Nicola (Ed.): ESOP 2007, LNCS 4421, pp. 411–425, 2007.

allows to understand clearly their structural/semantical common points and differences and provides their ready-to-run implementation on top of a common platform.

The paper is structured as follows. Next section presents $\pi@$ language, first by introducing its extensions to π-Calculus, then by giving its syntax and semantics. Section 3 is devoted to the explanation of the central ideas behind the encodings, followed by their formalisation and analysis. For a detailed treatment of BioAmbients and Brane Calculi see [24,4].

2 The $\pi@$ Language

The $\pi@$ calculus – pronounced like the french "paillette" – consists in π-Calculus with the addition of two features: polyadic synchronisation and prioritised communication. The first one is used to model *localisation* of communication typical of the majority of bio-inspired calculi, which usually formalise it by the explicit introduction of compartments (i.e. ambients and membranes in the case of the two languages considered here). Priority is exploited as a powerful mechanism for achieving *atomicity*, that is the completion, without overlapping, of complex atomic operations by the execution of several simple steps.

Before presenting $\pi@$, we shortly recall π-Calculus syntax and semantics, on which $\pi@$ is strongly based.

2.1 The π-Calculus

Here we recall the syntax and the reduction semantics of π-Calculus, chosen as the basis for $\pi@$ because of the simplicity and closeness to the semantics used for the majority of bio-inspired calculi. For a full threatment of π-Calculus we refer to [14,15].

Definition 1. *Let*

$\mathcal{N}$ *be a set of names on a finite alphabet,* $x, y, z, \ldots \in \mathcal{N}$*;*
$\overline{\mathcal{N}} = \{\overline{x} \mid x \in \mathcal{N}\}$

The syntax of π*-Calculus is defined as*

$$P \quad ::= \quad \mathbf{0} \;\Big|\; \sum_{i \in I} \pi_i.P_i \;\Big|\; P \mid Q \;\Big|\; !\,P \;\Big|\; (\nu x)P$$

$$\pi \quad ::= \quad \tau \;\Big|\; x(y) \;\Big|\; \overline{x}\langle y \rangle$$

Definition 2. *The congruence relation* $\equiv$ *is defined as the least congruence satisfying alpha conversion, the commutative monoidal laws with respect to both* $(\mid, \mathbf{0})$ *and* $(+, \mathbf{0})$ *and the following axioms:*

$$\begin{aligned} (\nu x)P \mid Q &\equiv (\nu x)(P \mid Q) && \text{if } x \notin fn(Q) \\ (\nu x)P &\equiv P && \text{if } x \notin fn(P) \\ !\,P &\equiv !\,P \mid P \end{aligned}$$

where the function fn *is defined as*

$$
\begin{array}{rcl@{\qquad}rcl}
fn(\tau) & \stackrel{def}{=} & \emptyset & fn(x(y)) & \stackrel{def}{=} & \{x\} \\
fn(\overline{x}\langle y\rangle) & \stackrel{def}{=} & \{x,y\} & fn(\mathbf{0}) & \stackrel{def}{=} & \emptyset \\
fn(\pi.P) & \stackrel{def}{=} & fn(\pi)\cup fn(P) & fn(\sum_{i\in I}\pi_i.P_i) & \stackrel{def}{=} & \bigcup_i fn(\pi_i.P_i) \\
fn(P \mid Q) & \stackrel{def}{=} & fn(P)\cup fn(Q) & fn(!\,P) & \stackrel{def}{=} & fn(P) \\
fn((\nu x)P) & \stackrel{def}{=} & fn(P)\setminus\{x\} & & &
\end{array}
$$

Definition 3. *π-Calculus semantics is given in terms of the reduction system described by the following rules:*

$$
\frac{}{\tau.P \;\rightarrow\; P} \qquad \frac{}{(\mu(y).P+M) \mid (\overline{\mu}\langle z\rangle.Q+N) \;\rightarrow\; P\{z/y\} \mid Q}
$$

$$
\frac{P \;\rightarrow\; P'}{P \mid Q \;\rightarrow\; P' \mid Q} \qquad \frac{P \;\rightarrow\; P'}{(\nu\, x)P \;\rightarrow\; (\nu\, x)P'} \qquad \frac{P \equiv Q \quad P \;\rightarrow\; P' \quad P' \equiv Q'}{Q \;\rightarrow\; Q'}
$$

2.2 Polyadic Synchronisation

In π-Calculus channels and names are usually synonyms. Polyadic synchronisation (introduced in [3]) consists in giving *structure* to channels: each channel is composed of one or more names and identified by all of them in the exact sequence they occur. For example, an email address is usually written in the form *username*@*domain*, where *username* and *domain* are two strings – two names – both necessary to identify the given email address. Moreover, their order is crucial since *domain*@*username* specifies another, likely unexisting, address. Following this analogy, π@ channels are written in the form $name_1@name_2@\ldots@name_n$ without limit in the number of names, even if just two suffice for most of the applications. In other words, a channel is indicated by a vector of names $(name_1, name_2, \ldots, name_n), n \geq 1$, and communication between two processes may happen only if they are pursuing a synchronisation along channels composed of the same number of names, with the same multiplicity and appearing order.

Apart from this, communication in π@ happens in the same way as in π-Calculus. For example, the transition

$$
\overline{comm}\langle d\rangle.P \mid comm(x).Q \quad\rightarrow\quad P \mid Q\{d/x\}
$$

is still valid in π@. Output actions are overlined as usual, even in case of polyadic synchronisations:

$$
\overline{polyadic@comm}\langle d\rangle.P \mid polyadic@comm(x).Q \quad\rightarrow\quad P \mid Q\{d/x\}
$$

Communication produces the same renaming effect, but with one difference: in π-Calculus the transmission of a name always stands for the transmission of a channel, while in π@ the transmitted name may represent a channel or just one of its components, or both. For example, in the following expression the transmitted name d represents a channel in the first output action $\overline{d}\langle y\rangle$, while in $\overline{d@comm}\langle y\rangle$ it is just the first part of the channel $d@comm$.

$$\overline{polyadic@comm}\langle d\rangle.P \mid polyadic@comm(x).\big(\overline{x}\langle y\rangle \mid \overline{x@comm}\langle y\rangle\big) \quad \rightarrow \\ P \mid \overline{d}\langle y\rangle \mid \overline{d@comm}\langle y\rangle$$

For concision and readability, polyadic synchronisation is often used also in conjunction with polyadic communication:

$$\overline{polyadic@comm}\langle a,b,c\rangle.P \mid polyadic@comm(x,y,z).Q \quad \rightarrow \\ P \mid Q\{a/x,b/y,c/z\}$$

Finally, the following transitions are not allowed:

$$\overline{x@y}\langle\rangle.P \mid y@x().Q \quad \not\rightarrow \qquad\qquad (x \neq y)$$

$$\overline{x}\langle\rangle.P \mid x@x().Q \quad \not\rightarrow$$

In the first expression, the output and input channels are composed of the same names, but with different appearing order. In the second one, channels are represented by the same name but with different multiplicity. In both cases the vectors of names do not match.

2.3 Priority

Priority behaves as expected: a high-priority process holds the central processing unit and executes its job before any low priority process. In $\pi@$ high priority synchronisations or communications are executed before any other low priority action. Usually a high priority action is indicated by underlining the name of the channel one or more times. For example, the expression

$$\overline{stand}\langle x\rangle.P \mid \overline{\underline{walk}}\langle y\rangle.Q \mid \overline{\underline{\underline{run}}}\langle z\rangle$$

contains three processes with different, increasing priority. To express more than three levels of priority another notation is used, where the priority of the process is represented by a number following the channel names. The above expression may be rewritten as

$$\overline{stand:2}\langle x\rangle.P \mid \overline{walk:1}\langle y\rangle.Q \mid \overline{run:0}\langle z\rangle$$

where a lower priority action is labelled with a higher number (the highest priority is denoted by 0).

Interaction between processes may occur only if channels have the same priority. In this example

$$\overline{x}\langle y\rangle.P \mid x(z).Q \quad \not\rightarrow$$

$$\overline{x}\langle y\rangle.P \mid \underline{x}(z).Q \quad \rightarrow \quad P \mid Q\{y/z\}$$

only the second interaction is allowed, because the expressions x and $\underline{x}$ denote actually two different channels. Finally, as expected, low priority actions occur only if no higher priority action may occur:

$$\overline{l}\langle w\rangle \mid l(x).P \mid \overline{\underline{h}}\langle y\rangle \mid \underline{h}(z).Q \quad \not\rightarrow \quad \mathbf{0} \mid P\{w/x\} \mid \overline{\underline{h}}\langle y\rangle \mid \underline{h}(z).Q$$

$$\overline{l}\langle w\rangle \mid l(x).P \mid \overline{\underline{h}}\langle y\rangle \mid \underline{h}(z).Q \quad \rightarrow \quad \overline{l}\langle w\rangle \mid l(x).P \mid \mathbf{0} \mid Q\{y/z\} \quad \rightarrow \quad \mathbf{0} \mid P\{w/x\} \mid \mathbf{0} \mid Q\{y/z\}$$

The first of the two transitions is not allowed because interactions on low-priority channel l may happen only after the high-priority communication on channel $\underline{h}$. For a detailed survey of priority in process algebras, see [9].

2.4 The π@ Syntax and Semantics

The π@ language is very close to π-Calculus: from a syntactical point of view the only difference is the structure of channels, composed of multiple names followed by the priority of the action. We use μ to denote a vector of names $x_1, \ldots, x_n$ and $\mu : k$ to denote a channel, that is a vector of names μ followed by a colon and a natural number k specifying the priority. As usual, $\overline{\mu : k}$ represents an output operation along channel $\mu : k$, while $\alpha : k$ stands for a generic input, output or silent action τ of priority k.

Definition 4. *Let*

$\mathcal{N}$ *be a set of names on finite alphabet,* $x, y, z, \ldots \in \mathcal{N}$*;*
$\mathcal{N}^+ = \bigcup_{i>0} \mathcal{N}^i\,, \qquad \mu \in \mathcal{N}^+$*;*
$\overline{\mathcal{N}}^+ = \{\overline{\mu} \mid \mu \in \mathcal{N}^+\}$*;*
$\alpha \in \left(\, \overline{\mathcal{N}}^+ \cup \mathcal{N}^+ \cup \{\tau\} \,\right)$*;*

The syntax of π@ *defined as*

$$P \quad ::= \quad \mathbf{0} \;\Big|\; \sum_{i\in I} \pi_i.P_i \;\Big|\; P \mid Q \;\Big|\; !\,P \;\Big|\; (\nu x)P$$

$$\pi \quad ::= \quad \tau{:}k \;\Big|\; \mu{:}k(x) \;\Big|\; \overline{\mu{:}k}\langle x\rangle$$

As previously introduced, some abbreviations are very often used in this paper:

$$\mu(x) = \mu{:}2(x) \qquad \overline{\mu}\langle x\rangle = \overline{\mu{:}2}\langle x\rangle$$
$$\underline{\mu}(x) = \mu{:}1(x) \qquad \overline{\underline{\mu}}\langle x\rangle = \overline{\mu{:}1}\langle x\rangle$$
$$\underline{\underline{\mu}}(x) = \mu{:}0(x) \qquad \overline{\underline{\underline{\mu}}}\langle x\rangle = \overline{\mu{:}0}\langle x\rangle$$

The definition for structural congruence $\equiv$ is exactly the same as given for π-Calculus, where the function fn is naturally extended to the π@ syntax, that is

$$fn(\mu : k(y)) \stackrel{def}{=} \{\mu_1, \ldots, \mu_n\}$$
$$fn(\overline{\mu : k}\langle y\rangle) \stackrel{def}{=} \{\mu_1, \ldots, \mu_n, y\}$$

where $\mu = \mu_1 @ \cdots @ \mu_n$. The reduction semantics is very similar, but defined in terms of an auxiliary function $I^k(P)$, representing the set of actions of priority k which the process P may immediately execute. For example, if

$$P = a.Q \mid \underline{b} \mid \underline{\overline{c}}.R \mid \underline{\overline{d}} + \underline{e}.S \mid \overline{a}.T$$

then $I^0(P) = \{\overline{c}, e\}$, $I^1(P) = \{b, \overline{d}\}$, $I^2(P) = \{a, \overline{a}, \tau\}$, where the availability of τ action derives from the interaction of the first and last process.

Definition 5. *Let $I^k(P)$ be*

$$\begin{aligned}
I^k\big(\sum_i \alpha_i : l_i.P_i\big) &= \{\alpha_i \mid l_i = k\};\\
I^k\big((\nu\ y)\ P\big) &= I^k(P) \setminus \big\{\alpha \mid y \in \{x_1, \ldots, x_n\} \wedge \\
&\qquad (\alpha = x_1 @ \ldots @ x_n \ \vee\ \alpha = \overline{x_1 @ \ldots @ x_n})\big\};\\
I^k\big(!P\big) &= I^k(P \mid P);\\
I^k\big(P \mid Q\big) &= I^k(P) \cup I^k(Q) \cup \{\tau \mid I^k(P) \cap \overline{I^k(Q)} \neq \emptyset\},\\
&\qquad \overline{I^k(Q)} = \big\{\overline{\alpha} \mid \alpha \in I^k(Q)\big\}
\end{aligned}$$

$\pi@$ semantics is given in terms of the following reduction system:

$$\frac{\tau \notin \bigcup_{i<k} I^i(M)}{\tau : k.P + M \ \rightarrow_k \ P} \qquad \frac{P \ \rightarrow_k \ P'}{(\nu\ x)P \ \rightarrow_k \ (\nu\ x)P'}$$

$$\frac{\tau \notin \bigcup_{i<k} I^i(M \mid N)}{(\mu : k(y).P + M) \mid (\overline{\mu} : k\langle z\rangle.Q + N) \ \rightarrow_k \ P\{z/y\} \mid Q}$$

$$\frac{cP \ \rightarrow_k \ P' \qquad \tau \notin \bigcup_{i<k} I^i(P \mid Q)}{P \mid Q \ \rightarrow_k \ P' \mid Q} \qquad \frac{P \equiv Q \quad P \ \rightarrow_k \ P' \quad P' \equiv Q'}{Q \ \rightarrow_k \ Q'}$$

$\pi@$ reduction rules are exactly the same of π-Calculus, except for the additional condition $\tau \notin \bigcup_{i<k} I^i(\ldots)$ which avoids the execution of low priority actions if higher priority communications (represented by τ actions) are immediately available.

2.5 Notation

In addition to standard reduction relation $\rightarrow_k$, some derived relations are used for the formulation of theorems. As usual, $\rightarrow_k^*$ is the reflective-transitive closure of $\rightarrow_k$, while $\rightarrow_k^{(n)}$ is used to evidence the length of the derivation, that is

$$P \rightarrow^{(n)} Q \quad \text{iff} \quad \exists\, P_1, \ldots, P_{n-1} : P \rightarrow P_1 \rightarrow \ldots \rightarrow P_{n-1} \rightarrow Q$$

Similar notation are used for the derived relations.

Definition 6. *Let P, Q, Q' be $\pi@$ processes. The reduction relations $\rightarrow$, $\rightarrowtail_k$, $\mapsto$, $\rightrightarrows_k$, are defined as follows:*

1. $P \rightarrow Q \triangleq P \rightarrow_k Q,\ k \in \mathbb{N}$;
2. $P \rightarrowtail_k Q \triangleq P \rightarrow_h Q,\ h \leq k$;
3. $P \mapsto Q \triangleq P \rightarrow_k P' \wedge P' \rightarrowtail^*_{k-1} Q,\ \tau \notin \bigcup_{i<k} I^i(Q)$.
4. $P \rightrightarrows_k Q \triangleq P \rightarrowtail^*_{k-1} Q,\ \tau \notin \bigcup_{i<k} I^i(Q) \ \wedge$
   $\big(P \rightarrowtail^*_{k-1} Q',\ \tau \notin \bigcup_{i<k} I^i(Q')$ *implies* $Q \equiv Q'\big)$.

$\rightarrow$ is the standard reduction relation, disregarding the priority of the reduction. $\rightarrowtail_k$ denotes the derivation through reduction with priority higher or equal to k. $\mapsto$ indicates that, after a reduction with a certain (low) priority and, in case, a sequence of higher priority actions, the process comes back to a state where it is ready to perform only low priority synchronisations. $\rightrightarrows_k$ states a confluence property of the process, meaning that all the states from which it is not possible to perform a reduction of priority higher than k and reachable only by reductions of priority higher than k, are congruent.

3 Encodings

The key feature which differentiates recent bio-inspired calculi from π-Calculus is the explicit formalisation of compartments. BioAmbients is a modified version of Ambient calculus [5], where compartments are represented by ambients, a sort of boxes containing processes or other nested boxes. In Brane compartments are bounded by membranes, on the surface of which processes compute. Both ambients and membranes are organised in a tree structure, both can dinamically modify this structure by performing for example *merge*, *enter*/*exit* or *exo* operations. The central issue is *how* they modify this structure: the most observable difference is the bitonality preserved by brane semantics and totally absent in BioAmbients. As remarked in [4], this peculiarity is enough to preclude an immediate embedding of one language into the other, thus not allowing a direct comparison of their expressivity. An alternative analysis can be performed by encoding both in a third formalism and compare their encoding functions. These functions must obviously satisfy some "reasonable" properties (as discussed in section 3.1) and they must also be as simple as possible by hiding irrelevant details. $\pi@$ features were chosen to meet these criteria: the lack of a predefined semantics for compartments together with the possibility of expressing localisation (by means of polyadic synchronisation) and complex atomic operations (by means of priority) place $\pi@$ one abstraction level lower, as a sort of *assembly* language for compartmentalised formalisms.

3.1 Requirements

The fundamental criterion guiding any encoding is the preservation of some addressed semantics. According to [16], this often means that the encoding function $[\![\cdot]\!]$ must at least fulfill the notion of *operational correspondence*, characterised

by two complementary properties: completeness and soundness. The first means that every possible execution of the source language may be simulated by its translation, the second ensures that all the states reached by the translation correspond to some state of the source. Since all the languages we consider are Turing-complete (even Brane [2,6], despite of its simplicity), as usual for concurrent languages we require some additional criteria. As remarked in [17], a *reasonable* encoding should also preserve the degree of distribution of the source language (i.e. homomorphism w.r.t. parallel composition) and should not depend on the channel (or compartment) names of the term to be encoded. This also implies a very valuable property, that is modular compilation, as discussed in [1]. In addition to the cited criteria, we also require the encoding to preserve the termination or diverging behaviour of the translated term, in order to obtain a totally faithful encoding function. The following definition formalises the notion of *reasonable encoding* used in this paper.

Definition 7. *An encoding* $[\![\cdot]\!]$ *is reasonable if it enjoys the following properties:*

1. *homomorphism w.r.t. parallel composition:*
   $[\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!]$;
2. *renaming preserving:*
   for any permutation of the source names θ, $[\![\theta(P)]\!] = \theta([\![P]\!])$;
3. *termination invariance:* $P \Downarrow$ *iff* $[\![P]\!] \Downarrow$, $P \Uparrow$ *iff* $[\![P]\!] \Uparrow$;
4. *operational correspondence:*
   (a) if $P \to P'$ *then* $[\![P]\!] \to^* [\![P']\!]$,
   (b) if $[\![P]\!] \to^* Q$ *then* $\exists P' : P \to^* P' \wedge Q \to^* [\![P']\!]$.

3.2 Basic Ideas

Compartment and their nesting are very intuitive abstractions: the simple statement that an object is enclosed in a box suggests that it is someway isolated from the external context; putting one box into another means that, after the operation, the inner box *with all its content* are located inside the outer one; merging the content of two boxes implies putting in the same box *all the enclosed objects*. To obtain this behaviour in $\pi@$ we must recognise the exact meaning of every operation on compartments and reproduce step by step the same semantics.

The first concept to unfold is nesting: compartments compose a dynamical tree structure which must be encoded in $\pi@$. As suggested in [15], these kind of structures can be represented as a set of processes linked by the share of private channels between parent and child nodes. Like in [22], the scoping of private names represents the boundaries of compartments, but thanks to polyadic synchronisation each private name may represent an unlimited number of private communication channels, as shown in section 2.2. If each node is supplied with one distinctive name, the simplest way to encode the tree is by ensuring that each node knows the name identifying its parent compartment.

Therefore, trivial changes in the tree structure may affect an unlimited number of processes: the simple disclosure of a compartment implies that all contained

processes must be notified of their new parent compartment name. The same situation occurs when splitting or merging the content of two compartments, like in $merge+/merge-$ and $exo/exo^{\perp}$ operations. In $\pi@$ this turns out to be a sort of *multicast* communication, where specifical groups of nodes – that is sibling and child processes – must receive on the proper channel a new compartment name. This result is achieved by a smart use of priority levels: a high priority loop notifies in turn all the interested processes and ends when such processes do not exist anymore. By a single line of code, we obtain in $\pi@$ the same mechanism typical of broadcast communication:

$$BCAST \quad \equiv \quad !\,\underline{bcast}(x,y).(\underline{\tau} + \underline{\overline{x}}\langle y\rangle.\underline{\overline{bcast}}\langle x,y\rangle)$$

The above process can be triggered by an output operation $\underline{\overline{bcast}}\langle chn, newchn\rangle$ and terminate when no high priority synchronisations are available, leaving no residual terms. Obviously, a high priority complementary output loop $!\ \underline{\overline{bcast}}\langle chn, newchn\rangle$ would cause the system to hang, since it prevents any other computation with normal priority. This is one of the reasons that do not allow a trivial translation of Brane and Bioambients replication operators and induce an explicit reproduction of their unfolding technique in both the encoding functions.

3.3 Encoding BioAmbients

Ambients are containers organised in a tree structure: running processes and nested sub-ambients are located inside them. If each node of the tree represents an ambient, nodes are complex structures: each node may contain zero or more parallel processes and may be linked zero or more nested sub-ambients. Consequently, for the implementation of the tree structure each encoded BioAmbients process must be aware of the name of its containing (immediate) ambient, but also of the name indicating the parent of its immediate ambient. This explains why the encoding function $[\![\ ]\!]^{\alpha}_{K,a,pa}$ requires the (bound) names a and ap, which represent the immediate ambient and the parent ambient, respectively. The free names oa, opa are placeholders standing for the immediate ambient and parent ambient of the outer processes, while bound names na and npa represent a new ambient or new parent ambient name received by the process. The first parameter K is the set of names used for the explicit unfolding of replicated processes when encoding the bang operator: the cardinality of K is the number of bangs in front of the process to encode.

Definition 8. *The function* $[\![\cdot]\!]^{\alpha}$ *from BioAmbients to* $\pi@$ *processes is defined as follows:*

$$\begin{aligned}
[\![\ \mathbf{0}\ ]\!]^{\alpha} &\triangleq \mathbf{0}\\
[\![\ P \mid Q\ ]\!]^{\alpha} &\triangleq [\![\ P\ ]\!]^{\alpha} \mid [\![\ Q\ ]\!]^{\alpha}\\
[\![\ (new\ n)P\ ]\!]^{\alpha} &\triangleq [\![\ (new\ n)P\ ]\!]^{\alpha}_{\emptyset,oa,opa}\\
[\![\ [\ P\ ]\ ]\!]^{\alpha} &\triangleq [\![\ [\ P\ ]\ ]\!]^{\alpha}_{\emptyset,oa,opa}
\end{aligned}$$

$$
\begin{aligned}
[\![\, !\, P \,]\!]^{\alpha} &\triangleq [\![\, !\, P \,]\!]^{\alpha}_{\emptyset,oa,opa} \\
[\![\, \mathbf{0} \,]\!]^{\alpha}_{K,a,pa} &\triangleq \mathbf{0} \\
[\![\, P \mid Q \,]\!]^{\alpha}_{K,a,pa} &\triangleq [\![\, P \,]\!]^{\alpha}_{K,a,pa} \mid [\![\, Q \,]\!]^{\alpha}_{K,a,pa} \\
[\![\, (new\ n)P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \nu n [\![\, P \,]\!]^{\alpha}_{K,a,pa} \\
[\![\, [\, P \,] \,]\!]^{\alpha}_{K,a,pa} &\triangleq \nu c [\![\, P \,]\!]^{\alpha}_{K,c,a} \\
[\![\, !\, P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \nu b(BANG(b,a,pa) \mid [\![\, P \,]\!]^{\alpha}_{K\cup\{b\},a,pa} \mid \\
&\quad\ !\, \underline{new@b}(na,npa).[\![\, P \,]\!]^{\alpha}_{K\cup\{b\},na,npa}) \\
[\![\, \sum_{i\in I,\ I\neq\emptyset} \xi_i.P_i \,]\!]^{\alpha}_{K,a,pa} &\triangleq BCAST \mid \nu s(!\, \underline{s}(na,npa). \\
&\quad\ ([\![\, \xi_i.P_i \,]\!]^{\alpha}_{K,na,npa} + TREE(s,na,npa)) \mid \\
&\quad\ [\![\, \xi_i.P_i \,]\!]^{\alpha}_{K,a,pa} + TREE(s,a,pa)) \\
[\![\, enter\ n.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq enter@n@pa(x).\underline{\overline{bcast}}\langle pa,a,x\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,x} \mid \Pi_K) \\
[\![\, accept\ n.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{enter@n@pa}\langle a\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, exit\ n.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq expel@n@pa(x).\underline{\overline{bcast}}\langle pa,a,x\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,x} \mid \Pi_K) \\
[\![\, expel\ n.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{expel@n@a}\langle pa\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, merge-\ n.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq merge@n@pa(x). \\
&\quad\ \underline{\overline{bcast}}\langle merge,a,x\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,x,pa} \mid \Pi_K) \\
[\![\, merge+\ n.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{merge@n@pa}\langle a\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, local\ n!\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{local@n@a}\langle m\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, local\ n?\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq local@n@a(m).([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, s2s\ n!\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{s2s@n@pa}\langle m\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, s2s\ n?\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq s2s@n@pa(m).([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, p2c\ n!\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{p2c@n@a}\langle m\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, c2p\ n?\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq p2c@n@pa(m).([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, c2p\ n!\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq \overline{c2p@n@pa}\langle m\rangle.([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
[\![\, p2c\ n?\{m\}.P \,]\!]^{\alpha}_{K,a,pa} &\triangleq c2p@n@a(m).([\![\, P \,]\!]^{\alpha}_{\emptyset,a,pa} \mid \Pi_K) \\
BANG(b,a,pa) &\equiv\ !\, \underline{b}(na,npa). \\
&\quad\ (\underline{\underline{unfold@b}}.\overline{\underline{new@b}}\langle na,npa\rangle + TREE(b,na,npa)) \mid \\
&\quad\ \underline{\underline{unfold@b}}.\overline{\underline{new@b}}\langle a,pa\rangle + TREE(b,a,pa)) \\
TREE(b,na,npa) &\equiv \underline{\underline{npa@na}}(x).\overline{\underline{b}}\langle na,x\rangle + \underline{\underline{merge@npa}}(x).\overline{\underline{b}}\langle na,x\rangle + \\
&\quad\ \underline{\underline{merge@na}}(x).\overline{\underline{b}}\langle x,npa\rangle \\
\Pi_K &\equiv \overline{\underline{\underline{unfold@k_1}}} \mid \cdots \mid \overline{\underline{\underline{unfold@k_n}}}, \\
&\quad\ K = \{k_1,\ldots,k_n\} \\
BCAST &\equiv\ !\, \underline{bcast}(x,y,z).(\underline{\underline{x@y}}\langle z\rangle.\overline{\underline{bcast}}\langle x,y,z\rangle + \underline{\tau})
\end{aligned}
$$

The strict relationship between BioAmbients and π-Calculus simplifies the encoding of base operators: parallel composition and restriction are homomorphically translated. Like for restriction, each ambient produces a private name, but in this case the new name is inserted in the tree structure by passing it to the subsequent encoding. Remarkably, the translation of each communication or capability choice requires a loop: in fact, each process ready to execute an action may be notified of an occurring change in the nesting tree structure, caused by other processes. Consequently, it should receive and replace the proper names representing its immediate and/or parent ambients before attempting to perform the desired actions: each $TREE$ subprocess is ready to handle this kind of events. Communications and capabilities are directly encoded by means of polyadic synchronisation: the possibility of using an unlimited number of names for each $pi@$ channel (up to three, in this case) simplifies extremely the simultaneous expression of localisation inside ambients and synchronisation on different directions ($p2p$, $s2s$, ...) equipped with names. After the execution of each capability, the reorganisation of the tree structure and the eventual unfolding of replicated processes is obtained by a sequence of high priority actions consisting in the triggering of one $BCAST$ loop and a set of $unfold@k_i$ synchronisations.

Finally, the encoding function $[\![\cdot]\!]^{\alpha}$ enjoys the requirements discussed in section 3.1, as stated by the following theorem.

Theorem 1. *$[\![\cdot]\!]^{\alpha}$ is a reasonable encoding (modulo structural congruence), that is: let P, P_1, P_2 be BioAmbients processes, let Q be a $\pi@$ process, then*

1. $[\![P_1 \circ P_2]\!]^{\alpha} = [\![P_1]\!]^{\alpha} \mid [\![P_2]\!]^{\alpha}$;
2. *for any permutation of the source names θ,* $[\![\theta(P)]\!]^{\alpha} = \theta([\![P]\!]^{\alpha})$;
3. $P \Downarrow$ *iff* $[\![P]\!]^{\alpha} \Downarrow$, $P \Uparrow$ *iff* $[\![P]\!]^{\alpha} \Uparrow$;
4. *(a) if* $P \rightarrow P_1$ *then* $\exists P_2 : P_2 \equiv P_1 \wedge [\![P]\!]^{\alpha} \rightarrow^* [\![P_2]\!]^{\alpha}$;
   (b) if $[\![P]\!]^{\alpha} \rightarrow^* Q$ *then* $\exists P_1 : P \rightarrow^* P_1 \wedge Q \rightarrow^* [\![P_1]\!]^{\alpha}$.

3.4 Encoding Brane Calculi

Like ambients, membranes are organised in tree structures: each node of the tree may contain membrane processes or nested membranes. Unlike BioAmbients, Brane Calculi present two main entities: systems and branes. Their distinction implies slightly different translations, because the encoding function of systems needs only two parameters (K, the set corresponding to the bang operators in front of the system and pc, the name representing the parent compartment) while an additional parameter is needed for encoding branes (c, the name of the compartment where the brane process resides). Similarly to BioAmbients encoding, oc and opc are placeholders standing for the compartment and parent compartment of outer processes, while nc and npc are bound names representing the new compartment and new parent compartment received during the tree structure reorganisation.

Definition 9. *The function* $[\![\cdot]\!]^{\beta}$ *from Brane to* $\pi@$ *processes is defined as follows:*

$$
\begin{aligned}
[\![\diamond]\!]^{\beta} &\triangleq \mathbf{0} \\
[\![P \circ Q]\!]^{\beta} &\triangleq [\![P]\!]^{\beta} \mid [\![Q]\!]^{\beta} \\
[\![!\, P]\!]^{\beta} &\triangleq [\![!\, P]\!]^{\beta}_{\emptyset, oc} \\
[\![\sigma(\!| P |\!)]\!]^{\beta} &\triangleq [\![\sigma(\!| P |\!)]\!]^{\beta}_{\emptyset, oc} \\
[\![\diamond]\!]^{\beta}_{K,pc} &\triangleq \mathbf{0} \\
[\![P \circ Q]\!]^{\beta}_{K,pc} &\triangleq [\![P]\!]^{\beta}_{K,pc} \mid [\![Q]\!]^{\beta}_{K,pc} \\
[\![!\, P]\!]^{\beta}_{K,pc} &\triangleq \nu b([\![P]\!]^{\beta}_{K\cup\{b\},pc} \mid\, !\, \underline{\underline{new@b}}(npc).[\![P]\!]^{\beta}_{K\cup\{b\},npc} \mid \\
&\quad\; !\, \underline{\underline{b}}(npc). \\
&\quad\; (\underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle npc\rangle + \underline{\underline{exo@npc}}(x).\overline{\underline{b}}\langle x\rangle) \mid \\
&\quad\; \underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle pc\rangle + \underline{\underline{exo@pc}}(x).\overline{\underline{b}}\langle x\rangle) \\
[\![\sigma(\!| P |\!)]\!]^{\beta}_{K,pc} &\triangleq \nu c([\![\sigma]\!]^{\beta}_{K,c,pc} \mid [\![P]\!]^{\beta}_{K,c}) \\
[\![\mathbf{0}]\!]^{\beta}_{K,c,pc} &\triangleq \mathbf{0} \\
[\![\sigma \mid \rho]\!]^{\beta}_{K,c,pc} &\triangleq [\![\sigma]\!]^{\beta}_{K,c,pc} \mid [\![\rho]\!]^{\beta}_{K,c,pc} \\
[\![!\, \sigma]\!]^{\beta}_{K,c,pc} &\triangleq \nu b(BANG(b,c,pc) \mid [\![\sigma]\!]^{\beta}_{K\cup\{b\},c,pc} \mid \\
&\quad\; !\, \underline{\underline{new@b}}(nc,npc).[\![\sigma]\!]^{\beta}_{K\cup\{b\},nc,npc}) \\
[\![a.\sigma]\!]^{\beta}_{K,c,pc} &\triangleq BCAST \mid \nu s(!\, \underline{\underline{s}}(nc,npc). \\
&\quad\; ([\![a.\sigma]\!]^{\beta}_{K,nc,npc} + TREE(s,nc,npc)) \mid \\
&\quad\; [\![a.\sigma]\!]^{\beta}_{K,c,pc} + TREE(s,c,pc)) \\
[\![phago_n.\sigma]\!]^{\beta}_{K,c,pc} &\triangleq phago@n@pc(x).\overline{\underline{\underline{bcast}}}\langle pc,c,x\rangle.([\![\sigma]\!]^{\beta}_{\emptyset,c,x} \mid \Pi_K) \\
[\![phago^{\perp}_n(\rho).\sigma]\!]^{\beta}_{K,c,pc} &\triangleq \nu x(\overline{phago@n@pc}\langle x\rangle.([\![\sigma]\!]^{\beta}_{\emptyset,c,pc} \mid [\![\rho]\!]^{\beta}_{\emptyset,x,c} \mid \Pi_K)) \\
[\![exo_n.\sigma]\!]^{\beta}_{K,c,pc} &\triangleq exo@n@pc(x).\overline{\underline{\underline{bcast}}}\langle exo,c,x\rangle.([\![\sigma]\!]^{\beta}_{\emptyset,pc,x} \mid \Pi_K) \\
[\![exo^{\perp}_n.\sigma]\!]^{\beta}_{K,c,pc} &\triangleq \overline{exo@n@c}\langle pc\rangle.([\![\sigma]\!]^{\beta}_{\emptyset,c,pc} \mid \Pi_K) \\
[\![pino(\rho).\sigma]\!]^{\beta}_{K,c,pc} &\triangleq \nu x\, \tau.([\![\sigma]\!]^{\beta}_{\emptyset,c,pc} \mid [\![\rho]\!]^{\beta}_{\emptyset,x,c} \mid \Pi_K) \\
BANG(b,c,pc) &\equiv\; !\, \underline{\underline{b}}(nc,npc). \\
&\quad\; (\underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle nc,npc\rangle + TREE(b,nc,npc)) \mid \\
&\quad\; \underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle c,pc\rangle + TREE(b,c,pc)) \\
TREE(b,nc,npc) &\equiv\; \underline{\underline{npc@nc}}(x).\overline{\underline{b}}\langle nc,x\rangle + \underline{\underline{exo@npc}}(x).\overline{\underline{b}}\langle nc,x\rangle + \\
&\quad\; \underline{\underline{exo@nc}}(x).\overline{\underline{b}}\langle npc,x\rangle \\
\Pi_K &\equiv\; \overline{\underline{\underline{unfold@k_1}}} \mid \cdots \mid \overline{\underline{\underline{unfold@k_n}}}, \\
&\quad\; K = \{k_1,\ldots,k_n\} \\
BCAST &\equiv\; !\, \underline{\underline{bcast}}(x,y,z).(\overline{\underline{\underline{x@y}}}\langle z\rangle.\overline{\underline{\underline{bcast}}}\langle x,y,z\rangle + \underline{\underline{\tau}})
\end{aligned}
$$

Like for BioAmbients encoding, each operation of the original language is translated with a synchronisation followed by a sequence of high priority actions which manage the reorganisation of the tree structure and the unfolding of replicated processes involved in the computation. The presence of two distinct replication operators leads to two slightly different encodings which reflect the fact that systems are only provided of parent compartment, while branes present also their immediate compartment.

Also the encoding function $[\![\cdot]\!]^{\beta}$ enjoys the requirements discussed in section 3.1.

Theorem 2. *$[\![\cdot]\!]^{\beta}$ is a reasonable encoding (modulo structural congruence), that is: let P, P_1, P_2 and ρ_1, ρ_2 be respectively Brane systems and processes, let Q be a $\pi@$ process, then*

1. $[\![P_1 \circ P_2]\!]^{\beta} = [\![P_1]\!]^{\beta} \mid [\![P_2]\!]^{\beta}$,
   $[\![\rho_1 \mid \rho_2]\!]^{\beta} = [\![\rho_1]\!]^{\beta} \mid [\![\rho_2]\!]^{\beta}$
2. *for any permutation of the source names θ,* $[\![\theta(P)]\!]^{\beta} = \theta([\![P]\!]^{\beta})$;
3. $P \Downarrow$ *iff* $[\![P]\!]^{\beta} \Downarrow$, $P \Uparrow$ *iff* $[\![P]\!]^{\beta} \Uparrow$;
4. *(a) if* $P \rightarrow P_1$ *then* $\exists P_2 : P_2 \equiv P_1 \wedge [\![P]\!]^{\beta} \rightarrow^* [\![P_2]\!]^{\beta}$;
   (b) if $[\![P]\!]^{\beta} \rightarrow^* Q$ *then* $\exists P_1 : P \rightarrow^* P_1 \wedge Q \rightarrow^* [\![P_1]\!]^{\beta}$.

3.5 Encodings Comparison

Brane and BioAmbients are different for several aspects. Brane has a very simple syntax, provided with only three base operations, lacks any restriction and choice operator, there is no explicit name communication mechanism. BioAmbients is provided with elaborate, multi-level communication primitives in addition to compartment operations. But in [4] all these operators are considered as possible Brane extensions and their encoding in $\pi@$ would be exactly the same of the original BioAmbients operators. Therefore, the crucial difference is not intended to be in the syntax, but in the semantics: Brane compartment operations have been designed to preserve *bitonality*, a concept totally absent in BioAmbients, furthermore processes are thought to be *on the surface* of membranes, not *inside* ambients.

By translating both languages in $\pi@$, we are able to discern at first sight where processes are exactly placed and what are the differences in the dynamical rearrangement of the tree structure. The encoding of *phago*, *exo*, *pino*, *enter*/*accept*, *exit*/*expel*, *merge*$\pm$ operations clearly shows that both kind of processes own the same information about their localisation in the tree, therefore the tree structure is very similar: the only difference is in the scoping of the names of their parent ambients. In fact, unlike the encoding of ambients, the encoding function of a Brane system P does not need the parameter c representing the immediate compartment of the process. This difference justifies the assumption that Brane processes are located *on* membranes. Bitonality simply arises in the order of the parameters given to the last term of the $TREE$ subprocess and in the

choice of the names broadcasted and recursively passed to the encoding function (this is particulary evident in the $exo^{\perp}$ operation, where the name of the parent compartment pc, instead of the immediate compartment c, is the object of communication).

In conclusion, the two analised languages present much more common points than differences: concurrency, interleaving semantics, compartments with tree nesting and very similar structure for nodes, implicit multicast communications within compartment boundaries. If we consider all the extensions proposed in [4], the two formalisms may be considered close variants of the same language.

4 Conclusions and Future Work

We presented a new calculus, $\pi@$, designed to be a core language for analysing formalisms which model localisation and compartmentalisation. We showed $\pi@$ at work by a formal comparison of the reasonable encodings of BioAmbients and Brane languages, which permitted to clarify their structural similarities and semantical differences.

This is the first part of a wide analysis towards a disparate variety of biologically inspired languages, like [21,11,12]. The generality of $\pi@$ features allow to extend its application not only to process calculi, but also to formalisms not pertaining to concurrency theory, like P systems [18,25].

Finally, thanks to the strong affinity with π-Calculus, we plan to implement a stochastic version of $\pi@$ as a direct extension of the SPIM simulator [19], hence providing a platform on top of which it is possible to immediately execute all the embedded formalisms.

Acknowledgements. we would like to thank Nadia Busi for the precious suggestions and support.

References

1. F. de Boer, C. Palamidessi. Embedding as a Tool for Language Comparison. In *Information and Computation 108(1)*, 1994.
2. N. Busi, R. Gorrieri. On the computational power of Brane Calculi. *Third Workshop on Computational Methods in Systems Biology*. Edinburgh, 2005.
3. M. Carbone, S. Maffeis. On the Expressive Power of Polyadic Synchronisation in pi-calculus. In *Nordic Journal of Computing* 10(2): 70-98, 2003.
4. L. Cardelli. Brane Calculi - Interactions of Biological Membranes. In *Computational Methods in Systems Biology*, 2004.
5. L. Cardelli, A. D. Gordon. Mobile Ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98.* Springer-Verlag, 1998.
6. L. Cardelli, G. Păun. An universality result for a (mem)brane calculus based on mate/drip operations. In *International Journal of Foundations of Computer Science*. World Scientific Publishing Company, 2005.
7. D. Chiarugi, M. Curti, P. Degano, R. Marangoni. VICE: A VIrtual CEll. *Computational Methods in Systems Biology*. 2004

8. M. Curti, P. Degano, C. T. Baldari. Causal π-Calculus for Biochemical Modelling In *Computational Methods in Systems Biology*. 2003.
9. R. Cleaveland, G. Lüttgen, V. Natarajan. Priority in Process Algebra. In J.A. Bergstra, A. Ponse, S. A. Smolka, editors, *Handbook of Process Algebra*, Elsevier, 2001.
10. V. Danos, C. Laneve. Formal Molecular Biology. In *Theoretical Computer Science 325 (1)*, 2004.
11. V. Danos, S. Pradalier. Projective Brane-calculus. *Computational Methods in Systems Biology: Second International Workshop, CMSB?04, 3082:134?148*. 2004.
12. C. Laneve, F. Tarissan. A simple calculus for proteins and cells In *Proc. of the Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'06)*. 2006.
13. P. Lecca, C. Priami, C. Laudanna, G. Constantin. Predicting cell adhesion probability via the biochemical stochastic pi-calculus. In *Symposium on Applied Computing*. 2004
14. R. Milner. The Polyadic π-Calculus: a Tutorial. In F. L. Hamer, W. Brauer and H. Schwichtenberg, editors, *Logic and Algebra of Specification*. Springer-Verlag, 1993.
15. R. Milner. Communicating and Mobile Systems: The π-Calculus. Cambridge University Press, 1999.
16. U. Nestmann, B.C. Pierce. Decoding Choice Encodings. In *Proc. of the 7th International Conference on Concurrency Theory (CONCUR '96)*. 1996.
17. C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculi. *Mathematical Structures in Computer Science 13(5): 685-719*. 2003.
18. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
19. A. Phillips, L. Cardelli. A correct abstract machine for the stochastic pi-calculus. *Transactions on Computational Systems Biology*. 2005.
20. C. Priami. Stochastic π-calculus. *The Computer Journal 38 (7)*. 1995.
21. C. Priami, P. Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology*, 2004.
22. A. Regev, W. Silverman, E. Shapiro. Representation and simulation of biochemical processes using the π-Calculus process algebra. In *Proc. of the Pacific Symposium on Biocomputing (PSB '01)*. World Scientific Press, 2001.
23. C. Priami, A. Regev, W. Silverman, E. Shapiro. Application of a stochastic passing-name calculus to representation and simulation of molecular processes. *Information Processing Letters 80*. 2001.
24. A. Regev, E. Panina, W. Silverman, L. Cardelli, E. Shapiro. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science*, 2004.
25. C. Versari. Encoding catalytic P systems in π@. In *Proc. of the Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'06)*. 2006.