Dottorato di Ricerca in Informatica
Università di Bologna e Padova
INF/01 INFORMATICA
Ciclo XXI

A core calculus for the analysis and implementation of biologically inspired languages

Cristian Versari

Marzo 2009

Coordinatore:
Simone Martini

Tutore:
Roberto Gorrieri

Abstract

The application of Concurrency Theory to Systems Biology is in its earliest stage of progress. The metaphor of cells as computing systems by Regev and Shapiro [85] opened the employment of concurrent languages for the modelling of biological systems. Their peculiar characteristics led to the design of many bio-inspired formalisms which achieve higher faithfulness and specificity.

In this thesis we present $\pi@$, an extremely simple and conservative extension of the π-calculus representing a keystone in this respect, thanks to its expressiveness capabilities. The $\pi@$ calculus is obtained by the addition of *polyadic synchronisation* and *priority* to the π-calculus, in order to achieve compartment semantics and atomicity of complex operations respectively.

In its direct application to biological modelling, the stochastic variant of the calculus, S$\pi@$, is shown able to model consistently several phenomena such as formation of molecular complexes, hierarchical subdivision of the system into compartments, inter-compartment reactions, dynamic reorganisation of compartment structure consistent with volume variation.

The pivotal role of $\pi@$ is evidenced by its capability of encoding in a compositional way several bio-inspired formalisms, so that it represents the optimal core of a framework for the analysis and implementation of bio-inspired languages. In this respect, the encodings of BioAmbients [88], Brane Calculi [22] and a variant of P Systems [78] into $\pi@$ are formalised. The conciseness of their translation into $\pi@$ allows their indirect comparison by means of their encodings. Furthermore it pro-

vides a ready-to-run implementation of minimal effort whose correctness is granted by the correctness of the respective encoding functions.

Further important results of general validity are stated on the expressive power of priority. Several impossibility results are described, which clearly state the superior expressiveness of prioritised languages and the problems arising in the attempt of providing their parallel implementation. To this aim, a new setting in distributed computing (the *last man standing* problem) is singled out and exploited to prove the impossibility of providing a purely parallel implementation of priority by means of point-to-point or broadcast communication.

Acknowledgements

This thesis represents both the last step of my studies and the synthesis of my first contributions to Research.

I must thank my supervisor Roberto Gorrieri not only for supporting me during these three years but also, in the first place, for having encouraged me to begin this path, as well as warned about all the difficulties and uncertainties that it involves.

I must thank Nadia Busi – who we all miss – of course for her peerless technical support, but above all for having shown me that beyond the professional gratifications, this world can be really enjoyed.

I must thank the referees Luca Cardelli and Jane Hillston for the many corrections and precious suggestions which considerably helped the improvement of this thesis.

I thank my mother for her endless patience and continuous help in everyday life without which I would have hardly succeeded in this work, and my father for having first instilled me the love for science which influenced many important choices during my life, including the one that led me to this.

I thank my dear friend Enrica for her emotional support, encouragement and nearness during the hardest times of these last years: thanks to her, my personal growth turned into a pleasure, instead of a mere necessity.

I finally thank all my colleagues and friends for the lighter and funny moments that helped me work even harder later, when (often!) needed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Scientific Research in the 20^{th} century has been characterised by the flowering of interdisciplinary fields, aiming at the extension of human knowledge by connecting and integrating information, perspectives, methods and tools which derive from different academic disciplines. Representative field of this approach is Systems Biology, whose domain spreads from several branches of Biology such as Biochemistry, Molecular Biology and Physiology to more distant areas, such as Physics, Mathematics, Statistics, Engineering and Informatics.

The study of complex interactions at molecular and cellular level in biological systems constitute the main focus of this recent discipline, with the ambitious task of deeply understanding their behaviour. The unconventional approach adopted subverts the traditional reductionist paradigm in an attempt to embrace the intrinsic complexity of the reality under examination. In other words, while the usual scientific strategy devotes its efforts to describing complex phenomena by grasping the laws that govern their basic constituents, Systems Biology proceeds on the assumption that the emerging behaviour of a complex system cannot be simply characterised by the exhaustive knowledge of its elements: it can only be captured by considering the system as a whole in the full complexity of interaction of its subparts. Therefore, "Systems Biology [...] is about putting together rather than

taking apart, integration rather than reduction." [75]

The promises of this young discipline are difficult to pursue as well as challenging: advances in the detailed knowledge of cellular dynamics are going to produce a deep impact on several research areas of primary importance – from the related fields of Bioinformatics to farther ones like Nanotechnology, all having important effects on many branches of medical research, e.g. Pharmacology, Preventive Medicine – and also affect the development of new disciplines such as Predictive and Personalised Medicine, Synthetic Biology, Natural Computing. On the other hand, such advances are bound to the application of cutting edge technologies or even to the development of newer ones for the collection of huge amounts of data, while new tools, techniques and computational power are needed for their analysis.

In particular, during these last years several efforts have been devoted to the development of many software tools [46, 91, 63, 90, 59, 61] for the analysis of the collected information and formulation of consistent models of biomolecular systems, whose predictive properties may substantially shorten the time required for wet experiments by allowing their realisation *in silico*. The need for model interchange between these software tools underlined for the first time the problem of defining a *common language* able to express the fundamental entities considered by Systems Biology and their interaction laws. The Systems Biology Markup Language (SBML) [52] was the first practical answer to this need, and it constitutes nowadays a widely adopted standard for Systems Biology related software.

Beyond this practical need of data interchange, the definition of SBML presupposed the conception of the first widely accepted theoretical domain where computational models of biological systems could have been expressed: each SBML model can be characterised by several entities, from the simple definition of *chemical species* and their *reaction laws* to the description of the (static) structure of the system by means of *compartments*, all of this completed by *quantitative parameters* related to physical or chemical units of measure, *reaction rates*, compartment sizes, and so on. Without aspiring to completeness, SBML captures the essential entities that any formalism devoted to these bio-modellings should consider.

As to strengthen the interdisciplinary character of this research, the assertion of cells as computing systems by Regev et al. [85] evidenced a new point of conjunction between Systems Biology and Computer Science. The abstraction of molecules as parallel, interacting entities of computation opened Concurrency Theory to a new field of application, by denoting abstract computer languages as optimal candidates for the representation of biomolecular systems. The major practical consequence of that simple metaphor was that the theoretical results, analysis techniques, software tools developed in recent years for the study of concurrent systems could be directly applied or adapted to the study of biological systems, so as to bring insights and explanations on their deepest functioning.

As first working example of biological modelling by means of these languages, a simple biochemical pathway was expressed [86] by means of the π-calculus [69, 70, 68], an abstract calculus for the formalisation of concurrent and *mobile* processes interacting over named channels. The essential primitives of this language turned out to fit very well the basic expressive needs of biochemical modelling: molecules were represented by parallel processes, chemical reactions were modelled as binary synchronisations over channels, simple molecular structures encoded by the sharing of private names. Stochastic simulation by means of Gillespie's algorithm [44, 45] provided the first kind of quantitative analysis applied to these models [89, 84, 60].

The peculiar features of this new setting for the application of concurrent calculi raised the need to design new ad-hoc primitives to be introduced in such languages, in order to better reproduce the structures and mechanisms of interaction typical of this biological world. This design phase flowed into the birth of the first so-called *bio-inspired process calculi* (e.g., [88, 22, 83, 37]), specifically aimed at the modelling of biological phenomena at molecular and cellular level of abstraction.

Nowadays a wide variety of formalisms (e.g., [19, 29, 35, 55, 49, 57, 38, 26, 4, 8]) has been proposed for the representation of biological systems. Several efforts have been directed to broaden the types of analysis to be applied, from stochastic simulation [80, 35, 27, 25, 26, 3] to differential equations [14, 19, 9, 56], from static analysis [74, 7] to causality [36, 48] and even advanced model checking techniques

[28, 30].

On the one hand the expressiveness provided by such formalisms and their specialisation allows a more faithful representation of the biological phenomena of interest, on the other hand it totally departs from the attempt of standardisation pursued with the definition of SBML: even if new likely essential primitives and structures have been identified, it is still far from clear what *the language* of Systems Biology may be – provided that such unique language exists – with respect to the features to be included and how they should be arranged. Definite answers to such issues will require years of work for the design of further formalisms, analysis and comparison of their expressiveness, development of related software tools and consequent evaluation of their effectiveness and usability.

Motivated by the intention to take some significant step in this precise direction, we focus on the idea of grasping the essence of the fundamental primitives of bio-inspired formalisms proposed so far and on the way they may better integrated into a common programming framework: with simplicity in mind, we try to design a basic calculus able to reproduce many of the salient mechanisms of biological modelling considered in several bio-inspired calculi. As a result we obtain far more than expected: a stochastic calculus for the simulation of biological systems with dynamic structure, a low-level language for the modular encoding and comparison of bio-inspired calculi and the outline of a framework for the implementation of the related software tools for analysis and simulation of models written in such calculi — *the π@ language*.

1.1 Contribution

The proper synthesis of biological primitives into one single language must take into account two distinct aspects.

From a biological perspective, the language needs to include a suitable subset of features able to model the most common situations of interest. In this respect concurrency, reactivity, nondeterminism are the basic ingredients of this recipe, con-

sidered by all the cited formal languages. Beyond them, peculiar phenomena such as formation of complexes, subdivision in compartments with dynamical structure, mobility of elements between compartments constitute further characteristics which inspired the definition of many of the first bio-inspired calculi (e.g. [88, 22, 83, 37]) and still guide an increasing number of proposals [31, 57, 21, 54].

From the point of view of computer science, the language must be designed in the best way in order to include and harmonise all the considered features without losing ease of use. An extraordinary virtue would be the possibility to seamlessly introduce even new features that have still to be identified and designed into the upcoming bio-inspired formalisms. Therefore, if the greedy strategy of directly merging all or parts of such calculi into one single formalism [79] can be effective in the immediate period, a foresighted approach must pursue general applicability and provide extreme flexibility in order to be extensible.

The $\pi@$ calculus presented here – whose name is pronounced *pi-at*, like the french word "paillette" – embodies these design principles, by addressing at the same time as high conservativeness as possible, in order to preserve the theoretical results and analysis techniques already conceived and to require a small effort for the adaptation of existing software tools or the development of new ones.

The $\pi@$ language is strongly based on the π-calculus, which represents an optimal starting point for its broad applicability. *Mobility* constitutes the peculiar feature of this calculus, expressed by the capability of estabilishing new communication links between processes thanks to the transmission of new channel names over the existing communication channels. As previously pointed out, this capability fitted the representation of molecular structures but turned out to be insufficient for the proper modelling of higher level structures such as compartments, which have been explicitly formalised later in several bio-inspired calculi.

Instead of following the same high-level approach, we try to obtain the maximum result with the minimum effort by extending the π-calculus in the first instance with *polyadic synchronisation* [18], a communication paradigm which considers channels composed of multiple names in the guise of how Internet domains are represented

by the juxtaposition of names. This simple extension – which does not affect the essence of the π-calculus at all – combined with mobility, gives $\pi@$ the capacity of representing compartments with dynamical structure with respect to both their hierarchical organisation and the possibility of exchanging elements between adjacent compartments.

This low-level approach is completed by enriching $\pi@$ with a general mechanism for the proper encoding of high-level instructions as sequences of low-level operations. While this top-down design strategy has been long consolidated for standard computer programming, in a concurrent setting it raises consistent safety issues due to the unpredictable behaviour of parallel, interacting systems. This problem is radically solved by extending $\pi@$ with *priority* [33], a scheduling scheme which grants the possibility of composing such sequences of operations *atomically*.

In synthesis, $\pi@$ ensues from the addition of polyadic synchronisation and priority to the π-calculus, the first one representing a minimal syntactic extension for the expression of complex biological structures, the second one constituting a general solution for achieving atomicity in concurrent settings, which corresponds to the ability of providing high-level operations at will by the atomic assembly of the existing lower-level instructions.

Such optimistic claims about the virtues of this language need to be supported under several point of views. From a biological perspective, its suitableness to the modelling of complex biological system needs to be demonstrated and its relationship with the existing bio-inspired calculi must be investigated. From a theoretical perspective, the expressive power gained from the addition of the chosen extensions (priority in particular) needs to be analysed in order to discover the full capabilities of the language and to determine its limits.

Some first important answers to such questions are given in this thesis. The expressiveness of priority, the main concept introduced in $\pi@$, is studied and original insights about its strong properties are unveiled, insights which transcend the leading biological thread and constitute relevant results in the field of Concurrency Theory.

The relation between $\pi@$ and some of the bio-inspired formalisms previously

cited is shown in terms of their *encodability* into $\pi@$. In particular, the *modular encodings* of BioAmbients [88], Brane Calculi [22] and a variant of P Systems [78] are formalised. Such encodings constitute the first examples of the wide applicability of the low-level approach adopted here and of the valuable properties of $\pi@$ primitives. The conciseness characterising these encodings gives evidence of the ease of implementation of such formalisms on top of $\pi@$, which fully supports the choice of $\pi@$ as the abstract core of a framework for the quick development of bio-inspired languages.

With respect to biological modelling, $\pi@$ consequently appears at least as expressive as the encoded formalisms. The direct application of $\pi@$ to biological modelling is accomplished by the presentation of a stochastic variant of the calculus, S$\pi@$, whose definition is associated with an extension of one of the most exploited simulation algorithms – Gillespie's [44, 45] – modified so that multiple compartments with varying volumes can be properly taken into account. Examples of modellings are also described and it is shown how some common biological phenomena (e.g. osmosis) can now be easily recovered in this language.

1.2 Structure of the thesis

The thesis is divided into two parts. The first one, from Chap. 2 to Chap. 5 is devoted to the analysis of the expressiveness of $\pi@$, while the second part, from Chap. 6 to Chap. 8 treats the stochastic counterpart of $\pi@$, the S$\pi@$ language.

In particular, Chap. 2 introduces some of the formalisms which are considered in the following chapters to demonstrate the encoding capabilities of $\pi@$: first, the π-calculus is presented and illustrated with an example of simple biological modelling. BioAmbients and Brane Calculi are then formalised and exemplified in the same way. Catalytic P Systems – the variant of membrane systems considered later for the encoding – are finally described.

Chapter 3 investigates the expressiveness of various kinds of priority in a distributed settings. Two different instances of *static* priority are considered, according

to the classification described in [33]: the expressiveness of two languages denoted respectively by *global* and *local* priority is evaluated with respect to both broadcasting and the usual point-to-point communication which characterises, for example, the π-calculus as well as many other calculi considered here. Consistent separation results of general validity are then stated, which give important hints about the expressive power of priority and the price to be paid for its exploitation. The results provided in this chapter were first presented in [98] and are extensively discussed in [97].

In Chap. 4 the $\pi@$ language is presented. Polyadic synchronisation and the type of priority exploited in this language are first introduced, then the calculus is formalised together with one simpler variant – the core-$\pi@$ – limited in the use of such extensions and representing a bridge between $\pi@$ and its stochastic counterpart, S$\pi@$. The application of the calculus to biological modelling is explained by means of a simple example, as for the bio-inspired calculi introduced in Chap. 2.

The encodings of BioAmbients, Brane Calculi and catalytic P systems into $\pi@$ is discussed in Chap. 5. First, the requirements which *suitable encodings* should satisfy are discussed. The ideas that allow such translations are then explained step by step during the formalisation of the respective encoding functions, which are finally formalised. The first definitions of such encodings were presented in [92, 93], while in [99] a gentle and detailed explanation of all the encoding ideas and related technical issues is given.

With Chap. 6 the second part of the thesis begins. A short introduction to Gillespie's stochastic simulation algorithm is given, then the S$\pi@$ calculus is formalised: the extension to multiple compartments of Gillespie's algorithm is defined, followed by an optimised implementation with lower computational complexity. S$\pi@$ syntax and semantics is then presented. Chap. 7 briefly describes a prototype, *SBML-capable* implementation of the MSSA (the Multi-compartment Stochastic Simulator, MSS), while Chap. 8 shows two practical examples of modelling in S$\pi@$, with the results of their stochastic simulation in the MSS. The first multi-compartment extension of Gillespie's algorithm was first presented in [95], and its optimised variant

in [96]. The improved version of this algorithm is discussed in [94].

Chap. 9 closes this work by summarising the results collected throughout the thesis and suggesting the next steps for future research in the direction followed here.

Part I

The π@ calculus

Chapter 2

Background

In this chapter the main languages of interest for the rest of the thesis are introduced.

Section 2.1 is a gentle introduction to the basic features of the π-calculus, in particular of its syntax and reduction semantics via a structural congruence. It also comprises a non-trivial, original example of a biological system modelled in π-calculus: the insulin secretion process of a pancreatic β cell in response to a rise of glucose in the blood. We discuss the limitations of this representation, in particular for the difficulties in handling properly the surrogate of compartments that π-calculus is able to express and for the lack of transactional mechanisms.

BioAmbients and Brane Calculi are introduced in the same way in Section 2.2 and Section 2.3 respectively.

Finally, Sect. 2.4 reports the definition of catalytic P Systems.

2.1 The π-calculus

The π-calculus [69, 70] is a derivative of CCS [64] where parallel processes interact through synchronisation over named channels, with the capability of receiving new channels and subsequently using them for interaction with other processes, in order to model mobility.

Names constitute the basic entities of the calculus. Each name represents a channel which can be used for synchronisation by parallel processes. For example,

the system

$$a(x).P \mid \overline{a}\langle z\rangle.Q \tag{2.1}$$

represents two parallel processes $a(x).P$ and $\overline{a}\langle z\rangle.Q$, the first one ready to receive some datum (whose local name is x) over the channel a, the second one ready to send some datum z over the same channel a. The datum z represents in turn another channel, which can be used by the first process for subsequent communications.

If $\overline{a}\langle z\rangle.Q$ sends z to $a(x).P$, then the subsequent behaviour of the two processes is specified by the expressions Q and P respectively. More precisely, we write that the system of Expr. (2.1) may evolve in the following way:

$$a(x).P \mid \overline{a}\langle z\rangle.Q \quad \rightarrow \quad P\{z/x\} \mid Q \tag{2.2}$$

where $P\{z/x\}$ represents the process P where all the occurrences of the placeholder x have been replaced by z. Here, x is said to be a *bound name*, in opposition to a which is *free*.

The *transition* of the system $a(x).P \mid \overline{a}\langle z\rangle.Q$ to the system $P\{z/x\} \mid Q$ is governed by to *reduction relation* "$\rightarrow$", which states that two processes may exchange data if they are ready to perform input/output respectively over the same channel.

The *nondeterministic choice* between two (or more) possible transitions is denoted by the *choice operator "+"*. For example, in the system

$$a(x).P' + b(y).P'' \mid \overline{a}\langle z\rangle.Q \mid \overline{b}\langle w\rangle.R$$

the first process may undergo two different, equally possible transitions, caused by a synchronisation with the second process or the third one, respectively. The first transition can be written as

$$a(x).P' + b(y).P'' \mid \overline{a}\langle z\rangle.Q \mid \overline{b}\langle w\rangle.R \quad \rightarrow \quad P'\{z/x\} \mid Q \mid \overline{b}\langle w\rangle.R$$

while the second as

$$a(x).P' + b(y).P'' \mid \overline{a}\langle z\rangle.Q \mid \overline{b}\langle w\rangle.R \quad \rightarrow \quad P''\{w/y\} \mid \overline{a}\langle z\rangle.Q \mid R$$

Depending on the occurring transition, the future behaviour of the first process is denoted by $P'\{z/x\}$ or $P''\{w/y\}$ respectively.

Since the order used for enumerating the possible choices is meaningless, i.e. the choice operator is commutative (and associative), we write that

$$a(x).P' + b(y).P'' \quad \equiv \quad b(y).P'' + a(x).P'$$

where "$\equiv$" represents a *congruence relation* between processes that are meant to be characterised by the same behaviour.

Anyway, the choice operator is not the first cause of nondeterminism. As usual for concurrent calculi, the parallelism of the system can produce nondeterministic behaviour. For example, the system

$$a(x).P \mid \overline{a}\langle z\rangle.Q \mid \overline{a}\langle w\rangle.R$$

is subjected to two transitions, depending on the process that will actually perform the output operation on channel a:

$$\begin{aligned} a(x).P \mid \overline{a}\langle z\rangle.Q \mid \overline{a}\langle w\rangle.R \quad &\rightarrow \quad P\{z/x\} \mid Q \mid \overline{a}\langle w\rangle.R \\ a(x).P \mid \overline{a}\langle z\rangle.Q \mid \overline{a}\langle w\rangle.R \quad &\rightarrow \quad P\{w/x\} \mid \overline{a}\langle z\rangle.Q \mid R \end{aligned}$$

It is possible to prevent unwanted interactions between processes by limiting the scope of a name:

$$(\nu\, a)\big(a(x).P \mid \overline{a}\langle z\rangle.Q\big) \mid \overline{a}\langle w\rangle.R \tag{2.3}$$

Thanks to the restriction operator "ν ", the name a in the first two processes represents a private channel between them. Even if the same name a occurs also in the third process, it constitutes a completely different communication channel. Any renaming of the restricted channels (as well as bound names, by a procedure called alpha-conversion) has no effect on the behaviour of the systems. In fact the following expression

$$(\nu\, b)\big(b(x).P \mid \overline{b}\langle z\rangle.Q\big) \mid \overline{a}\langle w\rangle.R$$

is equivalent to Expr. 2.3, since $b(x).P$ and $\bar{b}\langle z\rangle.Q$ are able to exchange data exactly as before.

In order to model recursive behaviour, an operator of *replication* is introduced in the language. A process P preceded by "!" is thought of as being replicated an unlimited number of times. That is

$$! P \equiv P \mid P \mid \cdots$$

The formal definition of π-calculus grammar follows.

Definition 2.1 *Let*

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

$\overline{\mathcal{N}} = \{\overline{x} \mid x \in N\}$

The syntax of π*-calculus is defined in terms of the following grammar:*

$$\begin{array}{lcl} P & ::= & \mathbf{0} \mid \sum_{i\in I} \pi_i.P_i \mid P \mid Q \mid ! P \mid (\nu\, x)P \\ \pi & ::= & \tau \mid x(y) \mid \overline{x}\langle y\rangle \end{array}$$

where

- $\mathbf{0}$ represents the null process;
- $x(y)$ expresses the capability of performing an input on the channel x and receiving a datum which is then bound to the name y;
- $\overline{x}\langle y\rangle$ expresses the capability of sending the name y on the channel x;
- τ is the invisible, uncontrollable action;
- $P \mid Q$ represents the parallel composition of processes;
- $! P$ stands for the unlimited replication of process P;
- $\sum_{i\in I} \pi_i.P_i$ represents the nondeterministic choice between several input/output communication capabilities, denoted also as $\pi_1.P_1 + \pi_2.P_2 + \ldots$;

- $(\nu\ x)P$ represents the scope restriction of the name x to process P.

The full definition of the congruence relation $\equiv$ follows. It depends in turn on the function $\mathrm{fn}(P)$ which returns the set of free names occurring in P.

Definition 2.2 *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*

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

The relation describing the possible transitions of a process is defined in terms of few simple reduction rules, which exploit the congruence relation previously given.

Definition 2.3 *π-calculus semantics is given in terms of the reduction system described by the following rules:*

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

$$\textit{PAR: } \frac{P \ \rightarrow\ P'}{P \mid Q \ \rightarrow\ P' \mid Q} \qquad \textit{RES: } \frac{P \ \rightarrow\ P'}{(\nu\ x)P \ \rightarrow\ (\nu\ x)P'}$$

$$\textit{STRUCT: } \frac{P \equiv Q \quad P \ \rightarrow\ P' \quad P' \equiv Q'}{Q \ \rightarrow\ Q'}$$

The TAU rule represents an internal, unobservable change of state of some process P. The transition of Expr. (2.2) is formalised by rule COMM. The PAR rule describes the meaning of the parallel operator: each process capable of some internal transition, can evolve even when put in parallel with other processes. The RES rule allows transition of processes in presence of restricted names. The key role of restriction (as well as the semantics of the replication) is hidden by rule STRUCT, which states that if two processes are structurally congruent, then they can perform the same transitions.

The capability of encoding the λ-calculus [67] in the π-Calculus implies the Turing-completeness of the π-Calculus. As an alternative proof, we provide the encoding of a very simple class of Random Access Machines (RAMs) which in [71] is shown to be Turing-complete. Each RAM is composed of a finite set of registers $r_1, \ldots, r_n$ holding arbitrary large natural numbers, and of a finite set of indexed instructions $(1 : I_1), \ldots, (m : I_m)$ which represent the program executed by the RAM. Each instruction can be of two kinds:

- $(i : Inc(r_j))$: increment by 1 the contents of the register r_j and execute the next instruction;
- $(i : DecJump(r_j, s))$: if the contents of the register r_j is not zero, then decrease it by 1 and execute the next instruction, otherwise jump to instruction number s.

The internal state of a RAM is described by a configuration $(i, c_1, \ldots, c_n)$ where i is the program counter which indicates the next instruction, while $c_1, \ldots, c_n$ are the values stored in the registers.

Definition 2.4 *A* Random Access Machine *(RAM) R is defined as a pair $R = (I, n)$, where*

$$I = \{(1 : I_1), \ldots, (m : I_m)\}$$

is the set of instructions of R, with $|I| = m$, n is the number of registers of R and each instruction I_i is of two possible kinds:

- $I_i = Inc(r_j) \quad (1 \leq j \leq n)$;
- $I_i = DecJump(r_j, s)) \quad (1 \leq j \leq n)$.

An internal configuration C of R is defined as a state vector

$$C = (i, c_1, \ldots, c_n)$$

with i representing the program counter and $c_1, \ldots, c_n \in \mathbb{N}$ the current values stored in the n registers of R.

The transition function $\rightsquigarrow_R$: $\mathbf{C} \rightarrow \mathbf{C}$ over the set of configurations $\mathbf{C}$ of a RAM R is defined as

$$(i, c_1, \ldots, c_n) \quad \rightsquigarrow_R \quad (i', c'_1, \ldots, c'_n)$$

if

- $I_i = Inc(r_j)$ *and* $i' = i + 1,\ c'_j = c_j + 1,\ c'_p = c_p$ *for* $p \neq j$;
- $I_i = DecJump(r_j, s),\ \ c_j > 0$ *and* $i' = i + 1,\ c'_j = c_j - 1,\ c'_p = c_p$ *for* $p \neq j$;
- $I_i = DecJump(r_j, s),\ \ c_j = c'_j = 0$ *and* $i' = s,\ c'_p = c_p$ *for* $p \neq j$.

Proposition 2.1 *The π-Calculus is Turing-complete.*

Proof: The main program of a RAM R can be encoded as a process M (specified by means of several processes $M_1, \ldots, M_m$, one for every instruction of the RAM), and each register r_j as an independent process R_j, whose internal state is related to the value stored in the register itself. The main program M interacts with each process register R_j over a small set of channels t_j, z_j, n_j, inc_j used respectively to:

- test the state of the register (output from M to R_j on t_j), which can be zero (corresponding to an incoming answer on z_j) or non zero (answer on n_j): in this last case the register is decremented;
- increment the register (output on inc_j);

Each instruction $(i : I_i)$ corresponds to the definition of a replicated process M_i, spawned by an output on the channel m_i:

- in the presence of an increment operation $(i : Inc(r_j))$,

$$M_i \equiv \quad ! \, m_i.\overline{inc}_j.inc_j.\overline{m}_{i+1}$$

- in the presence of a "jump if zero/decrement" operation $(i : DecJump(r_j, s))$, we have

$$M_i \equiv \quad ! \, m_i.\bar{t}_j.(z_j.\overline{m}_s + n_j.\overline{m}_{i+1})$$

Each register R_j is encoded as a stack of processes whose length corresponds to the value stored in the register itself. The first process in the stack reacts to the instructions given by the main process, according to the following definitions:

$$\begin{aligned} Z_j \equiv \quad & t_j.\overline{zloop_j}.\overline{z}_j + \\ & inc_j.(\nu \; a) \; \left(\overline{nloop_j}\langle a\rangle.\overline{inc_j} \;\middle|\; a.\overline{zloop_j}.\overline{a}\right) \\ ZL_j \equiv \quad & ! \; zloop_j.Z_j \end{aligned}$$

Recursive behaviour of Z_j is achieved by guarded replication on $zloop_j$ in a usual way. The first branch of the choice answers a possible query by sending one output on z_j, which signals the value zero stored in the register. One output on inc_j spawns a new process N_j (defined below) linked to a process Z_j, which represents a queue of length one. The process N_j is defined as follows:

$$\begin{aligned} N_j(a) \equiv \quad & t_j.\overline{a}.a.\overline{n}_j + \\ & inc.(\nu \; a') \; \left(\overline{nloop_j}\langle a'\rangle.\overline{inc}_j \;\middle|\; a'.\overline{nloop_j}\langle a\rangle.\overline{a'}\right) \\ NL_j \equiv \quad & ! \; nloop_j(a).N_j(a) \end{aligned}$$

Similarly to Z_j, each process N_j listens on t_j in order to answer on n_j that the value stored in the register is greater than zero, then the register is decremented by sending an output on a, which activates the next process in the stack. The increment operation is exactly the same as for Z_j.

Finally, the internal state $(i, c_1, \ldots, c_n)$ of a RAM R is encoded as

$$\left\{\!\left[(i, c_1, \ldots, c_n) \right]\!\right\}_R \quad = \quad \overline{m}_i \mid R_1^{c_1} \mid \cdots \mid R_n^{c_n} \mid ZL_1 \mid NL_1 \mid \cdots \mid ZL_n \mid NL_n$$

where

$$\begin{aligned} R_j^0 &\triangleq Z_j \\ R_j^k &\triangleq (\nu\ a_1, \ldots, a_k) \\ &\qquad \big(N_j(a_k) \mid a_k.\overline{nloop}\langle a_{k-1}\rangle.\overline{a_k} \mid \cdots \mid \\ &\qquad a_2.\overline{nloop_j}\langle a_1\rangle.\overline{a_2} \mid \\ &\qquad a_1.\overline{zloop_j}.\overline{a_1}\ \big) \qquad (k > 0) \end{aligned}$$

We have that for each configuration $\mathcal{C}'$ of a RAM R immediately reachable from $\mathcal{C}$, that is $\mathcal{C} \rightsquigarrow_R \mathcal{C}'$, there exists a sequence of reductions $M_1 \to \cdots \to M_p$ between their corresponding encodings and vice versa, so that

$$\mathcal{C} \rightsquigarrow_R \mathcal{C}' \qquad \Longleftrightarrow \qquad \left\{\!\left[\mathcal{C} \right]\!\right\}_R \equiv M_1 \to \cdots \to M_p \equiv \left\{\!\left[\mathcal{C}' \right]\!\right\}_R \qquad (p < 8)$$

Furthermore, the encoding fully preserves the determinism of the RAM (in fact only one reduction is possible for each encoded configuration $\left\{\!\left[\mathcal{C} \right]\!\right\}$ and for each of the above intermediate steps M_i between two encoded configurations) and consequently also its divergence.

□

For an extended treatment of the π-calculus we refer to [69, 70, 65, 68].

The key idea behind the modelling of biological systems by means of the π-calculus is that biochemical elements can be seen as parallel processes, and their interaction as communication. In particular, each *molecule* of the system can be represented by a process and its *reaction* with other molecules can be modelled as a communication over a fixed channel. For example, the chemical reaction

$$R: \quad R_1 + R_2 \ \to\ P_1 + P_2$$

where the molecules R_1 and R_2 react according to reaction R, and release P_1 and P_2 as products of the reaction, can be modelled in π-calculus as

$$R_1 \triangleq r.P_1 \qquad R_2 \triangleq \overline{r}.P_2 \qquad\qquad R_1 \mid R_2 \quad \rightarrow \quad P_1 \mid P_2$$

where each process is named as the corresponding molecule, and reaction R is associated with channel r.

Furthermore, the communication of restricted names between π-calculus processes can be exploited for the modelling of local bindings between molecules. If M_1 and M_2 represent two molecules ready to bind, the corresponding expression in π-calculus is

$$M_1 \triangleq (\nu\ b)(\overline{bind}\langle b\rangle.M_1') \qquad M_2 \triangleq bind(x).M_2'$$
$$M_1 \mid M_2 \quad \rightarrow \quad (\nu\ b)(M_1' \mid M_2'\{b/x\}) \qquad\qquad (b \notin \mathrm{fn}(M_2'))$$

where M_1' and $M_2'\{b/x\}$ (and no other process) share the name b after their reaction.

Restriction may be also exploited in order to model *compartments*. A compartment can be thought of as a box separating the external environment from its content. A typical biological example is the cell: its external membrane protects against the dispersion of cell material and regulates the exchange of substances. From an external point of view, the content of the compartment is completely hidden by the compartment itself. Hence we may represent in π-calculus the compartment C as

$$(\nu\ c_1, \ldots, c_k)\big(M_1 \mid \cdots \mid M_l \mid P_1 \mid \cdots \mid P_n\big)$$

where the compartment C is represented by a set of restricted names $c_1, \ldots, c_k$, the content of C is constituted by $P_1, \ldots, P_n$ under the hypothesis that

$$\mathrm{fn}(P_1 \mid \cdots \mid P_n) \subseteq \{c_1, \ldots, c_k\}$$

i.e. the direct interaction of such elements with some external process is prevented by restricting all their channels. The set of processes $M_1 \mid \cdots \mid M_l$ constitutes an interface of the compartment to the external world, that is, in the case of a cell, the membrane itself (and in particular all the transmembrane channels and proteins). These would constitute the only processes enabled to interact with the outer environment.

Modelling Biological Systems in π-calculus

We can now try to model in π-calculus the simple biological system drawn in Fig. 2.1. The figure sketches the insulin secretion process of a pancreatic β cell in response to a rise in glucose in the blood. The glucose is transported inside the cell by a transmembrane channel protein, $GLUT2$. Here it undergoes glycolysis, which leads to the production of pyruvate and ATP. The rise in ATP concentration inhibits the action of K^+ channels, which in turn causes a rise in K^+ ions near the membrane and consequently its depolarisation. Voltage-sensitive Ca^{2+} channels are therefore activated and allow the entry of Ca^{2+} ions, which activate the fusion of the insulin-containing vesicles with the cell membrane (exocytosis), with subsequent dumping of insulin molecules into the blood.

The corresponding π-calculus system will be composed of a compartment C representing the cell, with the shape

$$\begin{aligned}(\nu\, c_1, \ldots, c_k)\big(&CHAN_1 \mid \cdots \mid CHAN_l \\ &\mid MOL_1 \mid \cdots \mid MOL_n \\ &\mid VES \mid \cdots \mid VES\big)\end{aligned}$$

where membrane channels $CHAN_i$ are the only processes aware of names external to the cell itself, unlike the molecules MOL_i and the insulin vesicles VES.

The first elements to model are the glucose molecule and the glucose channel. The entry of glucose inside the cell can be modelled as an interaction between the process representing the glucose molecule and that representing the glucose channel. If the glucose molecule is represented by the process GLU, since its interaction with processes external to the cell must be prevented after its entry, it is worth substituting all the channels of GLU with new restricted names. Furthermore, the GLU process must exhibit some sort of recursive behaviour, because it must be able to synchronise at any time with some glucose channel and then be ready again for the other chemical interactions. Such recursive behaviour may be captured by an expression like

$$GLU \triangleq glutrans.GLU + \overline{glureact}.PYR$$

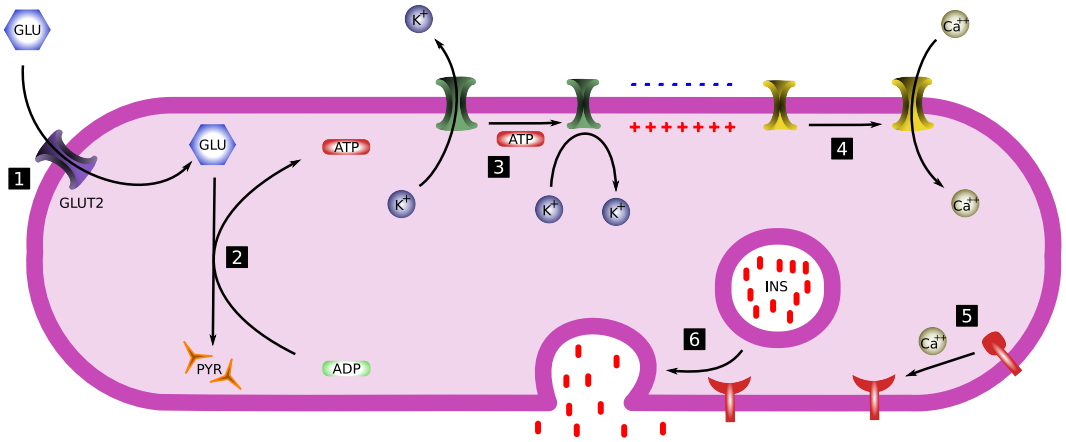


Figure 2.1: **Insulin secretion in pancreatic β cells in response to a rise in glucose concentration**. (1) The rise in glucose concentration of the blood is reflected by a rise in glucose inside the cell, as a consequence of the action of $GLUT2$ glucose transporter. (2) The rise of glucose accelerates the conversion of ADP into ATP, with consequent rise in intracellular ATP concentration. (3) ATP inhibits the action of ATP-sensitive K^+ channels, which reduce the expulsion rate of K^+ ions from the cell. (4) The increasing presence of K^+ depolarises the membrane and triggers the opening of voltage-sensitive Ca^{2+} channels. (5) Fusion proteins are activated by Ca^{2+} ions and (6) trigger the exocytosis of secretory vesicles containing insulin.

where the channel *glutrans* carries the interaction with the glucose channel, while a synchronisation over *glureact* triggers the glycolysis of GLU with consequent production of pyruvate PYR. The recursive behaviour would be obtained by direct recursion (after the transportation, indicated by the synchronisation over *glutrans*, the process becomes GLU again), but since it is not allowed in the version of the π-calculus considered here, it must be encoded by combining replication and restriction, in the following way:

$$GLU \triangleq (\nu\ g)\big(\overline{g}\ \big|\ !\ g.(glutrans.\overline{g} + glureact.PYR)\big)$$

The internal synchronisation over g allows us to spawn a new subprocess

$$(glutrans.\overline{g} + \overline{glureact}.PYR)$$

able to interact nondeterministically either over *glureact* and generate PYR, or synchronise over *glutrans* and then produce another process $\overline{g}$, thus returning to its initial state. The above formalisation still lacks a significant detail: after the interaction over *glutrans* and the entry in the cell, the process GLU must own a new set of restricted names. Such names must be received by GLU during the interaction with the glucose transporter:

$$\begin{aligned} &GLU(glutrans, glureact) \triangleq \\ &\quad (\nu\ g)\big(\overline{g}\langle glutrans, glureact\rangle \ \big|\ !\ g(t,r).(t(nt,nr).\overline{g}\langle nt,nr\rangle + \overline{r}.PYR)\big) \end{aligned} \tag{2.4}$$

Now the internal synchronisation over g spawns a new subprocess $GLUSUB$

$$\begin{aligned} &GLU(glutrans, glureact) \quad \rightarrow \\ &(\nu\ g)\big(GLUSUB \ \big|\ !\ g(t,r).(t(nt,nr).\overline{g}\langle nt,nr\rangle + \overline{r}.PYR)\big) \end{aligned}$$

with

$$GLUSUB \triangleq glutrans(nt,nr).\overline{g}\langle nt,nr\rangle + glureact.PYR$$

where the placeholders t, r have been replaced by $glutrans, glureact$ respectively in consequence of the input/output operation. $GLUSUB$ is ready to react over *glureact* or to trigger the simulated movement of GLU inside some other compartment whose respective names for the reaction and transportation of glucose are received as nr and nt by GLU at the time of the transportation itself.

The glucose transporter $GLUT2$, represented by the process $GLUCHAN$, can be easily modelled as

$$GLUCHAN(gt_{out}, gt_{in}, gr_{in}) \triangleq \ \ !\ \overline{gt_{out}}\langle gt_{in}, gr_{in}\rangle$$

where gt_{out} is the channel for glucose transportation *outside the cell*, gt_{in} is the name used for the same purpose but *inside the cell*, and gr_{in} is the channel for the

glycolysis of GLU inside the cell. As previously noticed, $GLUCHAN$ is a cross-compartment process, since it must be aware of channels both inside and outside the cell.

The effect of glycolysis is the conversion of ADP into ATP. Even if such process involves several other chemical components, for sake of simplicity it can be easily modelled as direct interaction between the ADP and GLU molecules. For the same reason, the inhibition of K^+ channel proteins by ATP can be modelled as direct interaction between the ATP molecule and the K^+ channel proteins, over some name $inhk$:

$$\begin{aligned} ADP(glureact, inhk) &\triangleq \overline{glureact}.ATP(inhk) \\ ATP(inhk) &\triangleq \overline{inhk} \end{aligned}$$

Since the consequent behaviour of the ATP molecule is not relevant for the present purposes, the corresponding process lacks any recursive formalisation.

In order to model the potassium molecule and channel protein, similar considerations can be applied. In particular, the potassium molecule K may be represented as

$$\begin{aligned} &K(ktrans, kreact) \triangleq \\ &\quad (\nu\ k)\big(\overline{k}\langle ktrans, kreact\rangle \mid !\ k(t,r).(t(nt,nr).\overline{k}\langle nt, nr\rangle + \overline{r})\big) \end{aligned}$$

Despite of their similarity, the processes GLU and K differ in the product of their reactions: GLU reduces to PYR, while in the current formalisation K reduces to the null process.

The expression of the K^+ channel protein, $KCHAN$, is a little more complex than the corresponding process $GLUCHAN$. In fact, $KCHAN$ can be inhibited in consequence of ATP binding. In other words, its recursive behaviour shall be interrupted after an interaction over the $inhk$ channel. This effect can be achieved by a slight modification of the recursion used for GLU:

$$\begin{aligned} &KCHAN(kt_{in}, kt_{out}, kr_{out}, inhk) \triangleq \\ &\quad (\nu\ kc)\big(\overline{kc} \mid !\ kc.(inhk + \overline{kt_{in}}\langle kt_{out}, kr_{out}\rangle.\overline{kc})\big) \end{aligned}$$

After the interaction over $inhk$, the process $\overline{kc}$ is not replicated and any further action of the process $KCHAN$ is prevented.

The polarisation of the membrane caused by the rise in K^+ concentration after the inhibition of K^+ channels can be modelled by a fictitious process POL triggered by K^+ ions, which then activates the Ca^{2+} channels:

$$POL(kreact, caact) \triangleq \quad kreact.\overline{caact}$$

The Ca^{2+} channel protein is formalised as the process $CACHAN$, which closely resembles $GLUCHAN$:

$$CACHAN(cat_{out}, cat_{in}, car_{in}, caact) \triangleq \quad caact.!\ \overline{cat_{out}}\langle cat_{in}, car_{in}\rangle$$

The channel is activated only after interaction over $caact$, as required. The calcium process CA is defined exactly as K:

$$\begin{aligned} &CA(catrans, careact) \triangleq \\ &\quad (\nu\ c)\big(\overline{c}\langle catrans, careact\rangle \mid !\ c(t,r).(t(nt,nr).\overline{c}\langle nt, nr\rangle + \overline{r})\big) \end{aligned}$$

The stimulation of the exocytosis by the rise in Ca^{2+} concentration can be modelled as mediated by a docking protein $DOCKP$ which is activated by Ca^{2+} ions and trigger the expulsion of the insulin molecules INS contained in the vesicle VES:

$$DOCKP(careact, dockves, ins_{out}) \triangleq careact.\overline{dockves}\langle ins_{out}\rangle$$

Since the vesicle VES (as well as all the INS molecules inside it) is completely embedded inside the cell, the $DOCKP$ process must also communicate to VES the external name(s) ins_{out} which enables the communication of the processes INS with the environment surrounding the cell.

$$\begin{aligned} VES(dockves) \triangleq \quad &(\nu\ ins_{in})\big(dockves(ins_{out}).!\ ins_{in}.INS(ins_{out}) \\ &\qquad \mid INS(ins_{in}) \mid \cdots \mid INS(ins_{in})\big) \end{aligned}$$

The process VES represents another compartment, where all the names of the embedded processes (INS) are restricted, while only one cross-compartment process

filters their interaction with the external environment. In this very simple model, the INS processes are formalised as

$$INS(ins) \triangleq \quad \overline{ins}$$

Finally, the whole system can be summarised as

$$\begin{aligned} SYS \triangleq \; & GLU(gt_{out}, gr_{out}) \mid K(kt_{out}, kr_{out}) \mid Ca(cat_{out}, car_{out}) \mid \\ & (\nu\ gt_{in}, gr_{in}, kt_{in}, kr_{in}, cat_{in}, car_{in}, inhk, caact, dockves, ins_{in}) \\ & \Big(GLU(gt_{in}, gr_{in}) \mid GLUCHAN(gt_{out}, gt_{in}, gr_{in}) \\ & \mid K(kt_{in}, kr_{in}) \mid KCHAN(kt_{out}, kt_{in}, kr_{in}, inhk) \mid \\ & \mid ADP(gr_{in}, inhk) \mid ATP(inhk) \mid POL(kr_{in}, caact) \\ & \mid Ca(cat_{in}, car_{in}) \mid CACHAN(cat_{out}, cat_{in}, car_{in}, caact) \\ & \mid DOCKP(car_{in}, dockves, ins_{out}) \mid VES(dockves)\Big) \end{aligned}$$

This modelling approach in π-calculus allows us to express to some extent the idea of compartments, but reveals several drawbacks.

The most evident is the need to encode compartment scoping by restricting *all* the free names of the enclosed processes. As an even worse consequence we have that cross-compartment processes must handle all the names of the interacting processes, taking care of the correspondence between distinct names representing the same channel inside different compartments. The model becomes even more complicated when such processes need to create new restricted names and communicate them externally. If we then require compartment operations which affect their nesting structure (merging, splitting, creation, movement of whole compartments) the above approach becomes practically unfeasible, both for the difficulties in name handling and the impossibility of ensuring atomicity of such complex operations without a purely centralised implementation.

2.2 BioAmbients

The BioAmbient calculus rises as an enhancement of the π-calculus in order to overcome the same technical difficulties we encountered during the modelling example of insulin secretion in β cells. More precisely, BioAmbients joins the communication power of the π-calculus and the compartment abstraction given by the Ambient calculus [23, 20]. Compartments are represented by ambients, denoted by square brackets:

$$Sys \triangleq [P \mid Q \mid [R]]$$

The above system Sys is composed of one root ambient containing three elements: the processes P and Q, and another nested ambient $[R]$, which in turns contains another process R.

Processes can communicate in the style of the π-calculus, but communication capabilities are extended to fit the needs of the new setting denoted by ambients. Processes can interact if they lie in the same ambient or in *nearby* ambients, where two ambients are nearby if one of them is directly nested into the other or they are children of the same parent ambient. Therefore, four directions of communications are introduced:

- intra-ambient, for processes inside the same ambient;
- sibling-to-sibling, when processes lie in compartments that are children of the same parent ambient;
- child-to-parent, for a process willing to communicate with a process located in the parent ambient;
- parent-to-child, representing the counterpart of the previous one.

Intra-ambient communication is denoted by the prefix *local*:

$$Sys \triangleq [local\ chan!\{d\}.P \mid local\ chan?\{x\}.Q \mid [R]]$$

The process $local\ chan!\{d\}.P$ is ready to send some datum d over the channel $chan$ to some process which must be located within the same ambient. $local\ chan?\{x\}.Q$

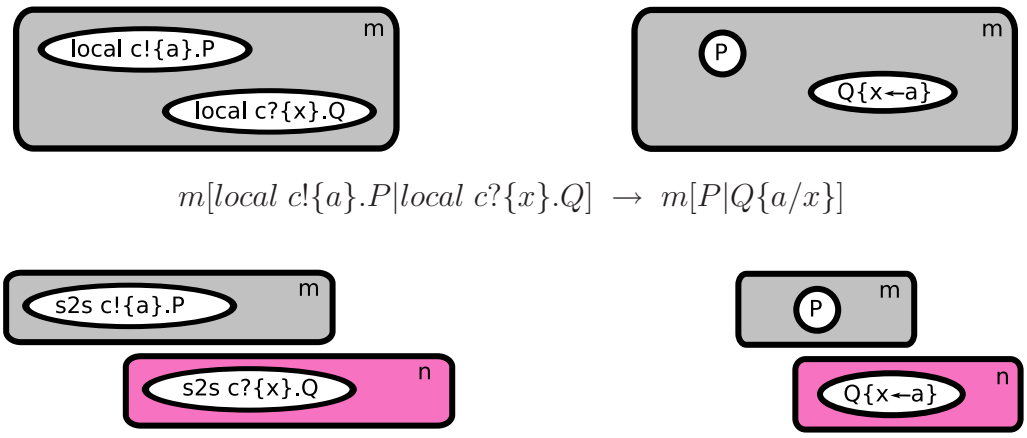


Figure 2.2: Graphic illustration of BioAmbients local and sibling-to-sibling communications.

is a candidate for such synchronisation, since it is listening on the same channel inside the same ambient. For these reason, the above system can reduce in the following way:

$$\begin{aligned} Sys \triangleq\ & [local\ chan!\{d\}.P \mid local\ chan?\{x\}.Q \mid [R]] \quad \rightarrow \\ & [P \mid Q\{d/x\} \mid [R]] \end{aligned}$$

The effect of the communication is the same as for the π-calculus: the name d is received and substituted for the local placeholder x in Q.

Sibling-to-sibling communication is denoted by the prefix $s2s$:

$$\begin{aligned} Sys \triangleq\ & [[s2s\ chan!\{d\}.P \mid Q] \mid [s2s\ chan?\{x\}.R \mid S]] \quad \rightarrow \\ & [[P \mid Q] \mid [R\{d/x\} \mid S]] \end{aligned}$$

Parent-to-child and child-to-parent are complementary, denoted by $c2p$ and $p2c$

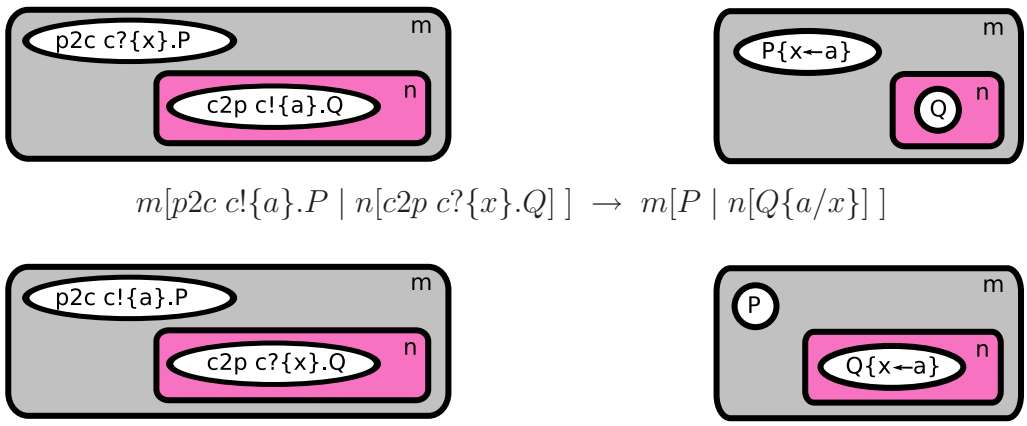


Figure 2.3: Graphic illustration of BioAmbients parent-to-child and child-to-parent communications.

prefixes, and permit both the input/output directions:

$$\begin{aligned} Sys_1 \triangleq\ & [[c2p\ chan!\{d\}.P \mid Q] \mid p2c\ chan?\{x\}.R \mid S] \quad \rightarrow \\ & [[P \mid Q] \mid R\{d/x\} \mid S] \\ Sys_2 \triangleq\ & [[c2p\ chan?\{x\}.P \mid Q] \mid p2c\ chan!\{d\}.R \mid S] \quad \rightarrow \\ & [[P\{d/x\} \mid Q] \mid R \mid S] \end{aligned}$$

In Sys_1 the outer process R receives the datum, while in Sys_2 it sends the datum to the inner process P.

In addition to the above communications, BioAmbients inherits from Mobile Ambients the operations needed to change dynamically the structure of nesting of ambients. Such operations are called *capabilities*. Processes cause the movement of whole ambients across the nesting tree, in agreement with the basic intuitions behind compartment semantics. For example, it is possible to cause the merging of two sibling ambients (children of the same parent ambient) in the following way:

$$\begin{aligned} & [[merge{+}\ n.P \mid Q] \mid [merge{-}\ n.R \mid S] \mid T] \quad \rightarrow \\ & [[P \mid Q \mid R \mid S] \mid T] \end{aligned}$$

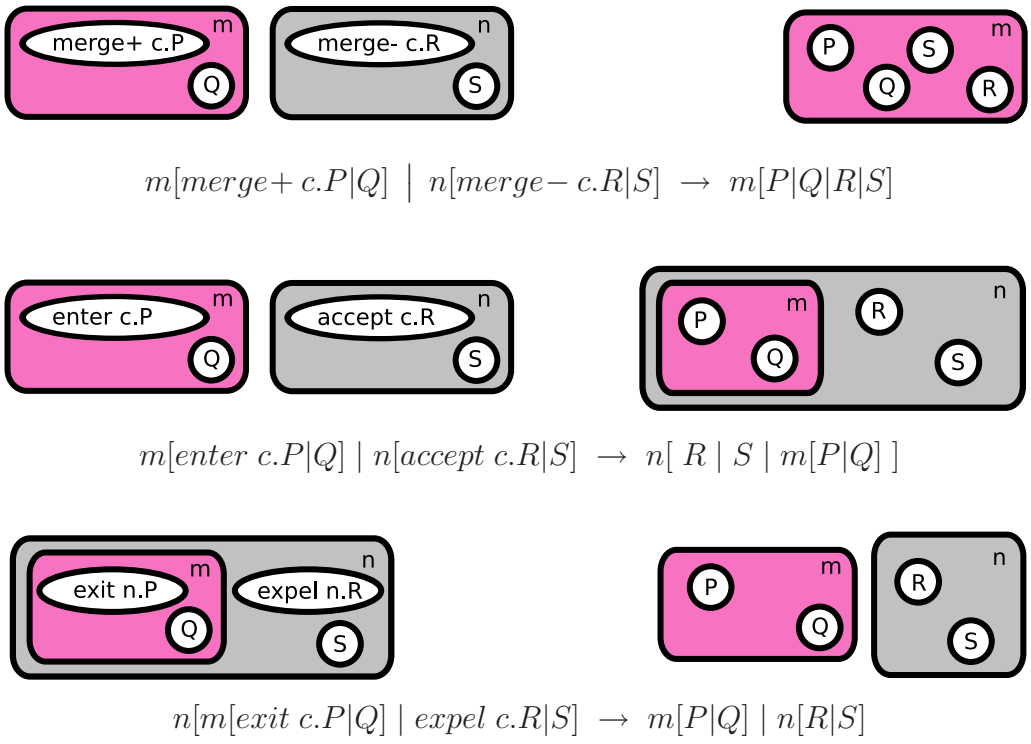


Figure 2.4: Graphic illustration of BioAmbients capabilities.

The *merge*+ and *merge*– capabilities are complementary and can be triggered only if the name n matches.

Ambients can also enter some other sibling ambient

$$[[enter\ n.P\ |\ Q]\ |\ [accept\ n.R\ |\ S]\ |\ T]\quad \rightarrow$$
$$[[[P\ |\ Q]\ |\ R\ |\ S]\ |\ T]$$

or exit their own parent ambient

$$[[[exit\ n.P\ |\ Q]\ |\ expel\ n.R\ |\ S]\ |\ T]\quad \rightarrow$$
$$[[[P\ |\ Q]\ |\ [R\ |\ S]\ |\ T]$$

by additional complementary capabilities, *enter*/*accept* and *exit*/*expel*.

We now give the definitions for the syntax, structural congruence and reduction semantics of BioAmbients, in the same style of π-calculus and $\pi@$ semantics. For further details we refer to [88].

Definition 2.5 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $n, m, p, \ldots \in \mathcal{N}$. The syntax of BioAmbients is defined as*

$$
\begin{array}{rcl}
\pi & ::= & \$n!\{m\} \;\Big|\; \$n?\{m\} \\
\$ & ::= & local \;\Big|\; s2s \;\Big|\; p2c \;\Big|\; c2p \\
M, N & ::= & enter\ n \;\Big|\; accept\ n \;\Big|\; exit\ n \;\Big|\; expel\ n \;\Big|\; merge{+}\ n \;\Big|\; merge{-}\ n \\
P, Q & ::= & (new\ n)P \;\Big|\; P \mid Q \;\Big|\; !\ P \;\Big|\; [P] \;\Big|\; \sum_{i\in I} \pi_i.P_i \;\Big|\; \sum_{i\in I} M_i.P_i
\end{array}
$$

Definition 2.6 *The congruence relation $\equiv$ is defined as the least congruence satisfying the following rules:*

$$
\begin{array}{rl}
P|Q \equiv Q|P & (P|Q)|R \equiv P|(Q|R) \\
P|\mathbf{0} \equiv P & [\mathbf{0}] \equiv \mathbf{0} \\
!\mathbf{0} \equiv \mathbf{0} & !P \equiv P|!P \\
(new\ n)\mathbf{0} \equiv \mathbf{0} & (new\ n)(new\ m)P \equiv (new\ m)(new\ n)P \\
(new\ n)(P|Q) \equiv P|(new\ n)Q & \quad if\ n \notin \mathrm{fn}(P) \\
(new\ n)[P] \equiv [(new\ n)P] & \\
\$n?\{m\}.P \equiv \$n?\{p\}.P\{p/m\} & \quad if\ p \notin \mathrm{fn}(P) \\
(new\ n)P \equiv (new\ m)P\{m/n\} & \quad if\ m \notin \mathrm{fn}(P)
\end{array}
$$

where $\mathrm{fn}(P)$ *is naturally extended to BioAmbients processes.*

Definition 2.7 *BioAmbients semantics is given in terms of the reduction system described by the following rules:*

$$
\begin{array}{rcl}
[(T + enter\ n.P)|Q]|[T' + accept\ n.R)|S] & \rightarrow & [[P|Q]|R|S] \\
[[(T + exit\ n.P)|Q]|(T' + expel\ n.R)|S] & \rightarrow & [P|Q]|[R|S] \\
[(T + merge{+}\ n.P)|(Q]|[T' + merge{-}\ n.R)|S] & \rightarrow & [P|Q|R|S] \\
(T + local\ n!\{m\}.P)|(local\ n?\{p\}.Q + T') & \rightarrow & P|Q\{m/p\} \\
(T + p2c\ n!\{m\}.P)|[(c2p\ n?\{p\}.Q + T')|R] & \rightarrow & P|[Q\{m/p\}|R] \\
[R|(T + c2p\ n!\{m\}.P)]|(p2c\ n?\{p\}.Q + T') & \rightarrow & [R|P]|Q\{m/p\} \\
[R|(T + s2s\ n!\{m\}.P)]|[(s2s\ n?\{p\}.Q + T')|S] & \rightarrow & [R|P]|[Q\{m/p\}|S]
\end{array}
$$

$$\frac{P \rightarrow Q}{(\nu\, n)P \rightarrow (\nu\, n)Q} \qquad \frac{P \rightarrow Q}{[P] \rightarrow [Q]} \qquad \frac{P \rightarrow Q}{P|R \rightarrow Q|R} \qquad \frac{P \equiv P' \quad P \rightarrow Q \quad Q \equiv Q'}{P' \rightarrow Q'}$$

The above reduction rules are graphically illustrated in Fig. 2.2, 2.3, 2.4.

2.2.1 Modelling the insulin example in BioAmbients

The example of Fig. 2.1 can be exploited again to explain the basic modelling ideas which may be applied in BioAmbients.

Since BioAmbients embeds directly the π-calculus (we can obtain a straightforward translation just by substituting each π input/output operation with a *local* communication) we may start sketching the modelling of the GLU molecule of Expr. (2.4) in the following way:

$$GLU(\ldots) \triangleq \quad (\nu\, g)(\mathit{local}\ g!\{\ldots\} \mid !\ \mathit{local}\ g?\{\ldots\}.GLU'(g,\ldots))$$

GLU' should both listen for a possible transportation of the molecule into another compartment and communicate its ability of reacting as glucose. We may then try to take advantage of the multiple directions of communication of BioAmbients and use just a unique name glu in the *local* direction for reacting, and in the $c2p$ and $p2c$ directions for modelling the movement to a new compartment:

$$\begin{aligned} GLU'(g, glu, \ldots) \quad \triangleq \quad & \mathit{local}\ glu!\{\}.PYR(\ldots) + \\ & p2c\ glu?\{g', glu', \ldots\} + \\ & c2p\ glu?\{g', glu', \ldots\} \end{aligned}$$

Unfortunately, in this way we cannot exploit at all the abstraction of ambient nor the directions of communication, because we are still modelling the localisation of processes by means of names: even if GLU' receives some new set of names after the $p2c$ or $c2p$ communication, it is not able to change ambient without some *exit* or *enter* capability. Consequently, in order to take some advantage from the novel primitives introduced in BioAmbients, we must forget the idea of name as means to model mobility or localisation, and leave this job to ambients: even a simple molecule like GLU should be modelled as an ambient on its own. Furthermore it

may be possible to exploit the way ambients move as a whole in order to simplify the expression for GLU:

$$GLU \triangleq \quad [!\, enter\ glu \mid !\, exit\ glu \mid c2p\ glu?\{\}.PYR]$$

The above formalisation embeds the idea that the movement into a new ambient and the chemical reactivity are completely orthogonal and independent. The *exit glu* capability is superfluous, since in the simple model of the system we are considering the glucose molecules are not going to leave the cell once they entered, but it is valid in general for any molecule or element undergoing some sort of "passive" conveyance into other compartments without knowledge of its direction. The movement of the ambient as a whole preserves the integrity of the process even if composed of several parallel subprocesses, but the previous expression does not describe correctly the real behaviour of the glucose molecule: in fact, after its degradation into PYR, the GLU process is still able to be transported across compartments, even if the molecule itself may not exist anymore. Consequently we are forced to exploit the same expedient used for π-calculus and $\pi@$ modelling, even in presence of ambients:

$$\begin{aligned} GLU \triangleq \quad [local\ glu!\{\} \mid !\, local\ glu?\{\}.(&enter\ glu.local\ glu!\{\} + \\ &exit\ glu.local\ glu!\{\} + \\ &c2p\ glu?.PYR)] \end{aligned}$$

Now the movement of the molecule is allowed in either direction and disabled after its degradation. It is worth noticing how the introduction of multiple communication directions (*local*, $s2s$, $p2c$, $c2p$) and capabilities (*merge*, *enter*/*accept*, *exit*/*expel*) reduces the number of channel names needed: each name embeds in fact seven distinct interactions. The scoping induced by ambients allows also to avoid the use of restriction, at least in this simple case.

The corresponding expression for the glucose channel becomes very simple:

$$GLUCHAN \triangleq \quad !\, accept\ glu$$

Even if the previous expression of GLU is correct intuitively, the mixing of communications and capabilities is not allowed in BioAmbients. A slight correction

allows us to overcome this issue:

$$GLU \triangleq \quad [local\ glu!\{\} \mid !\ local\ glu?\{\}.(s2s\ gludock!\{\}.enter\ glu.local\ glu!\{\}+ \\ c2p\ gludock!\{\}.exit\ glu.local\ glu!\{\}+ \\ c2p\ glu!.PYR)]$$

and $GLUCHAN$ must be corrected accordingly:

$$GLUCHAN \triangleq \quad !\ s2s\ gludock?\{\}.accept\ glu$$

The direction $s2s$ is justified by the structure of the system: the GLU molecule external to the cell is an ambient sibling of the ambient represented by the cell itself, where the process $GLUCHAN$ resides.

The formalisation of GLU as an ambient affects the expression of ADP. The *local* reaction (inside the cell) of glycolysis becomes an inter-ambient communication, reflected by the $p2c$ direction:

$$ADP \triangleq \quad p2c\ glu?\{\}.ATP$$

The interaction of ATP can be instead considered *local*, provided that the potassium channel is not modelled as an ambient:

$$ATP \triangleq \quad local\ inhk!\{\}$$

Under this hypothesis, the process $KCHAN$ is not substantially different from the corresponding π-calculus expression:

$$KCHAN \triangleq \quad (\nu\ kc)(local\ kc!\{\} \mid !\ local\ kc?\{\}.(local\ inhk?\{\} \\ + p2c\ kdock!\{\}.expel\ k\{\}.local\ kc!\{\}))$$

The loop which spawns a new $KCHAN$ subprocess is disabled after the inhibition by the ATP molecule. The expulsion of potassium K from the current ambient is

modelled in agreement with the previous considerations about the molecule GLU:

$$\begin{aligned} K \triangleq \quad & [local\ k!\{\} \mid !\ local\ k?\{\}.(s2s\ kdock!\{\}.enter\ k.local\ k!\{\}+ \\ & \qquad c2p\ kdock!\{\}.exit\ k.local\ k!\{\}+ \\ & \qquad c2p\ k!)] \end{aligned}$$

The other processes can be encoded by following similar considerations:

$$\begin{aligned} POL \triangleq \quad & p2c\ k?\{\}.local\ act!\{\} \\ CACHAN \triangleq \quad & local\ act?\{\}.!\ s2s\ cadock!\{\}.accept\ ca \\ CA \triangleq \quad & [local\ ca!\{\} \mid !\ local\ ca?\{\}. \\ & \qquad (s2s\ cadock!\{\}.enter\ ca.local\ ca!\{\}+ \\ & \qquad c2p\ cadock!\{\}.exit\ ca.local\ ca!\{\}+ \\ & \qquad c2p\ ca!\{\})] \\ DOCKP \triangleq \quad & p2c\ ca?\{\}.expel\ dockves \end{aligned}$$

The encoding of the vesicle VES requires more attention. In the biological model, the insulin molecules *never* lie inside the cell. Therefore, if in BioAmbients VES is represented by an ambient, one way to represent the operation would be to make the VES ambient exit the cell and afterwards dump all the insulin molecules INS in the blood:

$$VES \triangleq \quad [exit\ dockves.!\ expel\ ins \mid [exit\ ins.INS] \mid \cdots \mid [exit\ ins.INS]]$$

However, this encoding does not exploit the expressive power of ambients. A smarter approach would merge the content of VES with the ambient corresponding to the blood vessel after the expulsion of the vesicle from the cell:

$$VES \triangleq \quad [exit\ dockves.merge+\ insves \mid [INS] \mid \cdots \mid [INS]]$$

This encoding requires that some complementary process is ready for the merge operation in the parent ambient of the cell:

$$VESMERGE \triangleq \ !\, merge- \ insves$$

It is worth remarking that in both cases the exocytosis process is *not modelled atomically*. Finally, the system is given by the following expression:

$$\begin{aligned} SYS \triangleq \ & GLU \mid K \mid CA \mid VESMERGE \mid \\ & [GLU \mid GLUCHAN \mid K \mid KCHAN \mid ADP \mid ATP \mid POL \mid \\ & \ CA \mid CACHAN \mid DOCKP \mid VES] \end{aligned}$$

Thanks to the introduction of ambients, there is no need to keep explicit trace of the free names of each process. Unfortunately, possible problems emerging from the need of atomicity for complex operations are not resolved, exactly as in π-calculus.

2.3 Brane Calculi

The peculiar spatial rearrangement typical of biological membranes inspired the definition of Brane Calculi [22]. Here membranes constitute both the boundaries of compartments and the place where "computation" happens, that is where concurrent processes are thought to be located and to interact with the surrounding environment. Membranes are denoted by $(\!|\ \cdot\ |\!)$ and can be nested, exactly like ambients. In the following expression, the system S is composed of an outer membrane whose behaviour is specified by σ:

$$S \quad \triangleq \quad \sigma(\!|\ P \circ \rho(\!|\ Q\ |\!) \circ R\ |\!) \tag{2.5}$$

The outer membrane contains another membrane, whose behaviour is specified by ρ and whose content is Q.

S, P, Q, R are called *systems*: each system represents a collection of zero or more membranes that may be nested as we have just seen or composed in parallel by means of the operator $\circ$, like the three systems P, $\rho(\!|\ Q\ |\!)$ and R. Systems specify

the global structure of each Brane expression, i.e. the shape of the tree of nested or sibling membranes.

σ and ρ are called *(mem)branes*: each brane specifies the behaviour of the membrane with respect to the other membranes, by indicating which type of *actions* may be performed. For example, if

$$\sigma \triangleq exo^{\perp}.\sigma' \qquad \rho \triangleq exo.\rho'$$

the system of Expr. (2.5) becomes

$$S \triangleq exo^{\perp}.\sigma'(\!| P \circ exo.\rho'(\!| Q |\!) \circ R |\!) \tag{2.6}$$

σ specifies that the outer membrane is ready to perform an $exo^{\perp}$ action, while ρ is ready to perform the complementary exo action. The $exo/exo^{\perp}$ reduction formalises exocytosis, corresponding to the same phenomenon described in the example of insulin secretion. In consequence of the exocytosis, the content Q of the inner membrane is expelled out of the external membrane, and the two membranes are merged together:

$$S \rightarrow S' \qquad S' \triangleq Q \circ \sigma'|\rho'(\!| P \circ R |\!) \tag{2.7}$$

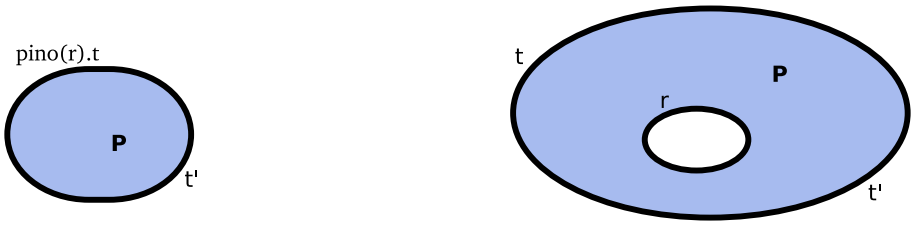
The expression $\sigma'|\rho'$ denotes the parallel composition of branes, where the '$|$' operator is kept distinct from the '$\circ$' operator for the parallel composition of systems. The peculiar behaviour of exocytosis preserves *bitonality*, that is the parity of the level of nesting of each Brane process with respect to the tree structure of membranes. In fact, consider the level of nesting of Q before and after the reduction: if we match such level with the number of membranes that surround a given process, then Q passes from level 2 to level 0 of nesting. Any $exo/exo^{\perp}$ operation will never move some process P from an odd level to an even one, or vice versa. This property is preserved not only by the $exo/exo^{\perp}$ action, but by all the actions of Brane.

The $phago/phago^{\perp}$ reduction causes the inverse effect of the exocytosis: an external process is engulfed by a sibling membrane by surrounding it with a portion of the membrane itself. For example, if

$$Q \triangleq phago.\gamma(\!| T |\!) \qquad \rho' \triangleq phago^{\perp}(\tau).\rho''$$

$$(\!|\ exo^{\perp}.t \mid t'(\!|exo.s \mid s'(\!|P|\!) \circ Q|\!)\ |\!) \quad \rightarrow \quad (\!|\ P \circ s \mid s' \mid t \mid t'(\!|Q|\!)\ |\!)$$

$$phago.s|s'(\!|P|\!) \circ phago^{\perp}(r).t|t'(\!|Q|\!) \quad \rightarrow \quad t|t_0(\!|r(\!|s|s'(\!|P|\!)|\!) \circ Q|\!)$$



$$pino(r).t|t'(\!|P|\!) \quad \rightarrow \quad t|t'(\!|P \circ r(\!|\!|\!)|\!)$$

Figure 2.5: Graphic illustration of Brane reduction rules.

then the system S' of Expr. (2.7) becomes

$$S' \quad \triangleq \quad phago.\gamma(\!|\ T\ |\!) \circ \sigma'|phago^{\perp}(\tau).\rho''(\!|\ P \circ R\ |\!)$$

and can undergo the following reduction:

$$S' \quad \rightarrow \quad \sigma'|\rho''(\!|\ P \circ R \circ \tau(\!|\ \gamma(\!|\ T\ |\!)\ |\!)\ |\!)$$

The external system $\gamma(\!|T|\!)$ has been engulfed and surrounded by the membrane τ, which is thought of as a portion of the original external membrane of S'.

The *pino* action is the simplest operation on membranes: it corresponds to the inward bending of a membrane which produces a new internal membrane without

any content:

$$pino(\sigma).\rho(\!|\ P\ |\!) \quad \rightarrow \quad \rho(\!|\ P\ \circ\ \sigma(\!|\ |\!)\ |\!)$$

The *pino* action has no complementary $pino^{\perp}$ co-action.

In [22], two calculi are presented: the phago-exo-pino and the mate-bud-drip variants.

A study on the relative expressive power of the two variants is reported in [13], where it is shown that the phago-exo-pino calculus is strictly more expressive than the mate-bud-drip one. Therefore, only the phago-exo-pino variant is considered here.

The formal definition of Brane follows, given in terms of a reduction semantics exactly as for the previous calculi.

Definition 2.8 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $n, m, p, \ldots \in \mathcal{N}$. The syntax of Brane is defined as*

$$\begin{array}{rcl} P, Q & ::= & \diamond \mid P \circ Q \mid\ !P \mid \sigma(\!|P|\!) \\ \sigma, \tau & ::= & \mathbf{0} \mid \sigma|\tau \mid\ !\sigma \mid a.\sigma \\ a & ::= & phago_n \mid phago_n^{\perp}(\sigma) \mid exo_n \mid exo_n^{\perp} \mid pino(\sigma) \end{array}$$

Definition 2.9 *The congruence relation $\equiv$ is defined as the least congruence satisfying the following rules:*

$$\begin{array}{cc} P \circ Q \equiv Q \circ P & \sigma|\tau \equiv \tau|\sigma \\ P \circ (Q \circ R) \equiv (P \circ Q) \circ R & \sigma|(\tau|\rho) \equiv (\sigma|\tau)|\rho \\ P \circ \diamond \equiv P & \sigma|\mathbf{0} \equiv \sigma \\ !\diamond \equiv \diamond & !\mathbf{0} \equiv \mathbf{0} \\ !(P \circ Q) \equiv\ !P \circ\ !Q & !(\sigma|\tau) \equiv\ !\sigma|!\tau \\ !!P \equiv\ !P & !!\sigma \equiv\ !\sigma \\ !P \equiv P \circ\ !P & !\sigma \equiv \sigma|!\sigma \\ \mathbf{0}(\!|\ \diamond\ |\!) \equiv \diamond & \end{array}$$

$$\begin{array}{rlrl} P \equiv Q \Longrightarrow & P \circ R \equiv Q \circ R & \sigma \equiv \tau \Longrightarrow & \sigma|\rho \equiv \tau|\rho \\ P \equiv Q \Longrightarrow & !P \equiv !Q & \sigma \equiv \tau \Longrightarrow & !\sigma \equiv !\tau \\ P \equiv Q \wedge \sigma \equiv \tau \Longrightarrow & \sigma(\!|P|\!) \equiv \tau(\!|Q|\!) & \sigma \equiv \tau \Longrightarrow & a.\sigma \equiv a.\tau \end{array}$$

Definition 2.10 *Brane semantics is given in terms of the reduction system described by the following rules:*

$$\frac{P \to Q}{P \circ R \to Q \circ R} \qquad \frac{P \to Q}{\sigma(\!|P|\!) \to \sigma(\!|Q|\!)} \qquad \frac{Q \equiv P \quad P \to P' \quad P' \equiv Q'}{Q \to Q'}$$

$$\begin{array}{rcl} phago_n.\sigma|\sigma_0(\!|P|\!) \circ phago_n^{\perp}(\rho).\tau|\tau_0(\!|Q|\!) & \to & \tau|\tau_0(\!|\rho(\!|\sigma|\sigma_0(\!|P|\!)|\!) \circ Q|\!) \\ exo_n^{\perp}.\tau|\tau_0(\!|exo_n.\sigma|\sigma_0(\!|P|\!) \circ Q|\!) & \to & P \circ \sigma|\sigma_0|\tau|\tau_0(\!|Q|\!) \\ pino(\rho).\sigma|\sigma_0(\!|P|\!) & \to & \sigma|\sigma_0(\!|\rho(\!|\diamond|\!) \circ P|\!) \end{array}$$

The above reduction rules are graphically illustrated in Fig. 2.5. For further details on Brane calculi we refer to [22].

2.3.1 Modelling the insulin example in Brane

In order to allow the reader to become a little familiar with the Brane language, we now show how the insulin secretion model can be expressed in this calculus. The explicit presence in the target model of molecules and molecule channels makes unfeasible its expression in the core phago-exo-pino variant of Brane that we formalised in the previous section.

Therefore we consider an additional class of actions which are proposed in [22] in order to handle explicitly sets of molecules, reactions and cross-membrane conveyance. The grammar of processes is then extended with molecules $m_1, \ldots, m_n$ and multisets of molecules

$$p, q \triangleq m_1 \circ \ldots \circ m_k$$

Multisets of molecules can be used to define *molecular reactions*, which constitute an additional class of Brane actions and have the following shape:

$$p_1(p_2) \rightrightarrows q_1(q_2)$$

with p_1, p_2, q_1, q_2 multisets of molecules. Like the other actions, molecular reactions are thought to be located on membranes and are part of branes. p_1 and p_2 represent the multisets of molecules which must be present outside and inside the membrane respectively, in order to allow the molecular reaction to occur. The multiset q_1 represents the molecules released outside the membrane after the reaction, while q_2 the molecules released inside it.

A cross-membrane channel like the calcium channel of Fig. 2.1 can be easily modelled as follows:

$$S \quad \triangleq \quad CA^{2+} \quad \circ \quad CA^{2+} \quad \circ \quad !CA^{2+}() \rightrightarrows (CA^{2+})(\!| \ P \ |\!)$$

The molecular reaction $!CA^{2+}() \rightrightarrows (CA^{2+})$ is enabled independently of the molecules present inside the membrane, and requires (at least) one molecule of CA^{2+} to be located outside of it. After the reaction, the molecule is released inside the membrane while disappears outside of it:

$$S \quad \rightarrow \quad CA^{2+} \quad \circ \quad !CA^{2+}() \rightrightarrows (CA^{2+})(\!| \ CA^{2+} \ \circ \ P \ |\!)$$

The chemical reaction

$$m_1 + m_2 \quad \rightarrow \quad m_3 + m_4$$

where the molecules m_1, m_2 appear as reactants and m_3, m_4 as products of the reaction, may be represented in Brane by the following action if we model the reaction as happening on the outer surface of the membrane:

$$m_1 \circ m_2() \rightrightarrows m_3 \circ m_4()$$

or

$$(m_1 \circ m_2) \rightrightarrows (m_3 \circ m_4)$$

if we want it to be located on the inner surface.

The explicit handling of molecular reactions in the calculus allows us to model the processes GLU, ATP, ADP, PYR, CA, K and INS as simple molecules. Their

behaviour is completely passive and specified by the molecular reactions present in the other processes.

The cross-membrane channel $GLUCHAN$, whose function is the conveyance of glucose inside the cell, is expressed straightforwardly:

$$GLUCHAN \quad \triangleq \quad !GLU() \rightrightarrows (GLU)$$

Glycolysis can be modelled as molecular reaction occurring inside the cell membrane:

$$GLYCOL \quad \triangleq \quad !(GLU \circ ADP) \rightrightarrows (ATP \circ PYR)$$

The encoding of the potassium channel requires it to be deactivated after its binding to some ATP molecule. The simplest way to obtain this behaviour with a single molecular reaction is the introduction of a fictitious catalyst $KCHANCAT$ present inside the cell, which allows the channel process $KCHAN$ to move the potassium ions across the membrane as long as it does not disappear:

$$KCHAN \quad \triangleq \quad !\,(KCHANCAT \circ K) \rightrightarrows K(KCHANCAT)$$

Such catalyst disappears in consequence of the presence of ATP:

$$KCHANINH \quad \triangleq \quad !\,(KCHANACT, ATP) \rightrightarrows ()$$

The process of polarisation of the membrane POL can be modelled as molecular reaction as well:

$$POL \quad \triangleq \quad (K) \rightrightarrows (ACT)$$

ACT is another fictitious molecule which activates the calcium channel:

$$CACHAN \quad \triangleq \quad (ACT) \rightrightarrows ().!\,CA() \rightrightarrows (CA)$$

The exocytosis of the vesicles is now rendered straightforwardly by the $exo/exo^{\perp}$ reduction:

$$VES \quad \triangleq \quad exo(\!|\, INS \circ \cdots \circ INS \,|\!)$$

Obviously, on the external membrane there must be some process ready to execute the complementary action:

$$DOCKP \quad \triangleq \quad (CA) \rightrightarrows ().exo^{\perp}$$

Finally, the system can be expressed as follows:

$$\begin{aligned} SYS \quad \triangleq \quad & GLU \circ K \circ CA \circ \\ & \sigma(\!| \, KCHANACT \circ K \circ ADP \circ ATP \circ CA \circ VES \, |\!) \end{aligned}$$

with

$$\begin{aligned} \sigma \quad \triangleq \quad & GLUCHAN \mid GLYCOL \mid KCHAN \mid KCHANINH \mid \\ & POL \mid CACHAN \mid DOCKP \end{aligned}$$

The addition of molecular reactions makes Brane a powerful language for biological modelling. This is evident in the formalisation of the previous example, which perfectly suited the bitonal properties of Brane. However, the expression of behaviours that are not included in the design of the calculus (like the calcium channel $CACHAN$, or other compartment-related operations which do not preserve bitonality) requires either the extension of the language with additional primitives or the introduction of intermediate, fictitious elements which may lead again to the same atomicity problems encountered during the modelling with π-calculus or BioAmbients.

2.4 P systems

Quoting from [78]:

> "The essential ingredient of a P system is its membrane structure, which can be a hierarchical arrangement of membranes, like in a cell (hence described by a tree), or a net of membranes (placed in the nodes of a graph), like in a tissue, or in a neural net. The intuition behind the notion of a membrane is that from biology, of a three-dimensional

> vesicle, but the concept itself is generalised/idealised to interpreting a membrane as a separator of two regions (of the Euclidean space), a finite *inside* and an infinite *outside*, also providing the possibility of a selective communication among the two regions.
>
> The variety of suggestions from biology and the range of possibilities to define the architecture and the functioning of a membrane-based-multiset-processing device are practically endless – and already the literature of membrane computing contains a very large number of models."

The high number of P systems models imposes a choice of which is most similar to process calculi and initially easier to encode: catalytic P systems have been chosen for the simplicity of evolution rules and static structure of the membrane tree. Below, preliminary definitions are recalled, then a formal definition of catalytic P systems is given.

Definition 2.11 *Given a set S, a* finite multiset *over S is a function $m : S \to \mathbb{N}$ such that the set $dom(m) = \{s \in S \mid m(s) \neq 0\}$ is finite. The* multiplicity *of an element s in m is given by the natural number $m(s)$. The set of all finite multisets over S, denoted by $\mathcal{M}_{fin}(S)$, is ranged over by m. A multiset m such that $dom(m) = \emptyset$ is called* empty *. The empty multiset is denoted by $\emptyset$.*

Given the multiset m and m', $m \subseteq m'$ holds if $m(s) \leq m'(s)$ for all $s \in S$ while $\oplus$ denotes their multiset union*: $m \oplus m'(s) = m(s) + m'(s)$. The operator $\setminus$ denotes* multiset difference*: $(m \setminus m')(s) =$ if $m(s) \geq m'(s)$ then $m(s) - m'(s)$ else 0. The* scalar product*, $j \cdot m$, of a number j with m is $(j \cdot m)(s) = j \cdot (m(s))$. The cardinality of a multiset is the number of occurrences of elements contained in the multiset: $|m| = \sum_{s \in S} m(s)$.*

Some basic definitions on strings, cartesian products and relations are then given.

Definition 2.12 *A string over S is a finite (possibly empty) sequence of elements in S. The length of a string is the number of occurrences of elements contained in S. With S^* we denote the set of strings over S, and $u, v, w, \ldots$ range over S.*

Given a string $u = x_1 \dots x_n$, the multiset corresponding to u is defined as follows: for all $s \in S$, $m_u(s) = |\{i \mid x_i = s \wedge 1 \leq i \leq n\}|$. With abuse of notation, we use u to denote also m_u.

The definition of membrane structure follows.

Definition 2.13 *Given the alphabet $V = \{[,]\}$, the set MS is the least set inductively defined by the following rules:*

- $[\ ] \in MS$
- *if $\mu_1, \mu_2, \dots, \mu_n \in MS$, $n \geq 1$, then $[\mu_1 \dots \mu_n] \in MS$*

We define the following relation over MS: $x \sim y$ if and only if the two strings can be written in the form $x = [_1 \dots [_2 \dots]_2 \dots [_3 \dots]_3 \dots]_1$ and $y = [_1 \dots [_3 \dots]_3 \dots [_2 \dots]_2 \dots]_1$ (i.e., if two pairs of parenthesis that are neighbours can be swapped together with their contents).

The set $\overline{MS}$ of membrane structures *is defined as the set of equivalence classes with respect to the relation $\sim^*$.*

We call a *membrane* each matching pair of parenthesis appearing in the membrane structure. A membrane structure μ can be represented as a Venn diagram, in which any closed space (delimited by a membrane and by the membranes immediately inside) is called a *region* of μ.

Definition 2.14 *A* catalytic P system *(of degree d, with $d \geq 1$) is a construct*

$$\Pi = (V, C, \mu, w_1^0, \dots, w_d^0, R_1, \dots, R_d, i_0)$$

where

1. *V is a finite alphabet whose elements are called* objects*;*
2. *$C \subseteq V$ is a set of* catalysts*;*
3. *μ is a* membrane structure *consisting of d membranes (usually labelled with i and represented by corresponding brackets $[_i$ and $]_i$, with $1 \leq i \leq d$);*

4. w_i^0, $1 \leq i \leq d$, *are strings over* V *associated with the regions* $1, 2, \ldots, d$ *of* μ*; they represent multisets of objects present in the regions of* μ *(the multiplicity of a symbol in a region is given by the number of occurrences of this symbol in the string corresponding to that region);*

5. R_i, $1 \leq i \leq d$, *are finite sets of* evolution rules *over* V *associated with the regions* $1, 2, \ldots, d$ *of* μ*; these evolution rules are of the forms* $a \rightarrow v$ *or* $ca \rightarrow cv$*, where* c *is a catalyst,* a *is an object from* $V \setminus C$*, and* v *is a string from* $((V \setminus C) \times \{here, out, in\})^*$*;*

6. i_0 *is a number between* 1 *and* d *and it specifies the* output *membrane of* Π.

Chapter 3

Expressiveness of priority

Priority is a frequently used feature of many computational systems. High-priority processes dispose of more central processing unit time in workstations, or preempt the execution of low priority processes through hardware/software-driven interrupt mechanisms. In order to model such systems, many basic process algebras have been enriched with some priority mechanisms (see, e.g., [2, 15, 34, 33, 81]). Priority is also implicitly used in many stochastic process calculi, where immediate actions take precedence over timed actions (see, e.g., [5, 50, 10], or where actions are equipped with an explicit priority level (e.g. [6]).

This chapter is devoted to the investigation of the expressiveness of priority in (untimed) concurrent systems, in order to delineate the expressive power gained by the addition of priority and to compare the relative expressive power of different priority mechanisms, by studying a couple of problems in distributed systems.

According to the classification in [33], the basic priority mechanisms reported in the literature can be divided into two main groups: those based on *global* priority (see, e.g., [1, 34]) and those based on *local* priority (see, e.g., [15, 81]). The difference is motivated by the *scope* of the priority effects on the system: in the case of global priority, a high-priority action is able to preempt any other low-priority action in the system, so that only higher priority processes are allowed to evolve. In the case of local priority, this effect is limited to the *location* of the process, where a location can be thought of as a site in a distributed system and may be represented by the

scope of some name or the guarded choice between actions with different priorities. An example may be helpful in showing the difference between the two. Consider the system S composed of five processes

$$S \triangleq \overline{a}.P \mid a.Q_1 + \underline{b}.Q_2 \mid \overline{\underline{b}}.R \mid c.T_1 + d.T_2 \mid \overline{c}.V$$

where the sum operator represents the usual choice between different actions, output actions are overlined and high-priority actions are underlined. According to the semantics of CCS^{sg} (CCS with a global notion of priority) and CCS^{sl} (CCS with local priority) reported in [33], the processes $a.Q_1 + \underline{b}.Q_2$ and $\overline{\underline{b}}.R$ are ready to perform a high-priority action on channel $\underline{b}$. In CCS^{sg} semantics this action is forced to happen before any other low priority transition in S, while in CCS^{sl} semantics, the action $\underline{b}$ only preempts the execution of the action a, so that the synchronisation on c of the last two processes may even happen first. In other words, with a global priority notion the only possible internal transition of the system S is

$$S \rightarrow \overline{a}.P \mid Q_2 \mid R \mid c.T_1 + d.T_2 \mid \overline{c}.V$$

while in presence of local priority also the evolution

$$S \rightarrow \overline{a}.P \mid a.Q_1 + \underline{b}.Q_2 \mid \overline{\underline{b}}.R \mid T_1 \mid V$$

can happen. In both cases only the reduction on channel a is preempted.

As a basic representative for a calculus with global priority, we consider a very minimal fragment, which we call FAP, of CCS [66] (without restriction and recursion, with asynchronous communication), enriched with static priority and global preemption (like in CCS^{sg} reported in [33]) where only the prefix operator on inputs is present and the asynchronous output is characterised by the possibility of assigning different priority to the outgoing messages.

As a representative for a calculus with local priority, we consider Phillips' CCS with priority guards (CPG for short) [81].

Moreover, we consider two well-known unprioritised calculi, namely the π-calculus introduced in the previous chapter and its broadcast-based version bπ-calculus [42], that will be compared with the two prioritised calculi above.

The two problems in distributed systems we will use to distinguish the expressive power of these four calculi are the *leader-election* problem [58] already used to study expressiveness gap between, e.g., synchronous and asynchronous π-calculus [76], and an apparently new problem we have called *the last man standing* problem (LMS for short), consisting of the capability of processes to recognise the absence of other processes ready to perform synchronisations or input/output operations. In other words, the LMS problem is solvable if a process is able to check that it is the only one active in a network.

We first analyse the expressiveness of global priority, in order to check if it can be proved to be more expressive than local priority as it would be natural to expect. As previously mentioned, in order to represent a global priority model we choose FAP, a fragment of CCS extended with stratified, global priority (a slight variant of the CCS with static priority and global preemption, CCS^{sg} studied in [33]) where the only operators are the parallel composition of processes and the prefix on inputs, while the asynchronous output models the dispatch of messages with two different levels of priority: the delivery of high-priority messages is ensured to happen before that of low-priority ones.

We prove that this very simple language (deprived of synchronous communication, choice, recursion or replication and hence finite) is able to write programs capable of solving the leader election problem in any connected graph of processes without knowledge of the number of processes involved in the election.

By applying the idea used in [42, 81] we have as a corollary that FAP cannot be distributively encoded in the π-calculus, but we prove this to remain true also for partially correct encodings which introduce divergence or failure in their computations as consequences of livelocked or deadlocked conditions. This result can also be extended to the translations of bπ-calculus and CPG into the π-calculus, thus relaxing the encoding conditions already stated in [42, 81].

Another consequence of the above leader election result in FAP is the impossibility of its encoding in CPG under uniformity and independence-preserving conditions, which constitutes the expected result on the expressiveness gap between global and

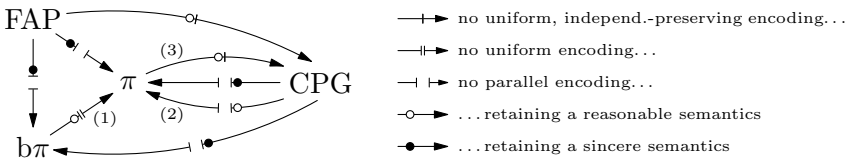


Figure 3.1: Impossibility results: (1) C.Ene, T.Muntean; (2, 3) I. Phillips; the remaining ones are presented in this chapter.

local priority in the chosen process algebra framework. It is worth considering that the separation between these two prioritised languages and the π-calculus is stronger than that between FAP and CPG themselves, which is a first hint of the expressive power of priority in both global and local approaches.

In order to strengthen the separation between prioritised and non-prioritised languages, we then introduce a new setting denoted as the *last man standing problem*. In this setting a bunch of n processes must realise if there is only one process participating to the LMS (and in that case, the only process would be the "last man standing"), i.e. understand if $n = 1$ or $n > 1$ in a distributed way. We prove that it is possible to solve the LMS both in FAP and CPG (but we claim that it is also possible within other priority approaches like [15, 33]), while it is not possible in non-prioritised languages like the π-calculus but also the bπ-calculus. This result implies that there exist no distributed encodings of FAP and CPG into the bπ-calculus, hence showing that the greatest expressiveness of priority does not derive from the broadcast-like power of preemption, but from the capability of processes to know if another process is ready to perform a synchronisation on some channel *or not*. In non-prioritised calculi it is possible to know if some process *is* ready to perform some synchronisation, but it is not decidable if, on the contrary, the condition does not hold. We show that in a distributed setting this simple capability is peculiar to priority (global or local, stratified or not) and cannot be obtained otherwise, even providing broadcast-like primitives and admitting divergent or deadlocked computations.

The above results are summarised in Fig. 3.1.

The chapter is structured as follows. In Section 3.1 the process algebras involved in the separation results are introduced (except for the π-calculus already defined) and a brief explanation of their main features is given. Section 3.2 contains a short discussion and the formal definitions of the properties of an encoding. In Sect. 3.2.1 and 3.2.2 the leader election and LMS problems are formalised. In Sect. 3.3.1 and 3.3.2 the respective separation results are shown, then a short discussion is reported.

3.1 Calculi

We introduce now the calculi of interest in this chapter by giving their syntax, semantics and a short explanation of their functioning.

3.1.1 CCS

One of the first calculi proposed for modelling concurrent systems is CCS [66]. The main features of CCS are the point-to-point communication capability of processes and the static nature of links.

Its fundamental entity is the *name*. Names, present in a potentially unlimited number, are identified with $x, y, \ldots$ and are members of the set N of names. They have no structure and represent communication channels. In addition to names, *processes* constitute the other basic entity of CCS. Processes are indicated as $P, Q, \ldots \in \mathcal{P}$ and are built by names according to the following syntax:

$$P, Q \quad ::= \quad \mathbf{0} \;\Big|\; a(x).P \;\Big|\; \overline{a}\langle d\rangle.P \;\Big|\; P + Q \;\Big| \tag{3.1}$$

$$P \mid Q \;\Big|\; (\nu\, a)P \;\Big|\; Z \stackrel{def}{=} P \tag{3.2}$$

where

- $\mathbf{0}$ represents the null process, capable of doing nothing;
- $a(x).P$ represents a process listening on channel a and ready to receive some datum whose local name, in P, is x; after receiving this datum, the process behaves as specified in the rest of the program P;

Prefix	$\dfrac{-}{\mu.P \xrightarrow{\mu} P}$
Sum	$\dfrac{P \xrightarrow{\mu} P'}{P+Q \xrightarrow{\mu} P'} \qquad \dfrac{Q \xrightarrow{\mu} Q'}{P+Q \xrightarrow{\mu} Q'}$
Parallel	$\dfrac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q} \qquad \dfrac{Q \xrightarrow{\mu} Q'}{P \mid Q \xrightarrow{\mu} P \mid Q'}$
Synchronisation	$\dfrac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$
Restriction	$\dfrac{P \xrightarrow{\mu} P'}{(\nu\ a)P \xrightarrow{\mu} (\nu\ a)P'} \quad \mu, \overline{\mu} \neq a$
Constant	$\dfrac{P \xrightarrow{\mu} P'}{Z \xrightarrow{\mu} P'} \quad Z \stackrel{def}{=} P$

Table 3.1: CCS semantic rules.

- $\overline{a}\langle d \rangle.P$ represents a process ready to perform an output on channel a, that is to send datum d to another process listening on a; after this output operation, the process behaves as specified in the rest of the program P;

- $P + Q$ represents the possibility of choice between two different behaviours, P and Q; if both P and Q are available, then the choice is nondeterministic;

- $P \mid Q$ represents the parallel composition of two processes, that is P and Q are concurrent processes running in parallel;

- $(\nu\ a)P$ represents the restriction of the name a: if some name a appears outside the process $(\nu\ a)P$, then it represents a different channel with respect to the channel a inside P;

- $Z \stackrel{def}{=} P$ allows us to define recursive behaviour of processes by associating the behaviour of P with the process name Z.

The following example shows two parallel processes able to communicate on

channel a.

$$a(x).P \mid \overline{a}\langle d\rangle.Q \quad \xrightarrow{\tau} \quad P[d/x] \mid Q$$

The labelled arrow represents the occurrence of the communication, that is the transition from one state – where $a(x).P$ is listening for a datum and $\overline{a}\langle d\rangle.Q$ is ready to send it – to another – Q sent the datum and P received it. The label τ over the arrow denotes a *silent transition*, that is the occurrence of a communication that is invisible to an external observer: in fact it happens silently between two processes, that are the only entities aware of the communication. The form $P[d/x]$ denotes that after the transition the variable x has been replaced by (relabelled as) d inside P.

The following example points out the nondeterminism arising from parallel composition. Here two pairs of processes (P, Q and P, R) are ready to communicate, so the system may evolve in two different ways:

$$a(x).P \mid \overline{a}\langle d_1\rangle.Q \mid \overline{a}\langle d_2\rangle.R \quad \xrightarrow{\tau} \quad P[d_1/x] \mid Q \mid \overline{a}\langle d_2\rangle.R$$
$$a(x).P \mid \overline{a}\langle d_1\rangle.Q \mid \overline{a}\langle d_2\rangle.R \quad \xrightarrow{\tau} \quad P[d_2/x] \mid \overline{a}\langle d_1\rangle.Q \mid R$$

In the first case, $ax.P$ received the datum d_1 from $ad_1.Q$, while in the second it received d_2 from $\overline{a}\langle d_2\rangle R$.

Nondeterminism may arise also in consequence of multiple available choices, like in the following example.

$$a(x) + b(y).P \mid \overline{a}\langle d_1\rangle.Q \mid \overline{b}\langle d_2\rangle.R \quad \xrightarrow{\tau} \quad P[d_1/x] \mid Q \mid \overline{b}\langle d_2\rangle.R$$
$$a(x) + b(y).P \mid \overline{a}\langle d_1\rangle.Q \mid \overline{b}\langle d_2\rangle.R \quad \xrightarrow{\tau} \quad P[d_2/x] \mid \overline{a}\langle d_1\rangle.Q \mid R$$

Here the process $a(x) + b(y).P$ has two choices: it may receive some datum on channel a (like in the first transition) or some other on channel b (like in the second one).

The set of available transitions from each state is determined by a finite set of semantic rules, reported in Table 3.1.

$$
\begin{array}{rcl}
\left\{\!\left[(i, c_1, \ldots, c_n) \right]\!\right\}_R & = & M_i^R \mid R_1^{c_1} \mid \cdots \mid R_n^{c_n} \\
\\
M_i^R & \triangleq & \left\{\!\left[(i : I_i) \right]\!\right\}_R \quad \text{with } R = (I, n) \text{ and } (i : I_i) \in I \\
\\
\left\{\!\left[\big(i : Inc(r_j)\big) \right]\!\right\}_R & = & \overline{inc}_j.M_{i+1}^R \\
\left\{\!\left[\big(i : DecJump(r_j, s)\big) \right]\!\right\}_R & = & \bar{t}_j.(z_j.M_s^R + n_j.M_{i+1}^R) \\
\\
R_j^0 & \triangleq & Z_j \\
R_j^k & \triangleq & (\nu\ a)\big(N_j\langle a\rangle \mid a.R_j^{k-1}\big) \qquad (k > 0) \\
\\
Z_j & \triangleq & t_j.\bar{z}_j.Z_j \ + \ inc_j.(\nu\ a)\ \big(N_j\langle a\rangle \mid a.Z_j\big) \\
N_j(a) & \triangleq & t_j.\overline{a}.\overline{n}_j \ + \ inc_j.(\nu\ a')\ \big(N_j\langle a'\rangle \mid a'.N_j\langle a\rangle\big)
\end{array}
$$

Table 3.2: Definition of the function $\left\{\!\left[\cdot \right]\!\right\}$ for the encoding of RAMs into CCS, CPG and bπ-calculus. Although different, these calculi share a common core which allows the exploitation of the same encoding function for such kind of RAMs.

Proposition 3.1 *CCS is Turing-complete.*

Proof: The encoding of a RAM into CCS is similar to the one given for the π-calculus, and is reported in Table 3.2. □

The main difference between the encodings of RAMs into the π-calculus and CCS is related to the presence of replication in the π-calculus, which is more tractable than recursion but requires some additional synchronisation to achieve recursive behaviour.

3.1.2 The CPG Language

The CPG language [81] derives from CCS, extended with a local notion of priority over actions. In this respect, CPG fully embeds (syntactically and semantically) CCS so that any CCS program is still valid and is denoted by the same semantics, but in addition each action can be guarded by a set of names representing the actions whose co-action availability may prevent its execution. For example, in the system S

$$S \triangleq \overline{a} : b.Q \mid \overline{b}.R$$

the first action b is guarded by $\overline{a}$, so that its execution is prevented by the presence of any parallel complementary action a. Since there is no such co-action in S, it can undergo the reduction

$$S \quad \rightarrow \quad Q \mid R$$

The system S'

$$S' \triangleq a.P \mid S$$

cannot reduce in the same way, because of the presence of the action a in $a.P$. The locality of this effect can be noticed in the following system

$$S'' \triangleq a.P \mid \overline{a} : b.Q \mid \overline{b}.R \mid c : b.T$$

where the following reduction over b is possible:

$$S'' \quad \rightarrow \quad a.P \mid \overline{a} : b.Q \mid R \mid T$$

In fact, the process $\overline{a} : b.Q$ is still stuck for the presence of the action a, while $\overline{b}.R$ is free to synchronise with $c : b.T$, since there is no presence of co-action $\overline{c}$ which would prevent the reduction with the last process.

Definition 3.1 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, y, \ldots \in \mathcal{N}$. CPG syntax is defined in terms of the following grammar*

$$P \quad ::= \quad \mathbf{0} \quad \Big| \quad A\langle \mathbf{x} \rangle \quad \Big| \quad \sum_{i \in I} S_i : \alpha_i.P_i \quad \Big| \quad P_1 \mid P_2 \quad \Big| \quad (\nu\, x)P$$

where

$$\alpha_i \quad ::= \quad x \;\Big|\; \overline{x} \;\Big|\; \tau$$

and each constant A is assumed to have a unique defining equation $A(\mathbf{x}) \triangleq P$. $S_i \subseteq \mathcal{N}$ represents the set of actions whose co-actions availability prevents the execution of α_i.

We report the reduction semantics given in [81], where $\mathcal{N}$ is a set of names, $\overline{\mathcal{N}}$ co-names, $\mathcal{U}$ set of names which can be used as priority guards, $\overline{\mathcal{U}}$ the respective co-names, $\mathrm{Std} = N \cup \overline{N}$, $\mathrm{Pri} = U \cup \overline{U}$, $\mathrm{Vis} = \mathrm{Std} \cup \mathrm{Pri}$, $\mathrm{Act} = \mathrm{Vis} \cup \{\tau\}$, with $u, v, \ldots$ ranging over Pri, $a, b, \ldots$ over Vis, $\alpha, \beta, \ldots$ over Act, $S, T, \ldots$ over finite subsets of Vis, $U, V, \ldots$ over finite subsets of Pri.

Definition 3.2 *The function $\mathrm{fn}(P) \subseteq \mathcal{N} \cup \mathcal{U}$ is defined by induction on $P \in \mathcal{P}$ as follows:*

$$\begin{aligned}
\mathrm{fn}(\sum_{i \in I} S_i : \alpha_i.P_i) = \{n \in \mathcal{N} \cup \mathcal{U} \mid \quad & \exists i \in I : \; n \in S_i \cup \{\alpha_i\} \vee \\
& \overline{n} \in S_i \cup \{\alpha_i\} \vee \\
& n \in \mathrm{fn}(P_i) \quad \} \\
\mathrm{fn}(P_1 \mid P_2) = \; & \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2) \\
\mathrm{fn}((\nu\, a)P) = \; & \mathrm{fn}(P) \setminus \{a\} \\
\mathrm{fn}(A\langle a_1, \ldots, a_n \rangle) = \; & \{a_1, \ldots, a_n\}
\end{aligned}$$

Definition 3.3 *The congruence relation $\equiv$ on CPG processes 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:*

$$\begin{aligned}
(\nu\, x)P \mid Q \;&\equiv\; (\nu\, x)(P \mid Q) && \text{if } x \notin \mathrm{fn}(Q) \\
(\nu\, x)P \;&\equiv\; P && \text{if } x \notin \mathrm{fn}(P) \\
A\langle \tilde{b} \rangle \;&\equiv\; P\{\tilde{b}/\tilde{a}\} \text{ if } A(\tilde{a}) \stackrel{def}{=} P
\end{aligned}$$

Definition 3.4 *The set* $\mathrm{off}(P) \subseteq \mathrm{Pri}$ *of "higher priority" actions "offered" by* P *is defined by induction on CPG processes by the following rules:*

$$\mathrm{off}(\sum_{i\in I} S_i : \alpha_i.P_i) = \{\alpha_i : i \in I, \alpha_i \in \mathrm{Pri}, \alpha_i \notin S_i\}$$
$$\mathrm{off}(P_1 \mid P_2) = \mathrm{off}(P_1) \cup \mathrm{off}(P_2)$$
$$\mathrm{off}(\nu\ aP) = \mathrm{off}(P) \setminus \{a, \overline{a}\}$$
$$\mathrm{off}(A\langle\tilde{b}\rangle) = \mathrm{off}(P\{\tilde{b}/\tilde{a}\})\ \textit{if}\ A(\tilde{a}) \stackrel{def}{=} P$$

Definition 3.5 *Let* P *be a CPG process and let* $S \subseteq \mathrm{Act}$ *be finite.* P *eschews* S *(written* $P\,\mathrm{eschews}\,S$*) iff* $\mathrm{off}(P) \cap \overline{S} = \emptyset$.

Definition 3.6 *CPG semantics is given in terms of the reduction system described by the following rules:*

$$\frac{}{S : \tau.P + M \rightarrow_{S\cap \mathrm{Pri}} P} \qquad \frac{S : a.P + M\,\mathrm{eschews}\,T \quad T : \overline{a}.Q + N\,\mathrm{eschews}\,S}{(S : a.P + M) \mid (T : \overline{a}.Q + N) \rightarrow_{(S\cup T)\cap \mathrm{Pri}} P \mid Q}$$

$$\frac{P \rightarrow_U P' \quad Q\,\mathrm{eschews}\,U}{P \mid Q \rightarrow_U P' \mid Q} \qquad \frac{P \rightarrow_U P'}{(\nu\ a)P \rightarrow_{U\setminus\{a,\overline{a}\}} (\nu\ a)P'} \qquad \frac{Q \equiv P \quad P \rightarrow_U P' \quad P' \equiv Q'}{Q \rightarrow_U Q'}$$

We say that $P \rightarrow Q \iff \exists X : P \rightarrow_X Q$.

The reduction relation $\rightarrow_X$ is parameterised by a set X of names representing the high priority actions whose co-action availability would prevent the occurrence of the described reduction.

We report the definition of barb for CPG in [82].

Definition 3.7 *Let* P *be a CPG process.* P *exhibits barb* α*, written* $P \downarrow \alpha$*, iff*

- $P \equiv (\nu\ \mathbf{y})(S : x.Q + R \mid T)$*, with* $\alpha = x$*,* $x \notin \mathbf{y}$ *or*
- $P \equiv (\nu\ \mathbf{y})(S : \overline{x}.Q + R \mid T)$*, with* $\alpha = \overline{x}$*,* $x \notin \mathbf{y}$.

CPG Turing completeness is directly inherited from CCS:

Proposition 3.2 *CPG is Turing-complete.*

3.1.3 The bπ-calculus

The bπ-calculus [42] is a variant of the π-calculus where the point-to-point synchronisation mechanism is replaced by broadcast communication. For example, while the π-calculus program

$$S \triangleq \overline{a}\langle b\rangle.P \mid a(x).Q \mid a(y).R \mid a(z).T$$

can evolve in one step to a system like S_1

$$S \rightarrow S_1 \triangleq P \mid Q\{b/x\} \mid a(y).R \mid a(z).T$$

where only one of Q, R, S is affected by the performed communication, in bπ-calculus the system S directly evolves to S_2

$$S \rightarrow S_2 \triangleq P \mid Q\{b/x\} \mid R\{b/y\} \mid T\{b/z\}$$

where all the processes listening on channel a receive the broadcasted message.

In order to uniform the presentation of the languages analysed here, in the following we introduce a variant of the bπ-calculus defined in terms of reduction semantics, instead of the labelled transition system used in [42].

Definition 3.8 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, y, \ldots \in \mathcal{N}$. The syntax of the b$\pi$-calculus is defined in terms of the following grammar*

$$P \quad ::= \quad \mathbf{0} \quad \Big| \quad A\langle \mathbf{x}\rangle \quad \Big| \quad \sum_{i\in I} \pi_i.P_i \quad \Big| \quad P_1 \mid P_2 \quad \Big| \quad (\nu\ x)P$$

where

$$\pi_i \quad ::= \quad \tau \quad \Big| \quad (x)y \quad \Big| \quad \langle x\rangle y$$

and each constant A is assumed to have a unique defining equation $A(\mathbf{x}) \triangleq P$.

Definition 3.9 *The congruence relation $\equiv$ on bπ-calculus processes 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:

$$\begin{aligned}
(\nu\, x)P \mid Q &\equiv (\nu\, x)(P \mid Q) && \text{if } x \notin \mathrm{fn}(Q)\\
(\nu\, x)P &\equiv P && \text{if } x \notin \mathrm{fn}(P)\\
A\langle\tilde{b}\rangle &\equiv P\{\tilde{b}/\tilde{a}\} \text{ if } A(\tilde{a}) \stackrel{def}{=} P
\end{aligned}$$

where the function fn *is defined as*

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

The definition of barb for the bπ-calculus follows, the same as for the π-calculus.

Definition 3.10 *Let P be a bπ-calculus process. P exhibits barb π, written $P \downarrow \pi$, iff*

- $P \equiv (\nu\, \mathbf{y})(x(z).Q + R \mid S)$, *with* $\pi = x$, $x \notin \mathbf{y}$ *or*
- $P \equiv (\nu\, \mathbf{y})(\overline{x}\langle z\rangle.Q + R \mid S)$, *with* $\pi = \overline{x}$, $x \notin \mathbf{y}$.

Definition 3.11 *bπ-calculus semantics is given in terms of the reduction system described by the following rules:*

$$\frac{R \not\downarrow \mu}{\prod_i(\mu(y_i).P_i + M_i) \mid (\overline{\mu}\langle z\rangle.Q + N) \mid R \;\rightarrow\; \prod_i P_i\{z/y_i\} \mid Q \mid R}$$

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

where $\prod$ represents the parallel composition of zero or more processes,

$$\prod_{i=1}^{n} P_i \quad = \quad P_1 \mid \cdots \mid P_n$$

for some $n \geq 0$.

Proposition 3.3 *The bπ-calculus is Turing-complete.*

Proof: The encoding of a RAM into the bπ-calculus is the same as given for CCS, in table 3.2. □

The encoding does not exploit the broadcast capabilities of the bπ-calculus, instead it carefully forces every output to be received by at most one process. Furthermore, since the one-to-many communication of the bπ-calculus allows the output message to be lost if some other process is not listening on the right channel at the right time (in contrast with the binary synchronisation of the π-calculus, which requires a previous handshake between the involved processes), such encoding assures the correctness of the computation by forcing each input to be exhibited *before* some output may be performed on the corresponding channel.

3.1.4 The FAP language

As previously outlined, the FAP language is a slight variant of a minimal CCS^{sg} fragment, that is CCS extended with static, global priority [33]: by keeping FAP minimal the expressive power of global priority can be better isolated. Only two operators are present in FAP: parallel composition and prefix. The prefix operation is allowed only after input actions, so that the output can be considered asynchronous as for the asynchronous π-calculus (see e.g. [76]). Output actions are characterised by two priority levels, meaning that high priority output synchronisations are guaranteed to happen before low priority ones. As an example, consider the system

$$S \triangleq a.P \mid a.Q \mid b.R \mid \overline{a} \mid \underline{\overline{a}} \mid \overline{b}$$

The processes $\overline{a}, \underline{\overline{a}}, \overline{b}$ model messages which must be delivered to the processes listening on the appropriate channels: The message $\underline{\overline{a}}$ has higher priority with respect to any other message in S and hence must be delivered fist. Consequently the only possible transitions of S are

$$S \rightarrow P \mid a.Q \mid b.R \mid \overline{a} \mid \overline{b} \qquad S \rightarrow a.P \mid Q \mid b.R \mid \overline{a} \mid \overline{b}$$

where the process receiving the message is nondeterministically chosen. After this transition, the low-priority messages $\overline{a}$ and $\overline{b}$ can finally be delivered. In order to simplify the notation, inputs lack any denotation of priority, but the results presented here are completely independent of this design choice.

Definition 3.12 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, \ldots \in \mathcal{N}$. FAP syntax is defined in terms of the following grammar*

$$P \quad ::= \quad \mathbf{0} \;\Big|\; x.P \;\Big|\; \overline{x} \;\Big|\; \underline{\overline{x}} \;\Big|\; P \mid Q$$

In order to keep it simple as well, we define FAP semantics in terms of a reduction system in the style of [65].

Definition 3.13 *Structural congruence for FAP is the congruence $\equiv$ generated by the following equations:*

$$P \mid \mathbf{0} \equiv P, \qquad P \mid Q \equiv Q \mid P, \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

Definition 3.14 *FAP operational semantics is given in terms of the reduction system described by the following rules:*

$$\frac{}{x.P \mid \overline{x} \mapsto P} \qquad \frac{}{x.P \mid \underline{\overline{x}} \twoheadrightarrow P}$$

$$\frac{P \twoheadrightarrow P'}{P \mid Q \twoheadrightarrow P' \mid Q} \qquad \frac{P \mapsto P' \qquad P \mid Q \not\twoheadrightarrow R}{P \mid Q \mapsto P' \mid Q}$$

$$\frac{P \equiv Q \quad P \mapsto P' \quad P' \equiv Q'}{Q \mapsto Q'} \qquad \frac{P \equiv Q \quad P \twoheadrightarrow P' \quad P' \equiv Q'}{Q \twoheadrightarrow Q'}$$

We say that $P \to Q \iff P \mapsto Q \vee P \twoheadrightarrow Q$.

Definition 3.15 *For any process in FAP, the function* fn *is defined as*

$$\mathrm{fn}(\mathbf{0}) = \emptyset \qquad \mathrm{fn}(\overline{x}) = \{x\} \qquad \mathrm{fn}(x.P) = \{x\} \cup \mathrm{fn}(P)$$
$$\mathrm{fn}(\underline{\overline{x}}) = \{x\} \qquad \mathrm{fn}(P \mid Q) = \mathrm{fn}(P) \cup \mathrm{fn}(Q)$$

As for previous languages, we define the notion of barb.

Definition 3.16 *A FAP process P exhibits barb α, written as $P \downarrow \alpha$, iff*

- $P \equiv x.Q \mid R$, $\alpha = x$, *or*
- $P \equiv \overline{x} \mid R$, $\alpha = \overline{x}$, *or*
- $P \equiv \underline{\overline{x}} \mid R$, $\alpha = \underline{\overline{x}}$.

3.2 Encodings

In order to provide the results previously outlined, we need now to formalise the encoding conditions relevant for the expressiveness separation between languages.

As extensively discussed in [100], there is no standard set of requirements that an encoding should obey. The kind of results we are going to provide suggests that such requirements should be very weak, in order to increase as much as possible the strength of the separation between the calculi considered here. For this reason we closely follow the approach adopted in [82], where the semantics of programs is evaluated exclusively in terms of *observations over maximal computations*. They consist in detecting a subset of the actions that a process can execute during its overall computations, disregarding the order in which they are executed (or, more precisely, the order in which they are *exhibited*). The properties based on this notion of observation are particularly weak if compared to the set of requirements of Sect. 5.1.2, where any suitable encoding must necessarily preserve the order of execution of the encoded processes.

We first formalise the notion of *observables* of a program computation, in the style of [82].

Definition 3.17 *Let L be a process language and processes $P, P_0, \ldots \in L$. A* computation $\mathcal{C}$ of P *is a finite or infinite sequence $P = P_0 \rightarrow P_1 \rightarrow \cdots$. $\mathcal{C}$ is* maximal *if it cannot be extended.*

A computation of a process P is the sequence of states P can reach during its execution. Each process P may present many different computations due to the nondeterminism intrinsic in concurrent calculi.

Definition 3.18 *Let L be a process language with names in $\mathcal{N}$ and processes $P_0, \ldots, .P_i \in L$. Let $\mathcal{C}$ be a computation $P_0 \to \cdots \to P_i \cdots$. Given a set of* intended observables $\mathrm{Obs} \subseteq \mathcal{N}$*, the observables of $\mathcal{C}$ are* $\mathrm{Obs}(\mathcal{C}) = \{x \in \mathrm{Obs} : \exists i\ P_i \downarrow x\}$.

The observables of a computation C are the set of all the external interactions the process may perform in the states reached during the computation.

Some of the separation results are based on the topology of the network of processes: for example, the encoding impossibility of the π-calculus into value-passing CCS [76] is based on the hypothesis that the encoding does not increase the connection of the network, that is all the processes which are independent (not sharing free names) in the source language must remain independent after the encoding. The same criterion will be necessary to separate FAP and CPG.

Definition 3.19 *Let L be a process language. Two processes $P, Q \in L$ are* independent *if they do not share any free names, that is* $\mathrm{fn}(P) \cap \mathrm{fn}(Q) = \emptyset$.

We now define the conditions an encoding may preserve, in the style of [82].

Definition 3.20 *Let L, L' be process languages. An encoding $[\![\cdot]\!]_{\cdot} L \to L'$ is*

1. observation-respecting *if* $\forall P \in L$,

   - *for every maximal computation $\mathcal{C}$ of P there exists a maximal computation $\mathcal{C}'$ of $[\![P]\!]$ such that* $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(\mathcal{C}')$;
   - *for every maximal computation $\mathcal{C}$ of $[\![P]\!]$ there exists a maximal computation $\mathcal{C}'$ of P such that* $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(\mathcal{C}')$;

2. weakly-observation-respecting *if* $\forall P \in L$,

   - *for every maximal computation $\mathcal{C}$ of P there exists a maximal computation $\mathcal{C}'$ of $[\![P]\!]$ such that* $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(\mathcal{C}')$;

- *for every maximal computation* $\mathcal{C}$ *of* $[\![P]\!]$ *there exists a maximal computation* $\mathcal{C}'$ *of* P *such that* $\mathrm{Obs}(\mathcal{C}) \subseteq \mathrm{Obs}(\mathcal{C}')$*;*

3. distribution-preserving *if* $\forall P_1, P_2 \in L, \quad [\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!]$*;*

4. renaming-preserving *if for any permutation* σ *of the source names in* L *there exists a permutation* θ *in* L' *such that* $[\![\sigma(P)]\!] = \theta([\![P]\!])$ *and the permutations are compatible on observables, that is* $\sigma|_{\mathrm{Obs}} = \theta|_{\mathrm{Obs}}$*;*

5. independence-preserving *if* $\forall P, Q \in L$*, if* P *and* Q *are independent then* $[\![P]\!]$ *and* $[\![Q]\!]$ *are also independent.*

The observation-respecting property is the minimal requirement that any reasonable encoding should preserve: it ensures that some intended observable behaviour is preserved by the translation. The weak variant of this condition admits encodings which introduce divergence or failure deriving from livelocks or deadlocks. Under certain conditions, like fairness or other hypotheses on scheduling or execution, the introduction of divergence or failure by the encoding may be tolerated [72, 77] because it would be guaranteed not (or be very unlikely) to happen anyway.

The distribution-preserving is a very important feature of an encoding in a concurrent framework: it implies its compositionality and above all it guarantees that the degree of parallelism of the translated system does not decrease.

The renaming-preserving property states that the encoding should not introduce asymmetries in the system, essential condition to preserve when impossibility results on problems such as the leader election are completely based on the symmetric topology of the network.

Independence-preserving represents the property of not increasing the connection of the network.

According to [76, 82], a distribution- and renaming-preserving encoding is called *uniform*, and *reasonable* if it also preserves the intended observables over maximal computations. We call *sincere* a weakly-observation-respecting encoding, which is complete but only weakly correct, in the meaning that it admits divergence or premature termination.

3.2.1 The leader election problem

An electoral system represents the situation of a bunch of processes (modelling for example a set of workstations in a network) aiming at reaching a common agreement, i.e. deciding which one of them (and no other) is the leader. The modelling of the election problem in process algebras requires that the system composed of the network of processes will sooner or later signal unequivocally the result of the election on some channels ω_i, which become consequently the observables of interest on the computations of the system.

Definition 3.21 *Let L be a process language, and processes $P_1, \ldots, P_k \in L$. A network Net of size k, with $k \geq 1$, is a system $\mathsf{Net} = P_1 \mid \cdots \mid P_k$.*

Definition 3.22 *A network Net of size k is an* electoral system *if for every maximal computation $\mathcal{C}$ of Net $\exists i \leq k : \mathrm{Obs}(\mathcal{C}) = \{\omega_i\}$, where $\mathrm{Obs} = \{\omega_i : i \in \mathbb{N}\}$.*

In order to keep notation simple, the definition of electoral system reflects the design choices kept in [76, 42, 81] which are based on the hypothesis that the system will never perform external interactions on channels which are not intended to be observable. As in [81, 82], the winner process (the leader) is supposed to signal the outcome of the election, while all the other processes simply do nothing.

Definition 3.23 *Given a network of size k, with $\mathsf{Net} = P_1 \mid \cdots \mid P_k$, two nodes P_i and P_j (with $i \neq j$) are* neighbours *(or connected) if they are not independent.*

Definition 3.24 *A network Net of size k, with $\mathsf{Net} = P_1 \mid \cdots \mid P_k$, is* connected *if one of the following conditions holds:*

- *$k = 1$*
- *$\exists i, j \leq k$, with $i \neq j$, such that P_i and P_j are neighbours and Net', with*

$$\mathsf{Net}' = P_1 \mid \cdots \mid P_{i-1} \mid P_{i+1} \mid \cdots \mid P_k$$

  is also connected.

In a connected network $\mathsf{N}et$, each node is connected at least to another node, and there exists no partition of $\mathsf{N}et$ into two subnetworks such that there are no cross connections between processes from one subnetwork to the other. This condition must hold when some information needs to be propagated between all the nodes.

Definition 3.25 *A network* $\mathsf{N}et$ *of size* k*, with* $\mathsf{N}et = P_1 \mid \cdots \mid P_k$*, is* fully connected *if* P_i *and* P_j *are not independent,* $\forall i, j \leq k$*.*

In this case each node may interact directly with any other node in the network.

3.2.2 The last man standing problem

The last man standing represents a very simple situation where a bunch of n processes in a fully connected network must realise if $n = 1$ or $n > 1$ in a distributed way. The possibility or impossibility to solve the LMS problem is based on the idea that in a given language L a process P *may* or *may not* know if another process Q is ready to perform some intended action on a given channel. Usually the only way P has to know the presence of Q is to *try* a synchronisation on it. Since the input (and often also the output) is blocking, P results blocked if the condition does not hold, or it follows another computation without knowledge of the presence of Q. The definition of LMS system follows.

Definition 3.26 *A network* $\mathsf{N}et_k$ *of size* k *is a* LMS system *if for every maximal computation* $\mathcal{C}$ *of* $\mathsf{N}et_k$

- $\mathrm{Obs}(\mathcal{C}) = \{y\}$ *if* $k = 1$,
- $\mathrm{Obs}(\mathcal{C}) = \{n\}$ *if* $k > 1$,

where $\mathrm{Obs} = \{y, n\}$.

3.3 Separation results

The separation results previously outlined follow. We first give those based on the leader election, and then those based on the LMS problem.

3.3.1 Leader-election-based separation results

The bπ-calculus and CPG were proved capable of solving the leader election in a fully connected network without knowledge of the number of processes [42, 81]. Here we show that in FAP this is possible in any (not only a fully) connected network.

Theorem 3.1 *Let $P_1, \ldots, P_k$ be FAP processes,* $\mathsf{Net} = P_1 \mid \cdots \mid P_k$. *Let*

$$\begin{aligned} P_n = \overline{m}_n \mid \underline{\overline{s}}_n \mid m_n.s_n.(\omega_n \mid \underline{\overline{d}}_{n1} \mid \cdots \mid \underline{\overline{d}}_{nz_n}) \\ \mid d_{n1}.(s_n \mid \underline{\overline{d}}_{n1} \mid \cdots \mid \underline{\overline{d}}_{nz_n}) \\ \vdots \\ \mid d_{nz_n}.(s_n \mid \underline{\overline{d}}_{n1} \mid \cdots \mid \underline{\overline{d}}_{nz_n}) \end{aligned}$$

where z_n is the number of neighbours of P_n, with $1 \leq n \leq k$ and P_i, P_h are neighbours iff $\exists j, l : d_{ij} = d_{hl}$, with ω_i, s_j, m_h distinct and $\omega_i \neq s_j \neq m_h \neq d_{pq}, \forall i, j, h, p, q$. If Net *is connected, then* Net *is an electoral system.*

Proof: For $k = 1$, any maximal computation of $\mathsf{Net} = P_1$ is of the form

$$P_1 \equiv \overline{m}_1 \mid \underline{\overline{s}}_1 \mid m_1.s_1.\omega_1 \quad \mapsto \quad \underline{\overline{s}}_1 \mid s_1.\omega_1 \quad \twoheadrightarrow \quad \omega_1$$

hence Net is an electoral system. In fact the first reduction is only possible by a synchronisation on channel m_1, while the second is a prioritised reduction on channel s_1. For $k \geq 1, \mathsf{Net} = P_1 \mid \cdots \mid P_k$, the first (low-priority) reduction can happen only on one of the m_n channels. Suppose – without loss of generality – that it happens on channel m_1. Then the only possible (high-priority) reduction is on channel s_1, and the system evolves as follows

$$\begin{aligned} \mathsf{Net} = & P_1 \mid P_2 \mid \cdots \mid P_k \mapsto \\ & Q_1 \mid P_2 \mid \cdots \mid P_k \twoheadrightarrow \\ & Q_2 \mid P_2 \mid \cdots \mid P_k = \mathsf{Net}_2 \end{aligned}$$

with

$$\begin{aligned} Q_1 = \underline{\overline{s}}_n \mid s_1.(\omega_1 \mid \underline{\overline{d}}_{11} \mid \cdots \mid \underline{\overline{d}}_{1z_1}) \mid d_{11}.(s_1 \mid \underline{\overline{d}}_{11} \mid \cdots \mid \underline{\overline{d}}_{1z_1}) \\ \vdots \\ \mid d_{1z_1}.(s_1 \mid \underline{\overline{d}}_{11} \mid \cdots \mid \underline{\overline{d}}_{1z_1}) \end{aligned}$$

and

$$Q_2 = \omega_1 \mid \overline{\underline{d}}_{11} \mid \cdots \mid \overline{\underline{d}}_{1z_1} \mid d_{11}.(s_1 \mid \overline{\underline{d}}_{11} \mid \cdots \mid \overline{\underline{d}}_{1z_1})$$
$$\vdots$$
$$\mid d_{1z_1}.(s_1 \mid \overline{\underline{d}}_{11} \mid \cdots \mid \overline{\underline{d}}_{1z_1})$$

Like in the previous case, after the first two reductions the system exhibits already the winner. We must verify that for any subsequent computation, no other barb on $\omega_i, i \neq 1$ is exhibited. To prove this, we must ensure that every input on $s_i, i \neq 1$ is exhibited by a sequence of high-priority reductions before some other low-priority reduction on any of the channels $m_i, i \neq 1$ may occur: in this way any high-priority message $\overline{\underline{s}}_i, i \neq 1$ would be extinguished and none of the remaining ω_i channels may be exhibited.

We first remark that for every reduction $P \rightarrow P'$ on channels d_{nj}, the number of

- high-priority outputs on channel d_{nj} in P and P' is the same (if $d_{nj} \neq d_{nh}$ $\forall n, j, h$, otherwise it also increases),
- inputs on channel d_{nj} decreases,
- inputs on channel s_i for some i increases,
- high-priority outputs on channels $d_{n'h}$, with $(n', h) \neq (n, j)$ may (only) grow.

For each neighbour P_i of P_1, there exists some j, h such that $d_{1j} = d_{ih}$. This means that in $\mathsf{N}et_2$, for each neighbour P_i of Q_2, there is an output $\overline{\underline{d}}_{ih}$ ready to react with the corresponding input on d_{ih} and to disclose an input on s_i. After each reduction on some d_{1j} the number of inputs on s_1 increases, but they can grow only up to a maximum z_1. This means that after at most $z_1 + 1$ (high-priority) reductions, an input on s_i for some i, P_i neighbour of Q_2, is exhibited. In turn, after at most another $z_i + 1 + 1$ reductions (we must take into account the reduction on s_i) another input on s_j, P_j neighbour of P_i or Q_2, is exhibited, and so on. Since the network is connected, after at most $n + \sum_{n \in N} z_n$ high-priority reductions, $\mathsf{N}et_2$ leads to a

system where every s_n is exhibited, thanks to the fact that the number of high-priority outputs $\overline{d}_{nj}$ never decreases, so that any input on d_{nj} can be elided. After this chain of high-priority reductions, a sequence of $k-1$ low-priority reductions on channels $m_i, i \neq 1$ occur, then (because of the lack of any output $\overline{s}_i$) the computation ends before any other ω_i is exhibited. □

The next lemma [82] is used to prove that in π-calculus the above results cannot be obtained without knowledge of the number of processes in the electoral system and also that the LMS problem is undecidable. As noted in [82], it would also hold for any language having comparable semantics of the parallel operator, such as Mobile Ambients [23].

Lemma 3.1 *For any π-calculus processes P_1, P_2, if P_i has a maximal computation with observables $O_i (i = 1, 2)$ then $P_1 \mid P_2$ has a maximal computation with observables O such that $O_1 \cup O_2 \subseteq O$.*

Next we state a similar but weaker result for the bπ-calculus, which is also needed for the separation from FAP and CPG based on the LMS problem.

Lemma 3.2 *For any bπ-calculus processes P_1, P_2, if P_i has a maximal computation with observables $O_i (i = 1, 2)$ then $P_1 \mid P_2$ has two maximal computations $\mathcal{C}_1, \mathcal{C}_2$ (not necessarily distinct) with respective observables O'_1, O'_2 such that $O_i \subseteq O'_i$.*

Proof: Let

$$P_i \rightarrow P_{i2} \rightarrow \cdots \rightarrow P_{in} \rightarrow \cdots$$

be the two maximal computations (empty, finite or infinite) of P_1 and P_2, with observables O_1 and O_2 respectively. If $P_1 \not\rightarrow$ and $P_2 \not\rightarrow$ (empty computations) then the sets of observables O_1, O_2 are trivially included in the observables of $P_1 \mid P_2$.

If $P_1 \rightarrow P_{12}$, then

$$P_1 \equiv (\nu\ \mathbf{a})(\pi.Q \mid R) \qquad \text{with} \quad \pi = \tau \quad \text{or} \quad \pi = \overline{x}\langle \mathbf{y} \rangle$$

and

$$P_1 \mid P_2 \quad \rightarrow \quad P_{12} \mid P'_2$$

where P'_2 may be the same as P_2 if (at least) one of the following conditions holds:

- $\pi = \tau$;
- $x \in \mathbf{a}$;
- $P_2 \not\downarrow x$.

In the same way, if $P_{12} \rightarrow P_{13}$ then

$$P_{12} \mid P_2' \quad \rightarrow \quad P_{13} \mid P_2''$$

and so on. By defining

$$\mathcal{C}_1 = \quad P_1 \mid P_2 \quad \rightarrow \quad P_{12} \mid P_2' \quad \rightarrow \quad P_{13} \mid P_2'' \quad \rightarrow \cdots$$

we have that $O_1 \subseteq O_1'$. Symmetrically, by defining

$$\mathcal{C}_2 = \quad P_1 \mid P_2 \quad \rightarrow \quad P_1' \mid P_{22} \quad \rightarrow \quad P_1'' \mid P_{23} \quad \rightarrow \cdots$$

we have $O_2 \subseteq O_2'$. □

The next theorem follows the idea in [42, 81]. As discussed for the definition of sincere semantics, here the condition on the preserved observables is weaker than those considered in [42, 81] but it is possible to relax them as well.

Theorem 3.2 *There is no distribution-preserving and weakly-observation-respecting encoding of FAP in the π-calculus.*

Proof: Consider $P_n = \overline{m}_n \mid \underline{s}_n \mid m_n.s_n.(\omega_n \mid \overline{d}) \mid d.(s_n \mid \overline{d})$ for $n = 1, 2$. By Theorem 3.1, $\mathsf{N}et_1 = P_1$, $\mathsf{N}et_2 = P_2$ and $\mathsf{N}et_{12} = P_1 \mid P_2$ are electoral systems. Let $[\![\cdot]\!]$ be a weakly-observation-respecting and distribution-preserving encoding of FAP into the π-calculus. Hence $[\![P_1]\!]$ has a maximal computation $\mathcal{C}_1$ with observables $\mathrm{Obs}(\mathcal{C}_1) = \omega_1$, because of the weakly-observation-respecting condition. But also $[\![P_2]\!]$ has a maximal computation $\mathcal{C}_2$ with observables $\mathrm{Obs}(\mathcal{C}_2) = \omega_2$.

So we have, for the distribution-preserving property,

$$[\![\mathsf{N}et_{12}]\!] = [\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!]$$

By Lemma 3.1, $[\![\mathsf{N}et_{12}]\!]$ has a maximal computation $\mathcal{C}_{12}$ with observables $\{\omega_1, \omega_2\} \subseteq \mathrm{Obs}(\mathcal{C}_{12})$, not included in the set of observables of any maximal computation of $\mathsf{N}et_{12}$, which contradicts $[\![\cdot]\!]$ being weakly-observation-preserving. □

In order to formalise the separation between FAP and CPG, the definitions pertaining to symmetric configurations of electoral systems are reported from [82]. As previously outlined in Sect. 3.2.1, in order to keep notation simple we omit restrictions and the restriction-preserving conditions present in [82].

Definition 3.27 *Let L be a process language with names in $\mathcal{N}$. A permutation is a bijection $\sigma : \mathcal{N} \to \mathcal{N}$ such that σ preserves the distinction between observable and non-observable names, that is $a \in \mathrm{Obs} \iff \sigma(a) \in \mathrm{Obs}$. Any permutation σ induces a mapping on processes: P is equal to $\sigma(P)$, except that any name a of P is mapped on $\sigma(a)$ in $\sigma(P)$. Given $\mathrm{Obs} = \{\omega_i : i \in \mathbb{N}\}$, a permutation σ induces a bijection $\hat{\sigma} : \mathbb{N} \to \mathbb{N}$ defined as $\hat{\sigma}(i) = j \iff \sigma(\omega_i) = \omega_j$, thus $\sigma(\omega_i) = \omega_{\hat{\sigma}(i)}$.*

Definition 3.28 *Let $\mathsf{N}et = P_1 \mid \ldots \mid P_k$ be a network of size k. An* automorphism *on $\mathsf{N}et$ is a permutation σ such that $\hat{\sigma}|_{\{1,\ldots,k\}}$ is a bijection.*

Definition 3.29 *Let σ be an automorphism on a network of size k. For any $i \in \{1, \ldots, k\}$ the orbit $\mathcal{O}_{\hat{\sigma}}(i)$ generated by $\hat{\sigma}$ is defined as*

$$\mathcal{O}_{\hat{\sigma}}(i) = \{i, \hat{\sigma}(i), \hat{\sigma}^2(i), \ldots, \hat{\sigma}^{(h-1)}(i)\}$$

where $\hat{\sigma}^j$ represents the composition of $\hat{\sigma}$ with itself j times, and h is least such that $\hat{\sigma}^h(i) = i$. If every orbit has the same size, then σ is well balanced*.*

Definition 3.30 *A network $\mathsf{N}et = P_1 \mid \ldots \mid P_k$ is* symmetric with respect to an automorphism σ *iff* $\forall i = 1, \ldots, k\ P_{\hat{\sigma}(i)} = \sigma(P_i)$. *$\mathsf{N}et$ is* symmetric *if it is symmetric with respect to some automorphism with a single orbit (of size k).*

Definition 3.31 *A* ring *is a network $\mathsf{N}et = P_1 \mid \ldots \mid P_k$ which has a single-orbit automorphism σ such that $\forall i, j < k$, if $\mathrm{fn}(P_i) \cap \mathrm{fn}(P_j) \neq \emptyset$ then $i = j$ or $\hat{\sigma}(i) = j$ or $\hat{\sigma}(j) = i$. A ring is* symmetric *if it is symmetric with respect to such an automorphism σ.*

The following lemma is also stated in [82].

Lemma 3.3 *Let L, L' be process languages. If $[\![\cdot]\!] : L \to L'$ is a uniform, observation-respecting and independence-preserving encoding, then for any electoral system* Net *which is a symmetric ring of size k, $[\![$ Net $]\!]$ is also a symmetric ring of size k which is an electoral system.*

Corollary 3.1 *For any $k \geq 1$, there is a symmetric ring of size k in FAP which is an electoral system.*

Proof: It is sufficient to choose a *connected* symmetric ring and apply Theorem 3.1. □

The following theorem, reported from [82], implies the non-encodability of π-calculus into CPG, but also of FAP into CPG.

Theorem 3.3 *For any composite non-prime $k \geq 6$, if* Net *is a symmetric ring of size k in CPG then* Net *is not an electoral system.*

Theorem 3.4 *There is no uniform, observation-respecting and independence-preserving encoding of FAP into CPG.*

Proof: By Lemma 3.3, Corollary 3.1 and Theorem 3.3. □

It is worth remarking that while the separation between π-calculus and CPG derives from the capability of communicating new names typical of the π-calculus, the separation between FAP and CPG is a strict consequence of the different scope of priority in the two languages. This can be straightforwardly explained by the simple configuration of Fig. 3.2, where a ring of size 6 is ideally represented both in FAP and CPG. As stated by Corollary 3.1 and Theorem 3.3, leader election is possible in the first system but not in the second one. In fact, the preemptive effect of global priority is not circumscribed in any way so that the production of a single high-priority output in FAP freezes all the processes trying to perform low-priority reductions. This effect can be exploited for electing the leader in the ring of Fig. 3.2(a), where the processes $P_1, \ldots, P_6$ are in a *low-priority state* (i.e. they are ready

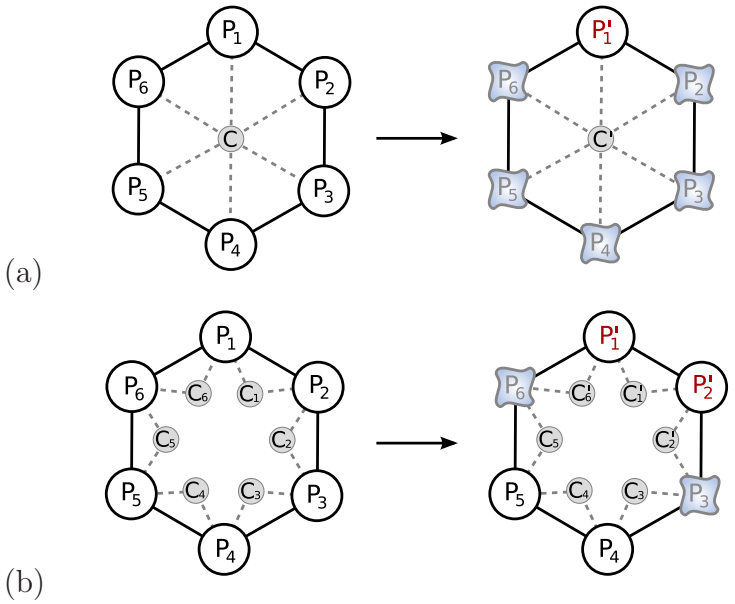


Figure 3.2: Expressiveness of global (a) versus local (b) priority by leader election in a symmetric ring of size 6. The execution of high-priority reductions in languages characterised by global priority such as FAP (a) implies the freezing of *all* the low priority reductions in the system, regardless of the scope of the channel over they occur. Conversely, languages such as CPG circumscribe the preemptive effect to the neighbour processes. This difference allows a partially distributed implementation of this kind of priority, in contrast with the complete centralisation required for the global variant.

to perform just low-priority reductions). The first process (e.g. P_1) which changes its internal state can immediately become the leader by producing a high-priority output which freezes all the other processes and propagates a chain of high-priority reductions in order to denote them as non-leader processes. The same mechanism cannot be exploited in CPG, because the preemptive effect is bound to the scope of restricted names, hence only the neighbour processes can be frozen in this way: thanks to binary synchronisation the process P_1 in Fig. 3.2(b) may denote process P_2 as non-leader and freeze P_3, P_6 (neighbours of P_2 and P_1 respectively) by exploiting local priority. However, the processes P_4, P_5 may still attempt the same local leader

election since they cannot detect in any way the state changes of the rest of the system, so that the symmetry of the ring is restored and may be never eventually broken.

The price for the superior expressiveness of global priority may be revealed in a distributed implementation by the overhead introduced for the additional synchronisations needed in order to freeze all the low-priority processes before the execution of a high-priority reduction. A unique, central coordinator (represented by C in Fig. 3.2(a)) should be introduced to achieve this result, while in CPG the localised effect of priority would allow the partial distribution of such coordination.

3.3.2 LMS-based separation results

In this section the separation results based on the last man standing problem are formalised. First we show that both in FAP and CPG the LMS can be solved, and then from this expressive capability we derive the impossibility of encoding FAP or CPG into π-calculus or bπ-calculus under distribution and weak preservation of observables hypotheses.

Lemma 3.4 *Let P be the following FAP process:*

$$P = \overline{m} \mid \underline{\overline{s}} \mid \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid m.s.(s.q.(\underline{\overline{k}} \mid n) \mid \overline{l} \mid l.q.y)$$

Then $\mathsf{Net}_k = \underbrace{P \mid \cdots \mid P}_{k}$ *is a LMS system of size k, $\forall k \geq 1$.*

Proof: For $k = 1$, $\mathsf{Net}_1 = P$, Net_1 has only one maximal computation $\mathcal{C}$

$$\begin{aligned} P \mapsto P_1 &= \underline{\overline{s}} \mid \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid s.(s.q.(\underline{\overline{k}} \mid n) \mid \overline{l} \mid l.q.y) \twoheadrightarrow \\ P_2 &= \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid s.q.(\underline{\overline{k}} \mid n) \mid \overline{l} \mid l.q.y \mapsto \\ P_3 &= \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid s.q.(\underline{\overline{k}} \mid n) \mid q.y \twoheadrightarrow \\ P_4 &= k.(s \mid q \mid \underline{\overline{k}}) \mid s.q.(\underline{\overline{k}} \mid n) \mid y \end{aligned}$$

where P_4 does not allow any further transition and $\mathrm{Obs}(\mathcal{C}) = \{y\}$, as required by a LMS system of size 1.

For $k = 2$, $\mathsf{N}et_2 = P \mid P$, $\mathsf{N}et_2$ has several possible computations $\mathcal{C}$ of the same length leading to the same final state

$$Q_9 = s \mid n \mid q.y \mid s \mid q \mid \overline{\underline{k}} \mid D$$

where

$$D = s.(s.q.(\overline{\underline{k}} \mid n) \mid \bar{l} \mid l.q.y)$$

One of these computations is

$$\begin{aligned}
\mathsf{N}et_2 &= P \mid P \mapsto P_1 \mid P \twoheadrightarrow P_2 \mid P \twoheadrightarrow \\
Q_3 &= \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid q.(\overline{\underline{k}} \mid n) \mid \bar{l} \mid l.q.y \mid \\
&\quad \overline{m} \mid \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid m.D \twoheadrightarrow \\
Q_4 &= k.(s \mid q \mid \underline{\overline{k}}) \mid \overline{\underline{k}} \mid n \mid \bar{l} \mid l.q.y \mid \\
&\quad \overline{m} \mid \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid m.D \twoheadrightarrow \\
Q_5 &= s \mid q \mid \overline{\underline{k}} \mid n \mid \bar{l} \mid l.q.y \mid \\
&\quad \overline{m} \mid \underline{\overline{q}} \mid k.(s \mid q \mid \underline{\overline{k}}) \mid m.D \twoheadrightarrow \\
Q_6 &= s \mid q \mid n \mid \bar{l} \mid l.q.y \mid \\
&\quad \overline{m} \mid \underline{\overline{q}} \mid s \mid q \mid \overline{\underline{k}} \mid m.D \twoheadrightarrow \\
Q_7 &= s \mid n \mid \bar{l} \mid l.q.y \mid \\
&\quad \overline{m} \mid s \mid q \mid \overline{\underline{k}} \mid m.D \mapsto \\
Q_8 &= s \mid n \mid q.y \mid \\
&\quad \overline{m} \mid s \mid q \mid \overline{\underline{k}} \mid m.D \mapsto \\
Q_9 &= s \mid n \mid q.y \mid \\
&\quad s \mid q \mid \overline{\underline{k}} \mid D
\end{aligned}$$

We have that Q_9 does not allow any further reduction. For each computation $\mathcal{C}$ leading to Q_9, $\mathrm{Obs}(\mathcal{C}) = \{n\}$ $\forall C$, as required by a LMS system of size 2. In fact, while $Q_9 \downarrow n$, any barb y is guarded by two inputs on channels l and q but each high-priority output $\underline{\overline{q}}$ is consumed before any low-priority output $\bar{l}$ may react.

For $k \geq 3$, $\mathsf{N}et_k$ has k^2 possible initial (low-priority) reductions on channel m which lead to congruent states. Like for $\mathsf{N}et_2$ the first reductions happen respectively

on channels m, s, s, q:

$$\mathsf{N}et_k \;\mapsto\; P_1 \mid \underbrace{P \mid \cdots \mid P}_{k-1} \twoheadrightarrow \cdots \twoheadrightarrow Q_4 \mid \underbrace{P \mid \cdots \mid P}_{k-2} = \mathsf{N}et_{k4}$$

For $k \geq 3$, a chain of high-priority reductions happen on channels k, s, q until every input on k is consumed. After this chain of high-priority reductions every output $\underline{s}, \underline{q}$ is consumed and no other observables $\{y, n\}$ can be exhibited. After other k low-priority transitions (one on l and $k-1$ on m) no further reduction is possible. Since for every computation no barb on y is exhibited, $\mathsf{N}et_k$ is a LMS system. □

Lemma 3.5 *Let P be a CPG process,*

$$P = a : b.\overline{a} \mid \overline{b}.(b : \tau.y \mid z : b.(\overline{b} \mid n \mid \overline{z}))$$

Then $\mathsf{N}et_k = \underbrace{P \mid \cdots \mid P}_{k}$ *is a LMS system of size k, $\forall k \geq 1$.*

Proof: For $k = 1$, $\mathsf{N}et_1 = P$, $\mathsf{N}et_1$ has only one maximal computation $\mathcal{C}$

$$P \rightarrow P_1 = \overline{a} \mid b : \tau.y \mid z : b.(\overline{b} \mid n \mid \overline{z}) \rightarrow P_2 = \overline{a} \mid y \mid z : b.(\overline{b} \mid n \mid \overline{z})$$

where P_2 does not allow any further transition and $\mathrm{Obs}(\mathcal{C}) = \{y\}$, as required by a LMS system of size 1.

For $k = 2$, $\mathsf{N}et_2 = P \mid P$, $\mathsf{N}et_2$ has two possible computations $\mathcal{C}, \mathcal{C}'$ of length 2 leading to states congruent to Q_2:

$$\mathsf{N}et_2 = P \mid P \rightarrow P_1 \mid P \rightarrow$$
$$Q_2 = \overline{a} \mid b : \tau.y \mid \overline{b} \mid n \mid \overline{z} \mid a : b.\overline{a} \mid b : \tau.y \mid z : b.(\overline{b} \mid n \mid \overline{z})$$

Q_2 is a final state and for both computations $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(C') = \{n\}$ as required by a LMS system of size 2.

For $k \geq 3$, $\mathsf{N}et_k$ has $k \cdot (k-1)$ possible computations of length 2 leading to states congruent to $\mathsf{N}et_{k2}$:

$$\mathsf{N}et_k \rightarrow P_1 \mid \underbrace{P \mid \cdots \mid P}_{k-1} \rightarrow Q_2 \mid \underbrace{P \mid \cdots \mid P}_{k-2} = \mathsf{N}et_{k2}$$

$\mathsf{N}et_{k2}$ is a final state and for any computation $\mathcal{C}$ of $\mathsf{N}et_k$, $\mathrm{Obs}(\mathcal{C}) = \{n\}$ as required for a LMS system of size k. □

The following is an alternative way with respect to Theorem 3.2 to prove the separation between FAP and π-calculus and acts as a template for the next theorems.

Theorem 3.5 *There is no distribution-preserving and weakly-observation-respecting encoding $[\![\cdot]\!]$ of FAP into the π-calculus.*

Proof: Suppose $[\![\cdot]\!]$ is distribution-preserving and weakly-observation-respecting. By Lemma 3.4 $\exists P : \mathsf{N}et_k$ is a LMS system for any $k \geq 1$, where $\mathsf{N}et_k = P \mid \ldots \mid P$. By the weakly-observation-respecting condition, $[\![\mathsf{N}et_1]\!]$ has a computation $\mathcal{C}_1$ with observables $\mathrm{Obs}(\mathcal{C}_1) = \{y\}$. By the distribution-preserving condition

$$[\![\mathsf{N}et_k]\!] = [\![P \mid \cdots \mid P]\!] = [\![P]\!] \mid \cdots \mid [\![P]\!] = [\![\mathsf{N}et_1]\!] \mid \cdots \mid [\![\mathsf{N}et_1]\!]$$

By Lemma 3.1 then there exists a maximal computation $\mathcal{C}_k$ of $[\![\mathsf{N}et_k]\!]$ such that $\mathrm{Obs}(\mathcal{C}_1) = \{y\} \subseteq \mathrm{Obs}(C_k)$, while no computation of $\mathsf{N}et_k$ contains observable y for $k \geq 2$, which contradicts the weakly-observation-respecting property of the encoding function $[\![\cdot]\!]$. □

Theorem 3.6 *There is no distribution-preserving and weakly-observation-respecting encoding of CPG in the π-calculus.*

Proof: By Lemma 3.5 exactly as for Theorem 3.5. □

Theorem 3.7 *There is no distribution-preserving and weakly-observation-respecting encoding of FAP into the bπ-calculus.*

Proof: By Lemmas 3.2 and 3.4, exactly as for Theorem 3.5. □

Theorem 3.8 *There is no distribution-preserving and weakly-observation-respecting encoding of CPG into the bπ-calculus.*

Proof: By Lemmas 3.2 and 3.5, exactly as for Theorem 3.5. □

3.4 Discussion

We have considered FAP, a finite fragment of CCS extended with global priority, and we have proved, by means of leader-election-based separation results, that it is not possible to encode it in CPG under uniformity and independence-preserving conditions on the encoding, thus providing the first expressiveness separation result between global and local priority within a process algebra framework. We have then proved that FAP cannot be distributively translated into the π-calculus even if allowing partially correct implementations, i.e. encodings which may introduce divergence in computations or also premature termination caused by deadlock.

We have then analysed another setting, called the last man standing (LMS) problem, which allows us to considerably strengthen the separation between prioritised (with both global or local priority) languages and non prioritised ones, by showing that even if we equip the language with broadcast-based primitives like the bπ-calculus, the expressiveness of priority cannot be obtained under parallel-preserving conditions.

In other words we have shown that, within the context of the process algebras considered here, it is not possible to have a distribution-preserving encoding of either global or local priority in non-prioritised languages even if we admit asymmetric translations or divergence/failure in the computation as a consequence of livelocks or deadlocks. This impossibility result does not depend on the capability of communication of names or values, synchrony or asynchrony of the output, scope extrusion, choice on the available input/outputs, recursion or replication, point-to-point or broadcast communication/synchronisation type, nor on Turing-completeness: in fact the non-encodability of FAP into π-calculus, bπ-calculus and CPG holds even if FAP is the one not Turing-complete amongst the calculi considered here and also the other non-encodability results would still hold by considering finite variants of the respective calculi. Therefore, such impossibility depends only on the power of the instantaneous preemption characteristic of the prioritised languages analysed here. As a consequence we can see that it is not possible to make any purely distributed

implementation of such kinds of priority on top of standard process calculi even if admitting good or randomised encodings like those considered in [72, 77] for the implementation of the choice operator. The strength of the separation suggests also that any encoding trying to preserve some relaxed condition on the distribution may be affected by severe performance issues due to the further synchronisations needed to preserve the constraint of instantaneous preemption.

Chapter 4

The π@ Calculus

In this chapter we present the π@ language — a simple, conservative and powerful extension of the π-calculus, with two new basic mechanisms: polyadic synchronisation and prioritised communication. In particular, polyadic synchronisation [18] allows the modeling of compartments in a natural way, but still in the classic message-passing flavour typical of the π-calculus; priority instead is extremely useful for implementing transactional mechanisms that are essential when dealing with compartment operations (e.g., dissolution of a membrane) that involve many components that are to be updated as a result. 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 π@ theoretical analysis. On the other hand, it is so powerful that it can be used with a pivotal role for comparing the various compartment-based formalisms. In this sense, we claim that π@ is the right compromise between the simple and elegant theory of π-calculus and the biological needs for modeling compartmentalised behavior. In order to match these expectations even more strikingly, we define a simpler sublanguage, called *core-*π@, where polyadic synchronisation is limited to its simplest form (two names at most are used in a channel name) and priority levels can only be two, and we show its great flexibility in modelling biological systems.

The chapter is structured as follows. Section 4.1 introduces polyadic synchronisation and Sect. 4.2 a form of prioritised communication, which constitute the

basic ingredients added to the π-calculus to obtain the language π@. This language is described in Section 4.3, with its syntax and reduction semantics. Moreover, the same biological case-study of the insulin secretion process introduced in Chap. 2 is now modeled in π@, showing that the new representation is much more faithful. Section 4.3.2 ends with the presentation of core-π@, the minimal subcalculus of π@ that we claim is powerful enough to model complex biological systems.

4.1 Polyadic Synchronisation

In the π-calculus, channels and transmitted names are usually synonyms. Polyadic synchronisation [18] consists in giving *structure* to channels: each channel is composed of one or more names and identified by all of them in relation to the exact sequence of their occurrence. 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 nonexistent, address. Similarly, polyadic synchronisation (in its simplest form) provides the capability of writing channels as $name_1@name_2$. In other words, a channel is indicated by a vector of two names $(name_1, name_2)$ and communication between two processes may happen only if they are pursuing a synchronisation along channels denoted by the same names.

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

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

produces the same renaming effect of a π-calculus transition, but with one difference: in the π-calculus, the transmission of a name always stands for the transmission of a channel, while in the above example the transmitted name constitutes only one component of it.

An extended form of polyadic synchronisation allows for the use of more than two names for each channels, like in the following example:

$$\overline{@c_1@c_2@c_3}\langle d\rangle.P \mid c_1@c_2@c_3(x).Q \quad \rightarrow \quad P \mid Q\{d/x\}$$

In general, there is no limit to the length of the vector of names representing a channel.

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

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

The benefits in terms of simplicity of biological modellings are immediate in presence of compartments. For example, if the π-calculus process

$$M(m_1,\ldots,m_n) \triangleq \pi.M' + \ldots$$

is a molecule subjected to movement across the cellular membrane (like the GLU process of Expr. (2.4)), all of its channels must be restricted after its conveyance into the cell:

$$M(m_1^{out},\ldots,m_n^{out}) \mid (\nu\ m_1^{in},\ldots,m_n^{in},\ldots)(M(m_1^{in},\ldots,m_n^{in}) \mid \cdots)$$

In particular, any channel protein able to convey the above molecule across different compartments should be aware of all the names $m_1^{out},\ldots,m_n^{out},m_1^{in},\ldots,m_n^{in}$. Polyadic synchronisation allows us to model the compartment just as *one* restricted name. For example, the above system may be converted as

$$M(m_1,\ldots,m_n,c_{out}) \mid (\nu\ c_{in},\ldots)(M(m_1,\ldots,m_n,c_{in}) \mid \cdots)$$

where each input or output action $\pi.M'$ inside M has been encoded as $\pi@c.M'$, and c represents the compartment M lies in (c_{out} or c_{in} in the above example). In this way, only one restricted name is needed to formalise a new compartment, regardless of all the channels the enclosed processes may use. This substantial simplification affects also the formalisation of cross-compartment processes: they need to handle only one additional name for each compartment they partially reside in.

4.2 Priority

The idea behind the notion of priority applied here to the π-calculus [33] is very similar to the mechanisms adopted for the implementation of schedulers which allow us to give processes several levels of priority, depending on their requirements (responsiveness, cpu load, real-time constraints, etc) for the task they accomplish.

Here we consider a particular kind of priority characterised by global, immediate preemption (the same considered for the definition of the fragment FAP in Chap. 2): each process denoted by high priority holds the central processing unit and executes its job before any low priority process may perform some other task. In π-calculus setting, this is equivalent to forcing high priority synchronisations or communications to happen before any low priority action. A high priority action is indicated by underlining the name of the channel. For example, the expression

$$\bar{l}\langle a\rangle.P \mid \underline{\bar{h}}\langle b\rangle.Q$$

contains two processes with different, increasing priority. In the above situation, both are blocked: in fact, no other process is ready to receive over the channel l or h. In presence of some process listening on channel l, the first process may react in the following way:

$$\bar{l}\langle a\rangle.P \mid l(x).P' \mid \underline{\bar{h}}\langle b\rangle.Q \quad \rightarrow \quad P \mid P'\{a/x\} \mid \underline{\bar{h}}\langle b\rangle.Q$$

In fact, this would be the only possible transition since no other reduction is available. Conversely, when both high and low priority actions are enabled, low-priority synchronisations can occur only after high-priority ones:

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

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

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 sake of clarity, communications between channels with the same name but different priorities are forbidden. For example, in the following system there are no possible reductions:

$$\underline{\overline{c}}\langle a\rangle.P \mid c(x).Q$$

An additional level of priority can be denoted by double underlining a channel name:

$$\overline{l}\langle a\rangle.P \mid \underline{\overline{h}}\langle b\rangle.Q \mid \underline{\underline{\overline{u}}}\langle c\rangle.R$$

If any reduction is available over the $\underline{\underline{u}}$ channel, it preempts any reduction over $\underline{h}$ and l. Additional levels of priority require the introduction of integer numbers as labels added to each input/output operation. For this reason the above expression may be also written as follows:

$$\overline{2:l}\langle a\rangle.P \mid \overline{1:h}\langle b\rangle.Q \mid \overline{0:u}\langle c\rangle.R$$

where 0 denotes the highest level of priority. The choice of identifying higher priority levels with decreasing integer numbers is arbitrary and does not affect the semantics or expressiveness of the language.

Since three levels of priority suffice for our encoding purposes, only the first of these two syntaxes will be used in the encodings and modelling examples, for its conciseness and readability.

For a detailed survey of priority in process algebras, we refer to [33].

At first sight, the idea of priority seems foreign and unnecessary for biological modelling purposes. However, the complexity of this realm makes almost impossible the design of a suitable and complete modelling language. The presence of several priority levels for operations allows the composition of high-level, complex operations as sequences of several simple, low-level steps (transitions) by avoiding any interference of external processes with the involved elements. Such low-level transitions can be composed in different ways, depending on the abstraction adopted to formalise the system of interest. This effect can be achieved by encoding each

high-level operation as a list of high priority actions, preceded by a single low priority operation which act as a *guard*. For example, if the processes P_1, P_2, P_3, P_4 represents four proteins which can bind together and form a new complex C, they can be modelled in π-calculus by exploiting the restriction operator and representing the binding as the sharing of a private name:

$$(\nu\, b)P_1 \mid P_2 \mid P_3 \mid P_4 \quad \rightarrow \quad \cdots \quad \rightarrow \quad (\nu\, b)\big(P_1' \mid P_2' \mid P_3' \mid P_4'\big)$$

Since more than one transition is needed to accomplish the whole process, if one of the requirements is its atomicity (this is often necessary when the modelled processes may meanwhile interact with other elements and give rise to undesirable situations not pertaining to the original model) then there is no way to obtain a satisfactory modelling in π-calculus. Priority instead allows to ensure that no other process can interfere during the formation of the above complex. The four molecules can be formalised as

$$P_1 \triangleq \overline{l}\langle b\rangle.\overline{\underline{h_1}}\langle b\rangle.\overline{\underline{h_2}}\langle b\rangle.P_1' \qquad P_2 \triangleq l(x).P_2'$$
$$P_3 \triangleq \underline{h_1}(x).P_3' \qquad P_4 \triangleq \underline{h_2}(x).P_4'$$

so that after the first synchronisation of P_1 and P_2 over the low priority channel l, a sequence of (two) high priority communications between P_1 and P_3, P_4 is triggered and cannot be interrupted by any other process. In fact, the possibility of performing the initial low priority action over l guarantees that any other high priority operation occasionally available in the system has been previously consumed.

4.3 The π@ language

The π@ language joins the expressiveness of polyadic communication and priority in order to model both localisation of processes inside compartments and atomicity of complex operations that require more than one reduction step for their completion. Thanks to the simplicity of such extensions, π@ is very close to the π-calculus: from a syntactical point of view the only difference is the structure of channels, composed

of multiple names and tagged by the priority of the action. We use μ to denote a vector of names $x_1, \ldots, x_n$ and $k : \mu$ to denote a channel, that is a natural number k specifying the priority level followed by a vector of names μ. In particular, $\overline{k : \mu}$ represents an output operation along channel $k : \mu$, while $k : \alpha$ stands for a generic input, output or silent action τ of priority level k.

Definition 4.1 *Let*

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

$\mathcal{N}^+ = \bigcup_{i>0} N^i \quad , \qquad \mu \in \mathcal{N}^+$;

$\overline{\mathcal{N}}^+ = \{\overline{\mu} \mid \mu \in N^+\}$;

$\alpha \in \left(\overline{\mathcal{N}}^+ \cup \mathcal{N}^+ \cup \{\tau\} \right)$;

The syntax of $\pi@$ *defined in terms of the following grammar:*

$$\begin{array}{lcl} P & ::= & \mathbf{0} \;\Big|\; \sum_{i \in I} \pi_i.P_i \;\Big|\; P \mid Q \;\Big|\; !\,P \;\Big|\; (\nu\, x)P \\ \pi & ::= & k :\!: \tau \;\Big|\; k : \mu(x) \;\Big|\; \overline{k : \mu}\langle x \rangle \end{array}$$

As previously introduced, the following abbreviations are used for readability:

$$\begin{array}{ll} \mu(x) = 2 : \mu(x) & \overline{\mu}\langle x \rangle = \overline{2 : \mu}\langle x \rangle \\ \underline{\mu}(x) = 1 : \mu(x) & \overline{\underline{\mu}}\langle x \rangle = \overline{1 : \mu}\langle x \rangle \\ \underline{\underline{\mu}}(x) = 0 : \mu(x) & \overline{\underline{\underline{\mu}}}\langle x \rangle = \overline{0 : \mu}\langle x \rangle \end{array}$$

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

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

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 4.2 *Let $I^k(P)$ be*

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

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

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

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

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

$\pi@$ reduction rules are exactly the same as those of the 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.

4.3.1 Modelling the insulin example in π@

The benefits deriving from polyadic synchronisation and priority for biological modellings can be noticed as long as even very simple systems are formalised, like the one in Fig. 2.1 already described in the π-calculus. We have previously discussed why a faithful description of a compartment in π-calculus implies that all the names of the enclosed processes are restricted, in order to prevent their interaction with the external environment. On the contrary, π@ allows the characterisation of each compartment by means of a single restricted name, as shown in Sect. 4.1. Therefore, the shape of the system of Fig. 2.1 becomes

$$(\nu\, c_{in})\big(CHAN_1 \mid \cdots \mid CHAN_l \mid MOL_1 \mid \cdots \mid MOL_n \mid VES \mid \cdots \mid VES\big)$$

where the cell is represented by the restricted name c_{in}, the cross-compartment processes $CHAN_1, \ldots, CHAN_l$ communicate to the external environment by a corresponding name c_{out}, and the set of free names of the processes MOL_1, ..., MOL_n, VES is irrelevant for their confinement inside the compartment, as soon as c_{out} does not appear in such set and c_{in} constitutes one of the names which identify each channel.

Therefore, the GLU process of Expr. (2.4) may be translated as follows:

$$\begin{aligned} GLU(comp, gt, gr) \triangleq (\nu\, g) \\ \big(\overline{g}\langle comp\rangle \mid \quad !\, g(c).(gt@c(c_{new}).\overline{g}\langle c_{new}\rangle + \overline{gr@c}.PYR)\big) \end{aligned} \tag{4.1}$$

The behaviour of the process is almost the same: replication is exploited for encoding the recursive behaviour of the molecule, which is always ready either to react or to be moved into another compartment. These two capabilities are formalised by an interaction over $gr@c$ and $gt@c$ respectively, where c represents the name of the compartment the molecule lies in. The movement of the molecule to another compartment is reflected by receiving a new compartment name and forgetting the previous one. Since such a name appears in all the channels that allow the molecule to interact with the other processes, GLU can interact only with elements enclosed within the same compartment. The simplification of Expr. (2.4) with respect to

Expr. (4.1) is minimal for the reason that the description of the GLU molecule includes only two public channels, but it becomes significant as soon as their number grows.

It is worth remarking that the current semantics of GLU may not correspond to the intended behaviour of the glucose molecule in the original biological model: if it requires the movement of the molecule across compartments to be instantaneous, then Expr. (4.1) does not constitute a correct formalisation because of the additional internal step needed to spawn the molecule inside the target compartment. Priority addresses exactly this very common modelling problem: it is possible to use high priority reductions to model all the operations that in the original model are purely atomic or not present at all:

$$\begin{aligned} GLU(comp, gt, gr) \triangleq\ & (\nu\ g) \\ & \big(\overline{g}\langle comp\rangle \ \big|\ \ !\ \underline{g}(c).(gt@c(c_{new}).\overline{g}\langle c_{new}\rangle + \overline{gr@c}.PYR)\big) \end{aligned} \tag{4.2}$$

The corresponding expression of the $GLUT2$ glucose transporter can be slightly simplified as well:

$$GLUCHAN(c_{out}, c_{in}, gt) \triangleq\ \ !\ \overline{gt@c_{out}}\langle c_{in}\rangle$$

Now, the channel process does not need to know all the names of the transported molecule. This is a very valuable property, since it means that there is no need to change the expression of the glucose transporter each time the formalisation of GLU changes, for example after the addition of new reaction capabilities. As previously discussed, $GLUCHAN$ is characterised by the presence of two compartment names, for the reason that it represents a cross-compartment molecule.

The processes representing ADP and ATP are almost unchanged:

$$\begin{aligned} ADP(glureact, inhk, c) &\triangleq\ \ \overline{glureact@c}.ATP(inhk, c) \\ ATP(inhk, c) &\triangleq\ \ \overline{inhk@c} \end{aligned}$$

In this case the use of polyadic synchronisation seems to make more complicated the expression of the above molecules, but the benefit will appear after the formalisation of the whole cell.

For the description of the potassium molecule K the same principles leading to Expr. (4.2) can be applied:

$$K(kt, kr, comp) \triangleq$$
$$(\nu\ k)\big(\overline{\underline{k}}\langle comp\rangle \mid !\ \underline{k}(c).(kt@c(c_{new}).\overline{\underline{k}}\langle c_{new}\rangle + \overline{kr@c})\big)$$

Even in this case the high priority of internal reductions can provide a more faithful modelling, like for the potassium channel $KCHAN$:

$$KCHAN(c_{in}, c_{out}, inhk) \triangleq$$
$$(\nu\ kc)\big(\overline{\underline{kc}} \mid !\ \underline{kc}.(inhk + \overline{kt@c_{in}}\langle c_{out}\rangle.\overline{\underline{kc}})\big)$$

The remaining processes can be translated in the same way:

$$\begin{aligned}
POL(c, kreact, caact) &\triangleq kreact@c.\overline{caact@c} \\
CACHAN(c_{out}, c_{in}, cat, caact) &\triangleq caact@c_{in}.!\ \overline{cat@c_{out}}\langle c_{in}\rangle \\
CA(comp, cat, car) &\triangleq (\nu\ ca)\big(\overline{\underline{ca}}\langle comp\rangle \mid \\
&\qquad !\ \underline{ca}(c).(cat@c(c_{new}).\overline{\underline{ca}}\langle c_{new}\rangle + \overline{car@c})\big) \\
DOCKP(c_{in}, car, dockves, c_{out}) &\triangleq car@c_{in}.\overline{dockves@c_{in}}\langle c_{out}\rangle \\
INS(c, ins) &\triangleq \overline{ins@c}
\end{aligned}$$

Since the vesicle VES constitutes another, nested compartment inside the cell, it can be as well represented by a new restricted name:

$$\begin{aligned}
VES(c_{out}, dv, ins) \triangleq\ & (\nu\ vc_{in})\big(dv@c_{out}(c_{ext}).!\ ins@vc_{in}.INS(c_{ext}, ins) \\
& \mid INS(vc_{in}, ins) \mid \cdots \mid INS(vc_{in}, ins)\big)
\end{aligned}$$

The internal compartment name for VES is vc_{in}, while c_{out} stands for the name corresponding to the cell compartment. c_{ext} will be bound to the name of the compartment surrounding the cell, where the insulin molecules will be dumped after the exocytosis of the vesicle VES.

If the process of exocytosis is abstracted as a single, atomic operation, then priority can be used for the loop of the process expelling all the insulin molecules.

It may be modified as follows:

$$dv@c_{out}(c_{ext}).!\ \underline{ins@vc_{in}}.INS(c_{ext}, ins)$$

and the INS molecules accordingly:

$$INS(c, ins) \triangleq \quad \overline{\underline{ins@c}}$$

In this way any sequence of movement of INS molecules would appear as atomic. The presence of the high priority action immediately after the replication constitutes a very dangerous operation: a high priority loop of this kind may block the whole system if there is no guarantee of its termination. In the case of the VES process the finite number of INS molecules guarantees such termination.

Finally, the whole system is represented as follows:

$$\begin{aligned} SYS \triangleq\ & GLU(c_{out}, gt, gr) \mid K(c_{out}, kt, kr) \mid Ca(c_{out}, cat, car) \mid \\ & (\nu\ c_{in}) \\ & \Big(GLU(c_{in}, gt, gr) \mid GLUCHAN(c_{out}, c_{in}, gt) \\ & \mid K(kt, kr, c_{in}) \mid KCHAN(c_{in}, c_{out}, inhk) \mid \\ & \mid ADP(gr, inhk, c_{in}) \mid ATP(inhk, c_{in}) \mid POL(c_{in}, kr, caact) \\ & \mid Ca(c_{in}, cat, car) \mid CACHAN(c_{out}, c_{in}, cat, caact) \\ & \mid DOCKP(c_{in}, car, dockves, c_{out}) \mid VES(c_{out}, dockves, ins)\Big) \end{aligned}$$

Like the VES compartment, the β cell is denoted by only one restricted name, c_{in}.

4.3.2 The core-π@ language

The π@ language is characterised by the capability of using an unbounded number of names for channels and of priority levels for reductions. The writing of complex encodings like those presented in Sect. 5.1.3 and 5.1.4 can be significantly simplified by such flexibility. However, two levels of priority and two names for each channel are sufficient for most purposes, like the modelling examples previously shown. For

this reason we introduce a subcalculus of $\pi@$, denoted as *core-*$\pi@$, characterised by only two levels of priority (i.e. normal actions and prioriotised actions) and two names for each channel. A valuable property of this core-$\pi@$ is its straightforward mapping into its stochastic counterpart S$\pi@$ [95, 96].

We introduce three distinct sets of names $\mathcal{N}, \mathcal{P}, \mathcal{C}$ denoting respectively unprioritised actions, prioritised actions and compartments. Each channel $x@a$ is denoted by an action name and a compartment name. In order to keep notation simple, compartment names may be omitted when superfluous.

Definition 4.3 *Let $\mathcal{N}, \mathcal{P}, \mathcal{C}$ be distinct sets of names on a finite alphabet, with m, n ranging over $\mathcal{N}$, p, q over $\mathcal{P}$, a, b over $\mathcal{C}$ and x, y over $\mathcal{X} = N \cup P \cup C$. The syntax of the core-$\pi@$ language is defined as*

$$\begin{array}{rcl} P & ::= & \mathbf{0} \;\Big|\; \displaystyle\sum_{i \in I} \pi_i.P_i \;\Big|\; P \mid Q \;\Big|\; !\,\pi.P \;\Big|\; (\nu\, x)P \\ \pi & ::= & \tau \;\Big|\; n@a(\mathbf{x}) \;\Big|\; \overline{n}@a\langle \mathbf{x} \rangle \;\Big|\; \underline{\tau} \;\Big|\; \underline{p}@a(\mathbf{x}) \;\Big|\; \underline{\overline{p}}@a\langle \mathbf{x} \rangle \end{array}$$

where $\mathbf{x}$ represents one or more names $x_1, \ldots, x_i$ ranging over $\mathcal{X}$.

Like for $\pi@$, the semantics of this core-$\pi@$ is given by means of a reduction system based on the following congruence relation.

Definition 4.4 *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:*

$$\begin{array}{rcll} (\nu\, x)P \mid Q & \equiv & (\nu\, x)(P \mid Q) & \text{if } x \notin \mathrm{fn}(Q) \\ (\nu\, x)P & \equiv & P & \text{if } x \notin \mathrm{fn}(P) \\ !\,\pi.P & \equiv & \pi.(!\,\pi.P \mid P) & \text{if } \mathrm{fn}(\pi) \cap \mathrm{bn}(\pi) = \emptyset \end{array}$$

where the function fn *is defined as*

$$\begin{array}{rlrl}
\mathrm{fn}(\tau) &\triangleq \emptyset & \mathrm{fn}(\underline{\tau}) &\triangleq \emptyset \\
\mathrm{fn}(n@a(\mathbf{x})) &\triangleq \{n, a\} & \mathrm{fn}(\overline{n}@a\langle\mathbf{x}\rangle) &\triangleq \{n, a, \mathbf{x}\} \\
\mathrm{fn}(\underline{p}@a(\mathbf{x})) &\triangleq \{p, a\} & \mathrm{fn}(\overline{\underline{p}}@a\langle\mathbf{x}\rangle) &\triangleq \{p, a, \mathbf{x}\} \\
\mathrm{fn}(\mathbf{0}) &\triangleq \emptyset & \mathrm{fn}((\nu\ x)P) &\triangleq \mathrm{fn}(P) \setminus \{x\} \\
\mathrm{fn}(\pi.P) &\triangleq \mathrm{fn}(\pi) \cup \mathrm{fn}(P) & \mathrm{fn}(\sum_{i\in I} \pi_i.P_i) &\triangleq \bigcup_i \mathrm{fn}(\pi_i.P_i) \\
\mathrm{fn}(P \mid Q) &\triangleq \mathrm{fn}(P) \cup \mathrm{fn}(Q) & \mathrm{fn}(!\,\pi.P) &\triangleq \mathrm{fn}(\pi.P)
\end{array}$$

Definition 4.5 *core-$\pi@$ semantics is given in terms of the following reduction system:*

$$\frac{}{\underline{\tau}.P + M \ \twoheadrightarrow\ P} \qquad \frac{M \ \not\twoheadrightarrow\ M'}{\tau.P + M \ \mapsto\ P} \qquad \frac{P \ \twoheadrightarrow\ P'}{(\nu\ x)P \ \twoheadrightarrow\ (\nu\ x)P'} \qquad \frac{P \ \mapsto\ P'}{(\nu\ x)P \ \mapsto\ (\nu\ x)P'}$$

$$\frac{}{(\underline{p}@a(\mathbf{x}).P + M) \mid (\overline{\underline{p}}@a\langle\mathbf{y}\rangle.Q + N) \ \twoheadrightarrow\ P\{\mathbf{y}/\mathbf{x}\} \mid Q}$$

$$\frac{M \mid N \ \not\twoheadrightarrow\ R}{(n@a(\mathbf{x}).P + M) \mid (\overline{n}@a\langle\mathbf{y}\rangle.Q + N) \ \mapsto\ P\{\mathbf{y}/\mathbf{x}\} \mid Q}$$

$$\frac{P \ \twoheadrightarrow\ P'}{P \mid Q \ \twoheadrightarrow\ P' \mid Q} \qquad \frac{P \ \mapsto\ P' \qquad P \mid Q \ \not\twoheadrightarrow\ R}{P \mid Q \ \mapsto\ P' \mid Q}$$

$$\frac{P \equiv Q \quad P \ \twoheadrightarrow\ P' \quad P' \equiv Q'}{Q \ \twoheadrightarrow\ Q'} \qquad \frac{P \equiv Q \quad P \ \mapsto\ P' \quad P' \equiv Q'}{Q \ \mapsto\ Q'}$$

The presence of only one additional level of priority allows avoidance of the definition of the $I^k()$ function of Def. 4.2. On the other hand, the definition of the reduction relation requires two rules for each corresponding rule of $\pi@$ semantics.

Chapter 5

Encodings in π@

In this chapter the expressive capabilities of π@ are demonstrated by means of the definition of the encoding functions of various bio-inspired formalisms. In the next section the encodings of BioAmbients and Brane Calculi are discussed, while in Sect. 5.2 the encoding in π@ of a variant of P Systems is treated.

5.1 Encoding Bio-inspired Calculi in π@

The key feature which differentiates many recent bio-inspired calculi from the π-Calculus is the explicit formalisation of compartments. BioAmbients is a modified version of the Ambient calculus [23, 20], 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 dynamically 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, which corresponds to the preservation of the parity of the nesting level of processes. As remarked in [22], this peculiarity is enough to preclude an immediate embedding of one language into the other.

Consequently, on the one hand they gain faithfulness because of their additional

primitives designed to model the addressed biological phenomena, on the other their specialisation does not allow us to mutually translate the models expressed in each language. Furthermore, the high abstraction level of such primitives hides the mechanisms underlying the idea of compartment, whose unfolding can reveal their strong resemblance.

π@ features were chosen to overcome all these issues: 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 π@ one abstraction level underneath, as a sort of *assembly* language for compartmentalised formalisms. As previously discussed, it allows consistent simplification with respect to the π-Calculus the formalisation of biological models which embed the notion of compartment. In this section we show that π@ is also able to supply the same high level features offered by bio-oriented languages like BioAmbients and Brane calculi. In particular, we show how both of them can be encoded directly in π@, by unfolding the basic functioning of compartment semantics and providing a common platform for their direct comparison and implementation.

5.1.1 Basic ideas

Compartments 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 π@ 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 π@. As suggested in [68], these kind of structures can be represented as a set of processes linked by the share of private channels between parent and child nodes. The encoding of the insuline secretion process is a simple example of this situation: the cell and the insuline vesicles define the

boundaries of compartments linked by the presence of cross-compartment processes (like cross-membrane molecular channels) which are able to interact with elements located in both compartments. Like in [89], the scoping of private names represents the boundaries of such compartments, but thanks to polyadic synchronisation each private name may represent an unlimited number of private communication channels, as discussed during the modelling of the insuline example. 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 BioAmbients $merge+/merge–$ and Brane $exo/exo^{\perp}$ operations. In $\pi@$ this turns out to be a sort of *multicast* communication, where specific groups of nodes – that is sibling and child processes – must receive on the appropriate 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:

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

The above process can be triggered by an output operation $\overline{\underline{\underline{bcast}}}\langle chn, newchn\rangle$ and terminate when no high priority synchronisations are available, leaving no residual terms. Obviously, a high priority complementary output loop $!\,\overline{\underline{\underline{bcast}}}\langle chn, newchn\rangle$ would cause the system to hang, since it prevents any other computation with normal priority. The avoidance of such high priority and non-terminating loops is often not trivial. In particular, a homomorphic translation of the replication operator would immediately cause them to appear, as we will discuss in the following encodings. This is one of the most difficult translation issues and will force us to represent *indirectly* the encoded replicated processes, by keeping explicit track of each replicated instance in our encoding functions.

5.1.2 Requirements

The fundamental criterion guiding any encoding is the preservation of some addressed semantics. While the weak kind of observational semantics considered in Sect. 3.2 turned out to be appropriate for the addressed *negative* results (i.e. of impossible encodability), some more strict semantics should be considered when providing *positive* results (i.e. of possible encodability). According to [73], this means that the encoding function $[\![\cdot]\!]$ should 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. As usual for concurrent languages we also require some additional criteria. As remarked in [76], a *reasonable* encoding should also preserve the degree of distribution of the source language (i.e. homomorphism with respect to 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 [40]. In addition to the cited criteria, we require that the encoding preserves the termination (denoted as $\Downarrow$) or diverging (denoted as $\Uparrow$) behaviour of the translated term, in order to obtain a totally faithful encoding function. The following definition formalises the notion of *suitable encoding* used in this chapter, which embeds all the properties discussed above.

Definition 5.1 *An encoding $[\![\cdot]\!]$ is* suitable *if it enjoys the following properties:*

1. *homomorphism with respect to 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:

$$P\Downarrow \iff [\![P]\!]\Downarrow \qquad\qquad P\Uparrow \iff [\![P]\!]\Uparrow$$

4. operational correspondence:

(a) if $P \to P'$ *then* $[\![P]\!] \to^* [\![P']\!]$,

(b) if $[\![P]\!] \to^* Q$ *then* $\exists P' : P \to^* P' \wedge Q \to^* [\![P']\!]$.

5.1.3 Encoding BioAmbients in $\pi@$

As we have just seen, 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 interact with zero or more nested sub-ambients. Consequently, for the implementation of this tree structure in $\pi@$, 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. In other words, the encoding function $[\![\cdot]\!]^{\alpha}$ from BioAmbients processes to $\pi@$ processes which we are now ready to formalise requires the (bound) names a and pa, representing the immediate ambient and the parent ambient of each BioAmbients process respectively. This is in accordance with the modelling of cross-compartment objects in $\pi@$ (like ion channels) previously shown: in fact, every BioAmbients process may likely interact with some other process placed in some child or parent or sibling ambient, that is every BioAmbients process is a potential cross-compartment element.

The similarity between the basic π-calculus operators and the corresponding operators inherited by BioAmbients allows us to encode some of them homomorphically

in $\pi@$. This is the case of parallel composition and restriction:

$$\begin{aligned} [\![P \mid Q]\!]^{\alpha}_{a,pa} &\triangleq [\![P]\!]^{\alpha}_{a,pa} \mid [\![Q]\!]^{\alpha}_{a,pa} \\ [\![(new\ n)P]\!]^{\alpha}_{a,pa} &\triangleq (\nu\ n)[\![P]\!]^{\alpha}_{a,pa} \end{aligned}$$

According to the previous considerations, the encoding function is decorated with two names representing the ambient a where the encoded process resides and the outer ambient pa. The name a of the ambient can be used for *local* communication directions:

$$\begin{aligned} [\![local\ n!\{m\}.P]\!]^{\alpha*}_{a,pa} &\triangleq \overline{local@n@a}\langle m\rangle.([\![P]\!]^{\alpha}_{a,pa}) \\ [\![local\ n?\{m\}.P]\!]^{\alpha*}_{a,pa} &\triangleq local@n@a(m).([\![P]\!]^{\alpha}_{a,pa}) \end{aligned}$$

The name pa of the parent ambient can be exploited for any $s2s, c2p/p2c$ or ambient capability ($merge$, $exit/expel$, $enter/accept$). For example, the $s2s$ communication can be encoded in the following way:

$$\begin{aligned} [\![s2s\ n!\{m\}.P]\!]^{\alpha*}_{a,pa} &\triangleq \overline{s2s@n@pa}\langle m\rangle.([\![P]\!]^{\alpha}_{a,pa}) \\ [\![s2s\ n?\{m\}.P]\!]^{\alpha*}_{a,pa} &\triangleq s2s@n@pa(m).([\![P]\!]^{\alpha}_{a,pa}) \end{aligned}$$

This encoding explains *where* such sibling-to-sibling communication is reasonably happening, that is inside the only compartment known by both processes: their parent ambient. Nevertheless, the effect of the communication is limited to the substitution of a name in the scope of the receiving process $[\![P]\!]^{\alpha}_{a,pa}$. The possibility of exploiting an unbounded number of names (three in this case) for each $\pi@$ channel allows us to model easily the "triple matching" typical of BioAmbients actions: in fact, in addition to their "proximity" (i.e. the localisation in the same ambient or parent ambient), the interaction between two processes can happen only if both the direction ($local$, $c2p/p2c$ or $s2s$) and the name n match.

The same considerations hold for $c2p/p2c$ communication:

$$\begin{aligned} [\![p2c\ n!\{m\}.P]\!]^{\alpha*}_{a,pa} &\triangleq \overline{p2c@n@a}\langle m\rangle.([\![P]\!]^{\alpha}_{a,pa}) \\ [\![c2p\ n?\{m\}.P]\!]^{\alpha*}_{a,pa} &\triangleq p2c@n@pa(m).([\![P]\!]^{\alpha}_{a,pa}) \end{aligned}$$

While two processes able to perform a *local* or $s2s$ communication lie in a symmetrical position, the $c2p/p2c$ operation introduces asymmetry in the system, since one process must be located in the parent ambient of the other. This is reflected by the compartment names used in the $\pi@$ encoding: the outer process uses its ambient name a to communicate, while the inner process its parent ambient name pa. There would be no way to make such an operation happen inside the child ambient, since the outer $\pi@$ process does not know (and *must* not know, in agreement with the abstraction of compartment scoping previously discussed) any name associated with the inner ambients.

The correspondence between ambient and parent ambient names of nested compartments is stated by the encoding of the compartment operator $[\cdot]$:

$$[\![\ [\ P\ ]\ ]\!]^{\alpha}_{a,pa} \quad \triangleq \quad (\nu\ c)[\![\ P\ ]\!]^{\alpha}_{c,a}$$

At first sight it corresponds to the encoding of the restriction operator of BioAmbients. The substantial difference can be devised in the names appearing as parameters of the encoding function in the right-hand side of the expression: the restricted name c represents the new ambient of the process P, while a represents both the ambient of $[\ P\ ]$ and the parent ambient of P.

So far, all the problems deriving from the introduction of compartments have been solved by the only use of polyadic synchronisation. The atomicity-related problems noticed during the formalisation of compartments in the π-calculus would emerge now in the attempt of encoding BioAmbients capabilities in $\pi@$. Consider for example the simple *merge* capability:

$$\begin{aligned}&[merge+\,.P \mid Q_1 \mid \cdots \mid Q_n] \mid [merge-\,.R \mid S_1 \mid \cdots \mid S_m] \quad \rightarrow \\ &[P \mid Q_1 \mid \cdots \mid Q_n \mid R \mid S_1 \mid \cdots \mid S_m]\end{aligned} \tag{5.1}$$

The encoding in $\pi@$ of the above system before the reduction would require two distinct restricted names corresponding to each of the ambients ready to be merged. After their merging, only one of such names must be present. This means that either $n+1$ or $m+1$ $\pi@$ processes must have replaced the name corresponding

to their current ambient. Furthermore, BioAmbients semantics requires that all of these $n+1$ or $m+1$ changes in the structure of the system happen instantaneously, that is no other communication or capability can be executed meanwhile. The only way to grant this constraint in the standard π-calculus (without introducing some divergent behaviour) would be to put as guard some *centralised monitor process* which enables the occurrence of one operation at a time, with the consequence of excluding any concurrent feature from the encoded system.

The presence of priority in $\pi@$ instead easily overcomes such kind of issues without any explicit centralised mechanism. The occurrence of the above *merge* synchronisation can be followed by a *sequence of high priority synchronisations* which notify all the involved processes of their new ambient (or parent ambient) name. This sequence of high priority operation is actually a *loop* which must terminate when all the addressed processes have received the desired data. A remarkable feature of such loop is that for any compartment-related operation of BioAmbients (but also of Brane) *all these processes reside in the same compartment*. This peculiarity allows us to express the loop in a very general way:

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

The $BCAST$ process implements a loop triggered by the receiving of three names: the first two names are used to identify the channel $x@y$ over which the communication loop will be executed. The third name is a datum which is sent to all the processes listening on $x@y$, representing the name of some ambient where some group of processes is going to move. The use of two levels of (high) priority (denoted by single or double-underlined actions, corresponding to integer levels 1 and 0 respectively) provides two very important properties:

- once triggered, the loop will execute entirely before the occurrence of any other encoded BioAmbients operation, since all such operations – both communications, as we have shown, and capabilities, as we will show – are guarded by an initial *low priority* (integer level 2 of priority, corresponding to no underline) action;

- the loop will end without leaving any "garbage" which may interfere with the system later on.

The first property is granted by the high priority of the replicated *bcast* input. After the communication of the datum z over the channel $x@y$, the process spawns another copy of itself by means of the same expedient previously showed for the encoding of recursive behaviour by exploitation of replication. The presence of the τ action with an intermediate prioritised level allows the termination of the loop when all the processes listening on $x@y$ have actually received the datum z, so that the remainder of the $BCAST$ process disappears completely *before* any other communication or capability may be performed.

Consequently, the implementation in $\pi@$ of the *merge* capability requires the triggering of the $BCAST$ loop with the correct parameters. First, we need to understand *where* two processes performing a $merge+/merge-$ capability are located: by checking the semantic rules of Def. 2.7 we can see that such processes must be located in sibling ambients (ambients that are children of the same outer ambient), i.e. the corresponding synchronisation in $\pi@$ happens in the parent ambient, exactly as showed for the $s2s$ communication. Second, the communication must carry on the name associated with one of the two ambients, so that the receiving process can use it as new compartment name for all of its subsequent synchronisations and forward it to all the other processes affected by the structural change in the ambient tree. Since the merging of two ambients is a symmetrical operation, the choice of the name to be communicated is arbitrary. Here we choose to keep the name associated with the ambient of the process exhibiting the $merge+$ capability:

$$[\![\, merge+\ n.P \,]\!]^{\alpha*}_{a,pa} \quad \triangleq \quad \overline{merge@n@pa}\langle a\rangle.([\![\, P \,]\!]^{\alpha}_{a,pa})$$

In agreeement with the above considerations, the encoding of $merge+$ is located in the parent ambient pa and communicates the name a of its ambient to the process ready for the complementary $merge-$:

$$[\![\, merge-\ n.P \,]\!]^{\alpha*}_{a,pa} \quad \triangleq \quad merge@n@pa(x).\ \overline{\underline{bcast}}\langle merge, a, x\rangle.([\![\, P \,]\!]^{\alpha}_{x,pa})$$

The *merge*– operation implies the receiving of some name to be substituted for the placeholder x which will be used as new ambient in $(\llbracket\ P\ \rrbracket^{\alpha}_{x,pa})$, while the parent ambient pa remains unchanged exactly like in the original BioAmbients system. The subterm $\overline{\underline{\underline{bcast}}}\langle merge, a, x\rangle$ triggers the $BCAST$ loop which in turn notifies all the sibling processes (represented by the siblings $S_1, \ldots, S_m$ of R in Expr. (5.1)) of their new ambient x (which, again, at this time has been replaced by the name of the ambient sent by the process performing the complementary *merge*+ capability). The name *merge* used to denote the channels $merge@n@a$ and $merge@n@pa$ is completely arbitrary and allows us to distinguish this broadcast-like communication occurring after the merging of compartments from the other broadcast-like loops that are triggered by the translation of the other BioAmbients capabilities.

Similar considerations lead to the encoding of the *enter*/*accept* and *exit*/*expel* capabilities. Their partial asymmetry requires to bring attention to the names transmitted during the reduction and the way they are used afterwards.

The *enter*/*accept* reduction is triggered by two processes whose localisation is symmetrical with respect to the global tree structure of the system: they are inside ambients children of the same parent ambient, exactly like in the case of the *merge* operation. After the reduction, however, their situation is asymmetrical since their respective ambients are one the child of the other. Consequently, the process performing the *accept* capability is not going to change location but must communicate the name corresponding to its ambient to the process ready for the *enter* capability, which will use such name to denote its new parent ambient:

$$\begin{aligned}
\llbracket\ accept\ n.P\ \rrbracket^{\alpha *}_{a,pa} &\triangleq \overline{enter@n@pa}\langle a\rangle.(\llbracket\ P\ \rrbracket^{\alpha}_{a,pa}) \\
\llbracket\ enter\ n.P\ \rrbracket^{\alpha *}_{a,pa} &\triangleq enter@n@pa(x).\overline{\underline{\underline{bcast}}}\langle pa, a, x\rangle.(\llbracket\ P\ \rrbracket^{\alpha}_{a,x})
\end{aligned}$$

As for the *merge*– operation, the encoding of the *enter* capability triggers the $BCAST$ loop which notifies all the involved processes of their new parent ambient.

The *exit*/*expel* reduction is the converse of the previous one: the two processes performing it lie in an asymmetrical position and end symmetrically distributed with respect to the ambient tree. In this case the process exhibiting the *exit* capability

and all of its siblings must be notified of their new parent ambient, represented by the name of the parent ambient of the process performing the complementary *expel*:

$$\begin{aligned} [\![\ expel\ n.P\ ]\!]^{\alpha*}_{a,pa} &\triangleq \overline{expel@n@a}\langle pa\rangle.([\![\ P\ ]\!]^{\alpha}_{a,pa}) \\ [\![\ exit\ n.P\ ]\!]^{\alpha*}_{a,pa} &\triangleq expel@n@pa(x).\overline{\underline{bcast}}\langle pa, a, x\rangle.([\![\ P\ ]\!]^{\alpha}_{a,x}) \end{aligned}$$

The *enter*/*accept* and *exit*/*expel* capabilities differ from *merge*+ /*merge*– for a subtle particular: in the *merge*– operation the name *merge* was sent as first parameter to $BCAST$, while in the other two the name *pa* was used. We may have used two distinct names $enter, exit$ for indicating the change of the name of the parent compartment in consequence of each of such capabilities, but their effect is actually the same, so they can be condensed in the simple idea of "substitution of parent compartment name". Similar reasoning can be argued for the encoding of Brane actions.

A substantial consequence of the above encodings of BioAmbients capabilities is the silent assumption that *all the encoded processes are always listening on the right channels for possible changes of their ambient or parent ambient names*, independently of the other actions they are ready to perform. Furthermore, *after the receiving of a new ambient or parent ambient name in consequence of the triggering of the $BCAST$ loop, each notified process must return exactly in its previous state*, except for the substitution of the old ambient or parent ambient name with the new one just received. As an example, consider again the situation of Expr. (5.1): the $\pi@$ processes corresponding to $S_1, \ldots, S_m$ are going to change their ambient name after the $BCAST$ loop triggered by P and R. This means that each of those m processes is listening on some channel $merge@a$, with a representing their current ambient name. More exactly, each of them is listening simultaneously on

- $merge@a$ in order to be notified of the merging of their ambient,
- $merge@pa$ for the merging of their parent ambient, and
- $pa@a$ for the change of the name of their parent ambient as a consequence of an *exit* or an *accept* operation.

In order to denote concisely this property, we may say that the $\pi@$ encoding of each BioAmbients communication or capability is put in nondeterministic choice with the three options expressed by the following process:

$$TREE(b,a,pa) \triangleq \underline{\underline{pa@a}}(x).\overline{\underline{b}}\langle a,x\rangle + \underline{\underline{merge@pa}}(x).\overline{\underline{b}}\langle a,x\rangle + \underline{\underline{merge@a}}(x).\overline{\underline{b}}\langle x,pa\rangle \tag{5.2}$$

The three expressions with shape $\overline{\underline{b}}\langle a,x\rangle$ or $\overline{\underline{b}}\langle x,pa\rangle$, as we will see later on, spawn another copy of the original process which will use the first received name as ambient and the second one as parent ambient. It is worth noticing the use of the received name x after each choice branch:

- the name received over $pa@a$ is used as new parent ambient name, as required by the *exit* and *enter* capabilities;
- the name received over $merge@a$ is used as new ambient name, when the merging operation affects the local ambient;
- the name received over $merge@pa$ is used as new parent ambient name, when the merging operation affects the parent ambient.

After the execution of the $BCAST$ loop, all the notified processes return to their previous state, that is they are ready again to behave as $S_1,\ldots,S_m$ in the case of Expr. (5.1), even if they are located in a new compartment. In other words, *the encoding of each communication or capability is represented by a loop which allows the process to return in its previous state after receiving some new ambient or parent ambient name*. With a slight abuse of notation we may then write that each communication or capability choice of BioAmbients can be encoded as follows:

$$\begin{aligned}&[\![\textstyle\sum_{i\in I,\, I\neq\emptyset} \xi_i.P_i]\!]^{\alpha}_{a,pa} \triangleq \\ &\quad (\nu\ s)(\overline{\underline{s}}\langle a,pa\rangle \mid\ !\ \underline{s}(na,npa).SUBSUM(s,na,npa))\end{aligned} \tag{5.3}$$

where $SUBSUM$ is just a shorthand for the following expression (since its definition does not depend only on the names s, na, npa but also on the branches of the choice $\sum_i \xi_i.P_i$):

$$SUBSUM(s, na, npa) \quad \triangleq \quad \sum\nolimits_{i\in I,\, I\neq\emptyset} [\![\, \xi_i.P_i \,]\!]^{\alpha*}_{na,npa} \;+\; TREE(s, na, npa) \tag{5.4}$$

Here, replication is exploited as usual for the modelling of recursive behaviour, which is needed in order to make each process return to the original state after the interaction with some triggered $BCAST$ loop. The presence of the $TREE$ process as part of the choice grants that each encoded process is listening on the right channels and ready to "be passively moved" inside some new ambient or parent ambient in consequence of the structural change in the nesting tree that has been triggered by another pair of processes. $\sum \xi_i.P_i$ represents either a choice between capabilities or between communications, since they are kept distinct in the definition of BioAmbients grammar. In the particular case of a single-branched choice in the form $\pi.P$ or $M.P$ the encoding is the same even if the choice operator is not written explicitly, because we still need to preserve the implicit choice branches with recursive behaviour expressed by $TREE$. The function $[\![\, \cdot \,]\!]^{\alpha*}$ decorated with an additional star denotes the encoding of each communication or capability, in order to distinguish it from the encoding of single-branched choices.

This $\pi@$ encoding of the choice operator determines severe consequences on the expression of BioAmbients replication. If we consider the following example

$$\begin{aligned} &[\,!\, merge{+}\, n.P \;\mid\; Q] \;\mid\; [merge{-}\, .R \;\mid\; S\,] \qquad \rightarrow \\ &[\,!\, merge{+}\, n.P \;\mid\; P \;\mid\; Q \;\mid\; R \;\mid\; S\,] \end{aligned} \tag{5.5}$$

then we may suppose to encode the replication homomorphically:

$$[\![\,!\, P \,]\!]^{\alpha}_{a,pa} \quad \triangleq \quad !\, [\![\, P \,]\!]^{\alpha}_{a,pa} \tag{5.6}$$

Unfortunately this would cause a high priority loop without termination in Expr. (5.3). In fact, each encoded process $[\![\, \sum \xi_i.P_i \,]\!]^{\alpha}_{a,pa}$ undergoes an internal, high

priority reduction on the private channel s which spawns a new copy of a subprocess $SUBSUM(s, a, pa)$. Consequently, a direct replication of Expr. (5.3) by Expr. (5.6) would cause an unbounded number of such reductions to happen immediately. Their high priority level would hang the entire system. It would be possible to overcome the problem by correcting the encoding of the choice in the following way:

$$\begin{array}{l} [\![\sum_{i\in I,\, I\neq\emptyset} \xi_i.P_i]\!]^{\alpha}_{a,pa} \quad \triangleq \\ \qquad (\nu\ s)(SUBSUM(s,a,pa) \mid !\ \underline{s}(na,npa).SUBSUM(s,na,npa)) \end{array} \tag{5.7}$$

with $SUBSUM$ corresponding to Expr. (5.4). The internal high priority reduction is eliminated because the spawned term $SUBSUM(s, a, pa)$ is now explicitly written in the encoding. The replication of Expr. (5.7) by Expr. (5.6) now can successfully translate the BioAmbients system of Expr. (5.5). In fact, after the first reduction on $merge@n@pa$, a copy of $[\![P]\!]^{\alpha}$ is spawned while the original $[\![!\ merge+\ n.P]\!]^{\alpha}$ is kept and ready to execute another $merge+$ reduction, in accordance with the right-hand side of Expr. (5.5).

However, Expr. (5.6) is far from being correct. In fact, if we consider the following system

$$\begin{array}{l} [merge+\ .P \mid Q] \mid [merge-\ .R \mid !\ S] \quad \rightarrow \\ [P \mid Q \mid R \mid !\ S] \end{array} \tag{5.8}$$

where the replication acts on the process S which is sibling of the process undergoing the $merge-$ reduction, we may immediately notice the presence of another high priority non-terminating loop, even without unfolding the encoding of the whole system. We just need to recall the meaning of "replication of S" as "an unbounded number of copies of S":

$$!\ S \quad \equiv \quad !\ S \mid S \quad \equiv \quad !\ S \mid S \mid S \quad \equiv \quad !\ S \mid S \mid S \mid S \mid \cdots$$

The encoding of replication of Expr. (5.6) would require *each of the copies of* $[\![S]\!]^{\alpha}$ *represented by* $!\ [\![S]\!]^{\alpha}$ *to be notified of the change of compartment name* triggered

by the *merge* operation. Since such number of copies is unbounded and the loop is characterised by high priority, the system would hang immediately.

Consequently, we are forced to introduce an indirect encoding of BioAmbients replication by changing the intuitive abstraction of the replication operator itself. Even if $!\,P$ represents an unlimited number of copies of P, there is no need to unfold all of such copies. More precisely, we may think that there is no need to unfold more than just *one* copy of P at each time. In other words, we may represent $!\,P$ as $!\,P \mid P$ and consider to unfold a new copy of P *only* when its previous copy undergoes some reduction $P \rightarrow P'$. Actually this is equivalent to keeping each replication in a sort of *normal form* where each replicated process is exactly unfolded once. Supposing that $P \rightarrow P'$, the reduction

$$!\,P \quad \rightarrow \quad !\,P \mid P'$$

would be then written as

$$!\,P \mid P \quad \rightarrow \quad !\,P \mid P \mid P'$$

where the copy of P in the right-hand side of the expression is unfolded from $!\,P$ only after the reduction of the first copy of P in the left-hand side to P'.

This behaviour can be straightforwardly obtained in $\pi@$ by encoding each BioAmbients replication as a loop where one copy of $[\![\, P \,]\!]^{\alpha}$ is always unfolded and undergoes the corresponding reductions of the BioAmbients process P. During any such reductions, $[\![\, P \,]\!]^{\alpha}$ causes another copy of itself to spawn in order to grant that the semantics of $!\,P$ is preserved. Furthermore, the encoding of $!\,P$ must observe the same migration rules expressed by the $TREE$ process of Expr. (5.2). In order to allow $[\![\, P \,]\!]^{\alpha}$ to spawn a new copy of itself whenever it undergoes some reduction we must introduce a new parameter in the encoding function, which represents the private channel over which the spawning event will be communicated: this parameter is needed because the encoding of the action which causes P to reduce may be several nested calls later with respect to the recursive definition of $[\![\, !\,P \,]\!]^{\alpha}$. In fact, consider the following system:

$$S \quad \triangleq \quad !\,(\mathit{local}\ n?.Q_1' \mid Q_2)$$

Its encoding in π@ requires three recursive calls of the encoding function $[\![\cdot]\!]^{\alpha}$, one for each of the following subsystems:

$$S \triangleq !\, P \qquad P \triangleq Q_1 \mid Q_2 \qquad Q_1 \triangleq \mathit{local}\ n?.Q_1'$$

By the previous considerations, we can keep one copy of P always unfolded and write S in the following way:

$$S \triangleq !\, P \mid P$$

After the reduction of Q_1 to Q_1', the system should appear as

$$S \mid \mathit{local}\ n!.\mathbf{0} \rightarrow !\, P \mid P \mid Q_1' \mid Q_2$$

where one copy of P is still unfolded. The unfolding of the corresponding π@ process $[\![P]\!]^{\alpha}$ must be triggered by the *local* n communication of $[\![Q_1]\!]^{\alpha}$. In other words, the π@ process $[\![Q_1]\!]^{\alpha}$ must cause $[\![!\, P]\!]^{\alpha}$ to spawn a new copy of $[\![P]\!]^{\alpha}$, even if $[\![!\, P]\!]^{\alpha}$ is defined recursively as a function of $[\![P]\!]^{\alpha}$, in turn defined as a function of $[\![Q_1]\!]^{\alpha}$. The only way to achieve this result is by introducing some new name k passed as parameter from $[\![!\, P]\!]^{\alpha}$ until the recursive call of the encoding function $[\![Q_1]\!]^{\alpha}$, which will use it to spawn a new copy of $[\![P]\!]^{\alpha}$ as soon as the *local* n operation is executed. Hence, the encoding of the *local* communication (and of all the other communications and capabilities) should be modified as follows:

$$[\![\mathit{local}\ n?\{m\}.P]\!]^{\alpha}_{k,a,pa} \triangleq \mathit{local}@n@a(m).([\![P]\!]^{\alpha}_{a,pa} \mid \overline{\underline{unfold@k}}) \tag{5.9}$$

$\overline{\underline{unfold@k}}$ spawns a new copy of $[\![P]\!]^{\alpha}$ in the encoding of $!\, P$:

$$\begin{aligned}
[\![!\, P]\!]^{\alpha}_{a,pa} &\triangleq (\nu\, k)(BANG(k,a,pa) \mid [\![P]\!]^{\alpha}_{k,a,pa} \mid \\
&\quad\ !\, \underline{new@k}(na,npa).[\![P]\!]^{\alpha}_{k,na,npa}) \\
BANG(k,a,pa) &\triangleq !\, \underline{k}(na,npa).SUBBANG(k,na,npa) \\
&\quad \mid SUBBANG(k,a,pa) \\
SUBBANG(k,na,npa) &\triangleq \underline{unfold@k}.\overline{\underline{new@k}}\langle na,npa\rangle.\overline{\underline{k}}\langle na,npa\rangle + \\
&\quad\ TREE(k,na,npa)
\end{aligned} \tag{5.10}$$

The recursive call $[\![P]\!]^{\alpha}_{k,na,npa}$, together with the new encodings suggested by Expr.(5.9), ensures that each communication or capability appearing after the above replication will spawn a new copy of $[\![P]\!]^{\alpha}$ with a high priority reduction on the channel $unfold@k$. Such spawning is mediated by the $BANG$ subprocess, which regulates also the relocation of $[\![! P]\!]^{\alpha}_{a,pa}$ by direct embedding of the $TREE$ subprocess discussed previously. It is worth remarking that such relocation in practice affects only the subprocess $BANG$, which is the only process regulating *where* (in terms of ambient and parent ambient) the following copies of $[\![P]\!]^{\alpha}$ will be spawned. In accordance with the previous considerations, one copy (at least one, more precisely) of $[\![P]\!]^{\alpha}$ is always kept unfolded. The relocation of this spawned copy into some new ambient or parent ambient after some *merge* or *exit/expel* or *enter/accept* capability happens transparently, since it already embeds in the correct way the subprocess $TREE$ as part of each encoded choice.

Two remarks allow us to refine Expr. (5.10), which is still not correct.

First, replications may be *nested* and more than one unfolding may be needed after some reduction. As an example, consider the following system

$$S \triangleq ! (\nu\ x)!\ local\ n?.R'$$

which we rewrite in terms of some additional shorthands:

$$S \triangleq ! P \qquad P \triangleq (\nu\ x)Q \qquad Q \triangleq ! R \qquad R \triangleq local\ n?.R'$$

The unfolding of one copy of each replicated process leads to the following expression:

$$S \quad \equiv \quad ! P \mid (\nu\ x)(! R \mid R)$$

The reduction of R to R' requires *two unfoldings*: one of P and one of R, corresponding to *one unfolding for each replication appearing before the communication or capability* (i.e. each replication appearing as ancestor not followed by choice in the syntactic tree generating the BioAmbients expression). Consequently, one name k as additional parameter in the corresponding $\pi@$ encoding is not enough if more than one nested replication is present in the BioAmbients process. Hence, the parameter k must be replaced by the set $K = \{k_1, k_2, \ldots, k_n\}$ containing one name

for each encoded replication which must be unfolded. This set is added with a new name after the encoding of each replication and becomes empty after the encoding of any BioAmbients choice, since any subsequent reduction of R' does not affect the replication of P and R anymore.

Second, the constant unfolding of one copy of a replicated process is not sufficient to express its full behaviour. Consider for example the following system:

$$S \triangleq !\,P \qquad P \triangleq (local\ n?.P_1 + local\ n!.P_2)$$

If we unfold only one copy of P, we miss the reduction that may happen between two distinct copies of P themselves. In fact we have that

$$S \equiv !\,P \mid P \mid P \quad \rightarrow \quad !\,P \mid P_1 \mid P_2$$

where P_1 and P_2 follow the reduction over *local* n. Consequently, at least *two* copies of $[\![\ P\ ]\!]^{\alpha}$ must be constantly kept unfolded in the corresponding encoded process.

Expr. (5.10) can be then modified as follows:

$$\begin{aligned}[\![\ !\,P\ ]\!]^{\alpha}_{K,a,pa} \triangleq\ & (\nu\ k)(BANG(k,a,pa) \mid \\ & [\![\ P\ ]\!]^{\alpha}_{K\cup\{k\},a,pa} \mid [\![\ P\ ]\!]^{\alpha}_{K\cup\{k\},a,pa} \mid \\ & !\,\underline{\underline{new@k}}(na,npa).[\![\ P\ ]\!]^{\alpha}_{K\cup\{k\},na,npa})\end{aligned}$$

The encoding of communications and capabilities is updated accordingly:

$$\begin{aligned}[\![\ local\ n?\{m\}.P\ ]\!]^{\alpha}_{K,a,pa} \triangleq\ & local@n@a(m).\Big([\![\ P\ ]\!]^{\alpha}_{\emptyset,a,pa} \mid \\ & \overline{\underline{\underline{unfold@k_1}}} \mid \cdots \mid \overline{\underline{\underline{unfold@k_n}}}\Big)\end{aligned}$$

with $K = \{k_1, \ldots, k_n\}$.

The full definition of $[\![\ \cdot\ ]\!]^{\alpha}$ is given in Table 5.1 and Table 5.2. Here, the names *oa* and *opa* represent two fictitious names needed for the correct initialisation of the encoding function, and corresponding to the outermost ambient and parent ambient of the entire encoded system.

The encoding function $[\![\ \cdot\ ]\!]^{\alpha}$ enjoys the requirements discussed in section 5.1.2, as stated by the following theorem.

Theorem 5.1 $[\![\cdot]\!]^{\alpha}$ *is a suitable encoding (modulo structural congruence), that is: let* P, P_1, P_2 *be BioAmbients processes, let* Q *be a* π@ *process, then*

1. $[\![P_1 \mid P_2]\!]^{\alpha} = [\![P_1]\!]^{\alpha} \mid [\![P_2]\!]^{\alpha}$;
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* $\exists P_1 : P \rightarrow^* P_1 \wedge Q \rightarrow^* [\![P_1]\!]^{\alpha}$.

5.1.4 Encoding Brane in π@

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). In fact, branes represent the boundaries of compartments: each new membrane corresponds to the definition of a new compartment, i.e. the name of the immediate compartment where the associated π@ process is located. Consequently, the function $[\![\cdot]\!]^{\beta}$ from Brane to π@ has two formal parameters when applied to the parallel composition of systems

$$[\![P \circ Q]\!]^{\beta}_{K,pc} \triangleq [\![P]\!]^{\beta}_{K,pc} \mid [\![Q]\!]^{\beta}_{K,pc}$$

and three for the parallel composition of branes:

$$[\![\sigma \mid \rho]\!]^{\beta}_{K,c,pc} \triangleq [\![\sigma]\!]^{\beta}_{K,c,pc} \mid [\![\rho]\!]^{\beta}_{K,c,pc}$$

Their encoding is almost the same as the encoding of parallel composition of BioAmbients processes. The names c and pc play the same role of a and pa for the encoding function $[\![\cdot]\!]^{\alpha}$. The appearance of the additional name c as a parameter of the encoding function occurs after the first application of $[\![\cdot]\!]^{\beta}$ to branes:

$$[\![\sigma(\!| P |\!)]\!]^{\beta}_{K,pc} \quad \triangleq \quad (\nu\, c)([\![\sigma]\!]^{\beta}_{K,c,pc} \mid [\![P]\!]^{\beta}_{K,c})$$

The encoding of a membrane corresponds to the introduction of the new name c that is used as compartment for branes, and as parent compartment for the inner system P. Any encoded brane occurring in $[\![P]\!]^{\beta}_{K,c}$ will use the above name c as parent compartment and will be able to interact over it with the surrounding branes represented here by $[\![\sigma]\!]^{\beta}_{K,c,pc}$. The name pc of the parent compartment is known only by the branes placed in the outer membrane, in accordance with the same intuition of compartment nesting exploited for the encoding of ambients.

The basic actions $exo/exo^{\perp}$, $phago/phago^{\perp}$ and $pino$ are encoded almost like BioAmbients capabilities: 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.

For example, the translation of the $exo/exo^{\perp}$ actions requires three names for each channel and triggers the $BCAST$ loop whose definition is exactly the same as for $[\![\ ]\!]^{\alpha}$:

$$[\![exo_n.\sigma]\!]^{\beta *}_{K,c,pc} \quad \triangleq \quad exo@n@pc(x).\underline{\overline{bcast}}\langle exo, c, x\rangle.([\![\sigma]\!]^{\beta}_{\emptyset,pc,x} \mid \Pi_K) \qquad (5.11)$$

$$[\![exo^{\perp}_n.\sigma]\!]^{\beta *}_{K,c,pc} \quad \triangleq \quad \overline{exo@n@c}\langle pc\rangle.([\![\sigma]\!]^{\beta}_{\emptyset,c,pc} \mid \Pi_K) \qquad (5.12)$$

The asymmetry of the situation recalls the encoding of the $exit/expel$ capability, since the internal process listens on the channel $exo@n@pc$ intuitively located in the parent compartment pc, while the external process is ready to send on the local channel $exo@n@c$. The main difference is in the choice and use of the transmitted name pc by the $TREE$ subprocess and in the compartment where the broadcast effect of $BCAST$ occurs. In order to understand the rearrangement of the membrane

tree structure, consider the following example:

$$S \triangleq \underbrace{\cdots \circ \overbrace{\sigma(\!| \underbrace{\tau(\!| P |\!)}_{c_3} |\!)}^{c_2}}_{c_1} \qquad \sigma \triangleq exo^{\perp}.\sigma'|\rho \qquad \tau \triangleq exo.\rho'|\gamma$$

c_1 represents the parent compartment of the whole system S, while c_2 is both the compartment of σ and the parent compartment of τ (and of the entire subsystem $\tau(\!| P |\!)$). The system S can reduce as follows:

$$S \quad \rightarrow \quad S' \qquad\qquad S' \triangleq \underbrace{\cdots \circ P \circ \overbrace{\sigma'|\rho|\rho'|\gamma(\!| \cdot |\!)}^{c_2}}_{c_1}$$

Two kinds of structural changes can be noticed in the reduction from S to S'. First, the parent compartment of P after the reduction coincides with the parent compartment c_1 of S: this justifies the name pc transmitted by the encoded $exo^{\perp}$ process. Second, the branes ρ' and γ changed both their compartment and their parent compartment. In fact, before the reduction, they were respectively c_3 and c_2, while after the reduction they become c_2 and c_1. This means that the corresponding $\pi@$ processes must replace their previous compartment name c_3 by their parent compartment c_2, and use as new parent compartment the name c_1 sent by the process performing the $[\![\, exo^{\perp} \,]\!]^{\beta}$ action. The recursive call $[\![\, \sigma \,]\!]^{\beta}_{\emptyset,pc,x}$ of Expr. (5.11) reflects this behaviour, as well as the definition of the $TREE$ process:

$$\begin{aligned} TREE(b,c,pc) \quad \triangleq \quad & \underline{\underline{exo@pc}}(x).\underline{\underline{b}}\langle c,x\rangle + \\ & \underline{\underline{exo@c}}(x).\underline{\underline{b}}\langle pc,x\rangle + \\ & \underline{\underline{pc@c}}(x).\underline{\underline{b}}\langle c,x\rangle \end{aligned}$$

The first branch of the choice is followed, in the previous example, by the subsystem P which receives the name of its new parent compartment. The second branch is followed by the brane γ which receives the name of its new parent compartment but replaces also the name of its immediate compartment by the name of the previous parent compartment.

The third branch is needed for the encoding of the $phago/phago^{\perp}$ reduction:

$$\begin{aligned}
[\![\, phago_n.\sigma \,]\!]^{\beta*}_{K,c,pc} &\triangleq phago@n@pc(x).\overline{\underline{bcast}}\langle pc, c, x\rangle.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,x} \mid \Pi_K) \\
[\![\, phago^{\perp}_n(\rho).\sigma \,]\!]^{\beta*}_{K,c,pc} &\triangleq (\nu\ x)\big(\overline{phago@n@pc}\langle x\rangle. \\
&\qquad ([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,pc} \mid [\![\, \rho \,]\!]^{\beta}_{\emptyset,x,c} \mid \Pi_K)\big)
\end{aligned}$$

In this case, the name transmitted by the $[\![\, phago^{\perp} \,]\!]^{\beta}$ process corresponds to the newly created membrane that surrounds the engulfed process and is used as new parent compartment.

The *pino* action has a similar, but simpler encoding:

$$[\![\, pino(\rho).\sigma \,]\!]^{\beta*}_{K,c,pc} \triangleq (\nu\ x)\tau.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,pc} \mid [\![\, \rho \,]\!]^{\beta}_{\emptyset,x,c} \mid \Pi_K)$$

In the right-hand side of the encoding, the reduction τ represents an invisible $\pi@$ transition, and must not be confused with the notation used for branes. The lack of complementary action and the very localised effect allows us to disregard any broadcast loop. The new created membrane is represented by a new restricted name x used as local compartment by $[\![\, \rho \,]\!]^{\beta}_{\emptyset,x,c}$, whose parent compartment coincides with the immediate compartment of the encoded process $[\![\, pino(\rho).\sigma \,]\!]^{\beta}_{K,c,pc}$.

The sequential composition of branes $\sigma.\sigma'$ is encoded exactly like the choice operator of BioAmbients, as a sort of single-branch choice:

$$\begin{aligned}
[\![\, a.\sigma \,]\!]^{\beta}_{K,c,pc} \triangleq\ & BCAST \mid \nu\ s(!\ \underline{\underline{s}}(nc, npc). \\
& ([\![\, a.\sigma \,]\!]^{\beta*}_{K,nc,npc} + TREE(s, nc, npc)) \mid \\
& [\![\, a.\sigma \,]\!]^{\beta*}_{K,c,pc} + TREE(s, c, pc))
\end{aligned}$$

The presence of two distinct replication operators in Brane leads to two slightly different encodings which reflect the fact that systems are only provided with parent compartment, while branes are also aware of their immediate compartment. The encoding of branes replication is the same as the one seen for BioAmbients replica-

tion:

$$\begin{aligned}
[\![\,!\,\sigma\,]\!]^{\beta}_{K,c,pc} &\triangleq (\nu\, b)(BANG(b,c,pc) \mid \\
&\quad [\![\,\sigma\,]\!]^{\beta}_{K\cup\{b\},c,pc} \mid [\![\,\sigma\,]\!]^{\beta}_{K\cup\{b\},c,pc} \mid \\
&\quad !\,\underline{new@b}(nc,npc).[\![\,\sigma\,]\!]^{\beta}_{K\cup\{b\},nc,npc}) \\
BANG(b,c,pc) &\triangleq !\,\underline{b}(nc,npc).SUBBANG(b,nc,npc) \mid \\
&\quad SUBBANG(b,c,pc) \\
SUBBANG(b,nc,npc) &\triangleq \underline{unfold@b}.\overline{\underline{new@b}}\langle nc,npc\rangle.\overline{\underline{b}}\langle nc,npc\rangle + \\
&\quad TREE(b,nc,npc)
\end{aligned}$$

The encoding of systems replication is simplified by the absence of name for the immediate compartment, which allow us to shrink the definition in the following way:

$$\begin{aligned}
[\![\,!\,P\,]\!]^{\beta}_{K,pc} &\triangleq (\nu\, b)(BANG'(b,pc) \mid \\
&\quad [\![\,P\,]\!]^{\beta}_{K\cup\{b\},pc} \mid [\![\,P\,]\!]^{\beta}_{K\cup\{b\},pc} \mid \\
&\quad !\,\underline{new@b}(npc).[\![\,P\,]\!]^{\beta}_{K\cup\{b\},npc} \\
BANG'(b,npc) &\triangleq !\,\underline{b}(npc).SUBBANG'(b,npc) \mid \\
&\quad SUBBANG'(b,pc) \\
SUBBANG'(b,npc) &\triangleq \underline{unfold@b}.\overline{\underline{new@b}}\langle npc\rangle.\overline{\underline{b}}\langle npc\rangle + \\
&\quad \underline{exo@npc}(x).\overline{\underline{b}}\langle x\rangle
\end{aligned}$$

Instead of the three branches of the $TREE$ process, only one is present, since only one of those branches (triggered by a reduction over $exo@npc$) is related to changes of the parent compartment name.

The full definition of $[\![\cdot]\!]^{\beta}$ is given in Table (5.3) and Table (5.4). Similarly to the BioAmbients encoding, *oc* and *opc* are placeholders standing for the compartment and parent compartment of the outermost processes. The non-trivial encoding in $\pi@$ of molecular reactions and other extensions proposed in [22] is not strictly related to compartment semantics, but is being considered for future work, due to its relevance in chemical modelling.

Also the encoding function $[\![\cdot]\!]^{\beta}$ enjoys the requirements discussed in section 5.1.2:

Theorem 5.2 *$[\![\cdot]\!]^{\beta}$ is a suitable 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* $\exists P_1 : P \rightarrow^* P_1 \wedge Q \rightarrow^* [\![P_1]\!]^{\beta}$.

5.1.5 Encoding Brane in core-$\pi@$

In the encodings of BioAmbients and Brane previously defined, we used three priority levels and up to three names for each channel in $\pi@$. In order to provide similar encodings in core-$\pi@$, we need now to use not more than two names for channel and two priority levels.

The expedient for reducing the number of names needed for encoding capabilities and actions is very simple: for example, the $exo_n/exo_n^{\perp}$ action can be encoded with the only name exo_n (where the underscore character '_' is used just for sake of clarity and is part of the string representing the name) instead of the two names exo and n joint by polyadic synchronisation as $exo@n$. However, the (unique) consequence of this change in the encoding is the multiplication of the names needed for representing each private name in BioAmbients. In fact, each BioAmbients name n must now be translated as a vector

$$(enter\_n, expel\_n, merge\_n, local\_n, s2s\_n, p2c\_n, c2p\_n)$$

in order to allow each different communication or capability to operate independently. Since Brane lacks restriction and name passing, its encoding in $\pi@$ is not affected by such conversion.

The reduction of the number of levels of priority influences the encoding of both languages. In the definition of the previous encoding functions, one prioritised intermediate level was used to "garbage-collect", by means of a prioritised τ action, the $BCAST$ processes which terminated the loop with broadcast-like effect. Here we recall the definition of $BCAST$:

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

A naive way to reduce the number of priority levels would be to decrease the priority level of both the τ transition (which would become a normal low priority transition) and all the other high priority reductions (which would still be prioritised). The definition of the $BCAST$ process would be consequently modified as follows:

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

The problem with this definition of $BCAST$ is that there is no way to know *when* the above τ transition will be executed and the loop will terminate its action. In principle this would not constitute an important issue, since the high priority of the reductions over $x@y$ and $bcast$ would force the loop to be interrupted when all the processes waiting for a possible broadcast on the corresponding channel have been notified of the new compartment or ambient name. Unfortunately, a spurious process

$$SP \quad \triangleq \quad \overline{\underline{x@y}}\langle z\rangle.\overline{\underline{bcast}}\langle x,y,z\rangle + \tau$$

would still be around, ready to interfere with the normal behaviour of the system over some channel $x@y$. Such interference may happen for example in the encoding of the following Brane system:

$$S \quad \triangleq \quad \overbrace{\underbrace{phago.\sigma|\sigma'(\!|\ P\ |\!)}_{c_2} \circ \underbrace{phago^{\perp}(exo.\gamma).exo^{\perp}.\rho(\!|\ Q\ |\!)}_{c_3}}^{c_1}$$

c_1 represents the name of the parent compartment of the whole system S, c_2 and c_3 are the names corresponding to the immediate compartments of its two subsystems. S undergoes the following reductions:

$$S \quad \rightarrow \quad S' \quad \rightarrow \quad S''$$

with

$$S' \quad \triangleq \quad exo^{\perp}.\rho(\!| \overbrace{exo.\gamma(\!| \underbrace{\sigma|\sigma'(\!| P |\!)}_{c_2} |\!)}^{c_4} \circ Q |\!)$$

and

$$S'' \quad \triangleq \quad \overbrace{\underbrace{\sigma|\sigma'(\!| P |\!)}_{c_2} \circ \rho|\gamma(\!| Q |\!)}^{c_1}$$

After the first reduction, a fresh membrane is represented by the name c_4 which is then lost after the following *exo* operation. The $BCAST$ loop triggered by the encoding of the first reduction would be spawned by the process performing the *phago* action with parameters c_1, c_2, c_4, in order to make all the branes in σ' replace their parent compartment c_1 by the fresh compartment c_4. Such $BCAST$ loop may leave the following spurious process:

$$SP \quad \triangleq \quad \overline{\underline{c_1@c_2}}\langle c_4\rangle.\overline{\underline{bcast}}\langle c_1, c_2, c_4\rangle + \tau$$

The use of three priority levels, with an intermediate prioritised level for τ, would have forced SP to disappear before the transition

$$[\![\, S' \,]\!]^{\beta} \rightarrow [\![\, S'' \,]\!]^{\beta}$$

while with only two levels it may be still present in parallel composition with $[\![\, S'' \,]\!]^{\beta}$. This means that the subprocess $[\![\, \sigma|\sigma'(\!| P |\!) \,]\!]^{\beta}$ of the encoded system S'' may be still subject to the effect of SP and be moved again into the parent compartment identified by c_4, even if such compartment is not present anymore in the original Brane process.

In order to avoid this annoying side effect, we must ensure that any residual SP process is totally isolated from the the system. This can be achieved by using

two names c, c_b for each compartment, c for the low-priority synchronisation of processes corresponding to actions, communications and capabilities, while c_b for the high-priority sequence of broadcast actions. After each broadcast, the name c_b is forgotten completely by all the processes and replaced by a new name c'_b for the next broadcast. Consequently, each SP process remaining after the regular end of the broadcasting loop is not able to communicate with the system, so it is going to disappear spontaneously by executing the internal τ action after a while. If we denote any pair of names n, n_b as $\ddot{n}$, the new encoding functions $[\![\cdot]\!]^{\beta'}$ from Brane to core-$\pi@$ and $[\![\cdot]\!]^{\alpha'}$ from BioAmbients to core-$\pi@$ have almost the same shape as the previous functions $[\![\cdot]\!]^{\beta}$ and $[\![\cdot]\!]^{\alpha}$. For example, the parallel composition of Brane systems is encoded as

$$[\![P \circ Q]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} \triangleq [\![P]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} \mid [\![Q]\!]^{\beta'}_{K,\ddot{p}\ddot{c}}$$

with the only difference being the names of compartments which are now paired as discussed. The definition of the processes $TREE$ and $BCAST$ reveals how name pairs are handled:

$$\begin{aligned} BCAST &\triangleq ! \, \underline{bc}@bcast(x, y, ny_b, \ddot{z}). \\ &\qquad (\tau + \underline{x}@y\langle ny_b, \ddot{z}\rangle.\overline{\underline{bc}}@bcast\langle x, y, ny_b, \ddot{z}\rangle) \\ TREE(b, \ddot{c}, \ddot{p}\ddot{c}) &\triangleq \underline{pc}@c_b(nc_b, \ddot{x}).\overline{\underline{cycle}}@b\langle c, nc_b, \ddot{x}\rangle + \\ &\qquad \underline{exo}@pc_b(npc_b, \ddot{x}).\overline{\underline{cycle}}@b\langle \ddot{c}, \ddot{x}\rangle + \\ &\qquad \underline{exo}@c_b(nc_b, \ddot{x}).\overline{\underline{cycle}}@b\langle \ddot{p}\ddot{c}, \ddot{x}\rangle \end{aligned}$$

The presence of the names *bc* and *cycle* is due to the definition of core-$\pi@$ grammar, which does not allow the denotation of channels by means of only one name.

The $BCAST$ process is not substantially changed, except for the number of names (five instead of three) handled during the loop: the first two identify the channel $x@y$ where the broadcast will occur, while $\ddot{z} = z, z_b$ denote the two names associated with the compartment undergoing some structural change. The name ny_b constitutes the replacement of y for the next broadcast occurring in the related compartment.

Accordingly, the $TREE$ subprocess receives the three names $nc_b, \ddot{x}$ (with $\ddot{x} = x, x_b$) instead of only one. nc_b is used in the first branch of the choice as new broadcast name associated with c substituted by c_b, in consistency with the encoded *phago* operation which updates the names of the parent compartment pc for all the processes in c. The encoded *exo* action is characterised by the peculiar property of eliminating already the name of the inner compartment where the broadcast happens, so that the new name nc_b is useless in the second and third branch of the $TREE$ process (and in fact it disappears). The same would happen with BioAmbients *merge* capability.

The encoding of actions should be consistently updated. The $phago^{\perp}$ co-action is not substantially changed:

$$\begin{aligned}[\![\, phago^{\perp}_{n}(\rho).\sigma \,]\!]^{\beta' *}_{K,\ddot{c},\ddot{pc}} \quad &\triangleq \quad (\nu\ \ddot{x})\big(\overline{phago\_n}@pc\langle\ddot{x}\rangle. \\ &\qquad ([\![\, \sigma \,]\!]^{\beta'}_{\emptyset,\ddot{c},\ddot{pc}} \mid [\![\, \rho \,]\!]^{\beta'}_{\emptyset,\ddot{x},\ddot{c}} \mid \Pi_K)\end{aligned}$$

Each action which triggers a $BCAST$ process requires the creation of the new name nc_b previously discussed:

$$\begin{aligned}[\![\, phago_{n}.\sigma \,]\!]^{\beta' *}_{K,\ddot{c},\ddot{pc}} \quad &\triangleq \quad phago\_n@pc(\ddot{x}).(\nu\ nc_b) \\ &\qquad (\underline{\overline{bc}}@bcast\langle pc, c_b, nc_b, \ddot{x}\rangle.([\![\, \sigma \,]\!]^{\beta'}_{\emptyset,c,nc_b,\ddot{x}} \mid \Pi_K)\end{aligned}$$

nc_b is substituted for c_b here as well, as we can notice from the parameters used in the recursive call of the encoding function.

In Table 5.5 and Table 5.6 the full encoding of Brane in core-$\pi@$ is reported. The encoding of BioAmbients is based on the same ideas and does not require further insights.

5.2 Encoding catalytic P systems in $\pi@$

P systems and process calculi present many similar aspects: interaction between different elements (by direct communication in π-calculus-like languages, by transition rules in P systems), localisation of interaction (within the same membrane in P

systems, within the scope of a name in π-calculus-like languages or within the same compartment in bio-inspired calculi), concurrency, nondeterminism. However, they come from distant areas, in fact they are composed of totally different elements: P systems are made of objects, (sometimes prioritised) rules, membranes. In process calculi there is only one kind of elements: processes. So it is quite natural to expect that every element of a P system will be translated in a π@ process. One non trivial difference to overcome is the maximal parallelism typical of P systems: even if process calculi deal with concurrent objects and describe parallel evolution, there is no immediate correspondence with such a strong constraint in π@, which means that maximal parallelism must be injected in some way within the behaviour of every process.

In the next section basic ideas of encoding catalytic P systems in π@ are presented, first by defining the general form of the encoding function and then by specifying its simplest parts without considering the difficulties introduced by maximal parallelism. In Section 5.2.2 the encoding function is finally specified, preserving modularity and divergence/termination properties of the encoded systems even after the introduction of maximal parallelism constraints.

5.2.1 Encoding ideas

Despite their similarities, the different features of membrane systems and process calculi make less obvious the encoding of the selected kind of P systems in π@, with respect to both *how* such an encoding may be realised and *which requirements* it should satisfy. We may try to adapt the set of constraints defined for a suitable encoding to this new setting.

As for the previous encoding, we need to preserve some kind of *operational correspondence*: if there is a transition between two configurations of a P system $C_1 \longrightarrow C_2$, the encoded system $[\![C_1]\!]$ should be able to perform (in one or more steps) a transition $[\![C_1]\!] \Longrightarrow [\![C_2]\!]$, while if $[\![C_1]\!] \Longrightarrow Q$, then a configuration C should exist such that $Q \Longrightarrow [\![C]\!]$ and $C_1 \longrightarrow \ldots \longrightarrow C$.

The encoding should also fulfill some modularity requirement, although the syntax and semantics of P systems do not allow the expression of such requirements as homomorphism of the operator of parallel composition. In fact there is no such operator in P systems, and the description of the state of the system (the kind of elements and their multiplicity) is separated from the specification of their behaviour (i.e. the reaction rules they undergo) and the structure of the system (the tree of membranes). The more challenging result in terms of modularity would consist of the possibility of encoding separately each of these entities: in the following we show how this can be achieved.

In this section we expose the main ideas which help to provide an encoding satisfying the above conditions, while in the next section we show the final version of the encoding function which also satifies additional constraints about the preservation of divergence and termination.

Before sketching the first encodings, we define the encoding function in its general form:

Definition 5.2 *Given a catalytic P system* $\Pi \in \Pi_{Dom}$

$$\Pi = (V, C, \mu, w_1^0, \ldots, w_d^0, R_1, \ldots, R_d, i_0)$$

where Π_{Dom} *represents the domain of all the possible catalytic P systems, the encoding function* $[\![\ ]\!] : \Pi_{Dom} \longrightarrow \mathcal{P}$ *is defined as*

$$[\![\, \Pi \,]\!] \stackrel{def}{=} [\![\, \mu \,]\!]_{MS} \mid [\![\, w_1^0, 1 \,]\!]_S \mid \ldots \mid [\![\, w_d^0, d \,]\!]_S \mid [\![\, R_1, 1 \,]\!]_R \mid \ldots \mid [\![\, R_d, d \,]\!]_R$$

where

$$[\![\ ]\!]_{MS} : \overline{MS} \longrightarrow \mathcal{P}, \qquad [\![\ ]\!]_S : V^* \times I\!N \longrightarrow \mathcal{P}, \qquad [\![\ ]\!]_R : R_{Dom} \times I\!N \longrightarrow \mathcal{P},$$

$\overline{MS}$, V^*, R_{Dom} *standing respectively for the sets of all possible membrane structures, multisets of objects (i.e. strings) and evolution rules.*

The next task is the definition of $[\![\ ]\!]_{MS}$, $[\![\ ]\!]_S$, $[\![\ ]\!]_R$, on which the definition of $[\![\ ]\!]$ depends, but some remarks are needed first.

Even without specifying the functions $[\![\ ]\!]_{MS}$, $[\![\ ]\!]_S$, $[\![\ ]\!]_R$, it is easy to observe that the encoding is completely modular: every piece can be added, eliminated or modified without affecting the rest of the encoded system. In this way we obtain exactly our aim, rules do not depend on objects and vice versa, but there is something more: objects or rules encoding does not depend on membrane structure, since the only information needed is the number (or the label) of the membrane containing the object or rule.

A P system is essentially a set of objects observing certain evolution rules: if we have the rule $ca \to cbd$, the object a can evolve to two new objects b, d, but only if a catalyst c is nearby. Perhaps the most natural way to compose the steps of this evolution is: "if an object of type a meets the catalyst c, then it evolves into two new objects b and d", thus the objects are thought of as the main actors in this process. But if we add another rule, $a \to e$, then the behaviour of the object a considerably changes: "one possibility is that if the object a meets the catalyst c, then it evolves into two new objects b and d; another one is that the object a evolves into the object e". So if we think of objects as the main actors of evolution, probably we are not going to find a modular encoding. The key for avoiding this problem is to spot the real actors: evolution rules. The rule $ca \to cbd$ could be expressed in this way: "forever do: check if an object of type a is nearby, then check if a catalyst c is nearby too; if they are, replace a with b and d". Hence, adding another rule would be not difficult: $a \to e$ would just become another process executing "forever do: if a is nearby, then replace it with e". The behaviour of objects would then become trivial, in fact a would be "if some rule asks for object a, say a is present". Obviously other issues would then pop up, caused for example by rules competing for the same objects, but we will face them later.

Following this interpretation, we could have (ignoring the second parameter of

the encoding functions, for the moment)

$$[\![a]\!]_S = \overline{a}\,, \qquad [\![c]\!]_S = \overline{c}$$

$$[\![ca \rightarrow cbd]\!]_R \quad = \quad !\, c.a.(\overline{c} \mid \overline{b} \mid \overline{d})$$

In fact

$$\begin{array}{l} \overline{a} \mid \overline{c} \mid !\, c.a.(\overline{c} \mid \overline{b} \mid \overline{d}) \quad \rightarrow \\ \overline{a} \mid a.(\overline{c} \mid \overline{b} \mid \overline{d}) \mid !\, c.a.(\overline{c} \mid \overline{b} \mid \overline{d}) \quad \rightarrow \\ \overline{c} \mid \overline{b} \mid \overline{d} \mid !\, c.a.(\overline{c} \mid \overline{b} \mid \overline{d}) \end{array}$$

but in this way if the object a is not present, object c disappears even if not used and the rule process is deadlocked:

$$\overline{c} \mid !\, c.a.(\overline{c} \mid \overline{b} \mid \overline{d}) \quad \rightarrow \quad a.(\overline{c} \mid \overline{b} \mid \overline{d}) \mid !\, c.a.(\overline{c} \mid \overline{b} \mid \overline{d})$$

So a correction is necessary

$$[\![ca \rightarrow cbd]\!]_R \quad = \quad !\, c.(c + a.(c \mid b \mid d))$$

so that the checking phase does not delete even unused objects. But another issue appears, because the rule above may execute forever without producing its output objects, even if input objects a and c are present: we have introduced divergence. In fact, after checking the presence of c, we should be able to give some kind of precedence to the second term of the choice, in order to ensure that if some object a is present, then the rule completes its job. By means of prioritised choice, it is possible to model this kind of precedence:

$$[\![ca \rightarrow cbd]\!]_R \quad = \quad !\, c.(\tau.\underline{c} + a.(\underline{c} \mid \underline{b} \mid d))$$

In this way we obtain that if a given rule *may* be applied, for sure it is applied, however divergence is not completely removed yet: if only object c is present, the above process would continue its checking forever. Since maximal parallelism introduces again the same kind of problem, we defer the final solution to the next section.

One aspect we can consider before dealing with complex encodings is the localisation of communication: objects located in one membrane shall not interact with rules located in another membrane. $\pi@$ allows the modelling of this constraint in a very simple way, by means of polyadic synchronisation. The encoding functions $[\![\ ]\!]_R$, $[\![\ ]\!]_S$ require two parameters: the first is the object or rule to be encoded, the second is the number (or label) of the surrounding membrane. Localised communication is then obtained easily by specifying the membrane number for every object manipulated by evolution rules:

$$[\![\ c, n\ ]\!]_R \quad = \quad \overline{\underline{c@n}}\ , \qquad [\![\ a, n\ ]\!]_R \quad = \quad \overline{\underline{a@n}}$$

$$[\![\ ca \rightarrow cbd, n\ ]\!]_R \quad = \quad !\ \underline{c@n}.(\tau.\overline{\underline{c@n}} + \underline{a@n}.(\overline{\underline{c@n}} \mid \overline{\underline{b@n}} \mid \overline{\underline{d@n}}))$$

The next step is to understand how to express rules producing objects outside membrane boundaries. For example, encoding the rule $ca \rightarrow c(b, out)$, which ejects b into the external membrane, requires some knowledge about membrane structure. The definition 5.2 asserts that the only process owning this knowledge is $[\![\ \mu\ ]\!]_{MS}$, which is supposed to be properly queried. Thus, a possible solution may be

$$[\![\ ca \rightarrow c(b, out), n\ ]\!]_R \quad = \quad !\ c@n.(\tau.\underline{c@n} + a@n.(\underline{c@n} \mid \underline{out@n}(x).\underline{b@x}))$$

where the instruction $\underline{out@n}(x)$ queries the membrane structure process about the label of the outer membrane.

Hence, membrane structure process could be seen as a service replying to two types of queries: "tell me the label of the membrane surrounding membrane n", or "tell me the label of one of membranes inside membrane n". The second type of query introduces some kind of nondeterminism, because each membrane may contain more than one child membrane, so it is necessary to preserve this nondeterministic behaviour after the encoding. The easiest way to translate a membrane structure in $\pi@$ is considering each membrane as a separated process. Thus, the first type of query imposes that every membrane process "knows" the label of its surrounding membrane. In other words, if the membrane structure is a tree, we are going to implement it with *pointers* from child membranes to parent membranes. But the

second type of rule requires that pointers from parent membranes to the inner ones also exist: including these pointers in the child membranes processes allows the encoding of each membrane completely disregarding its content.

Definition 5.3 *Given a membrane structure $\mu \in \overline{MS}$, the encoding function*

$$[\![\ ]\!]_{MS} : \overline{MS} \longrightarrow \mathcal{P}$$

is defined as

$$[\![\ [_1\ \mu_{i_1}, \ldots, \mu_{i_k}\ ]_1\ ]\!]_{MS} \stackrel{def}{=}$$

$$! \ \overline{out@1}\langle outside\rangle \mid [\![\ \mu_{i_1}, 1\ ]\!]'_{MS} \mid \ \ldots\ \mid [\![\ \mu_{i_k}, 1\ ]\!]'_{MS}$$

where

$$[\![\ ]\!]'_{MS} : \overline{MS} \times I\!N \longrightarrow \mathcal{P}$$

is defined as

$$[\![\ [_n\ \mu_{i_1}, \ldots, \mu_{i_k}\ ]_n,\ p\ ]\!]'_{MS} \stackrel{def}{=}$$

$$! \ \overline{out@n}\langle p\rangle \mid ! \ \overline{in@p}\langle n\rangle \mid [\![\ \mu_{i_1}, n\ ]\!]'_{MS} \mid \ \ldots\ \mid [\![\ \mu_{i_k}, n\ ]\!]'_{MS}$$

The main encoding function $[\![\ ]\!]_{MS}$ is defined in terms of an auxiliary function $[\![\ ]\!]'_{MS}$, which requires an additional parameter: the label of the parent membrane. The skin has no parent membrane, so we provide a fictitious name *outside* to express the compartment containing the objects ejected from skin and never allowed to enter again. The encoding of each membrane is completely modular, hence any change to the membrane structure tree affects only the encodings of the nodes directly involved: adding or removing a whole subtree requires no changes to the remaining structure.

Before facing divergence issues and maximal parallelism, the definition of $[\![\ ]\!]_S$ is given.

Definition 5.4 *Given a string $s \in V^*, s = s_1 s_2 \dots s_n$, the encoding function*

$$[\![\ ]\!]_S : V^* \longrightarrow \mathcal{P}$$

is defined as

$$[\![\ s\ ]\!]_S \quad \stackrel{def}{=} \quad \underline{\overline{s_1}} \mid \underline{\overline{s_2}} \mid \dots \mid \underline{\overline{s_n}}$$

Again, the encoding is completely modular. Furthermore, it does not distinguish catalysts from common objects, in fact the distinction is only useful before compile-time to check the correctness of P system rules. The reason for the higher degree of priority will be clear in the next section.

5.2.2 Final encodings

Two issues persist: the divergence introduced in the encoding of evolution rules and maximal parallelism. Divergence is caused essentially by an endless loop. In fact, the rule

$$[\![\ ca \to cbd\ ]\!]_R \quad = \quad !\, c.(\tau.\underline{c} + a.(\underline{c} \mid \underline{b} \mid d))$$

contains a guarded loop, but the guard has no effect if the catalyst c is present, because it never disappears. So an additional guard is required:

$$[\![\ ca \to cbd\ ]\!]_R \quad = \quad \overline{coin} \mid !\, coin.c.(\tau.c + a.(c \mid b \mid d \mid \overline{coin}))$$

First, a *coin* is given, then every execution of the rule eats a *coin*, but only a succesfull execution produces another *coin*: after the first failure, the loop ends. Yet, there are issues: in presence of two or more rules, the *coin* owned by a rule could be used by another one. Furthermore, if object a or c is absent, the process may pause in a spurious state and allow other rules to begin computing before ending execution or releasing back the catalyst c. Even if it is possible to prove that the final result does not change, this breaks the idea of *atomicity* for the application of a rule. Thus, channel *coin* must be private and the priority of every action shall be increased:

$$[\![\ ca \to cbd\ ]\!]_R \quad = \quad (\nu\ \ coin)(\overline{coin} \mid !\, coin.(\underline{\tau} + c.(\underline{\tau}.c + a.(c \mid b \mid d \mid \overline{coin}))))$$

Maximal parallelism is a heavy constraint on the order of actions: objects may be used only once for every tick of a sort of global clock. From an algorithmic point of view, it may be thought of as a global loop of this kind:

while (in each tick at least one rule can be applied) do
while (in current tick some rule can be applied) do
apply a rule, freezing, for current tick, produced objects
done
let clock tick, unfreezing just produced objects
done

The outermost loop can be translated with the same scheme used to overcome divergence in the encoding of rules: the only requirement is that rules signal in some way their application performed during each clock tick, in order to allow the loop to continue if at least one rule has been applied at least once:

$$\begin{aligned}\llbracket\, ca \to cbd \,\rrbracket_R \quad = \quad & (\nu\ coin)(!\ coin.(\underline{\tau}.\underline{\underline{tick}}.\overline{\overline{coin}} + \underline{c}.(\underline{\tau}.(\underline{\overline{c}} \mid \underline{\underline{tick}}.\overline{\overline{coin}}) + \\ & \underline{a}.(\underline{\underline{tick}}(\underline{\overline{c}} \mid \underline{\overline{b}} \mid \underline{\overline{d}}) \mid \overline{coin} \mid \overline{\underline{\underline{worked}}}))) \mid \underline{\underline{tick}}.\overline{coin})\end{aligned}$$

Now, every application of the rule produces a $\overline{\underline{\underline{worked}}}$ process, signaling its application, and the rule waits for its *coin* from the $\overline{\underline{\underline{tick}}}$ of the global clock. When the rule application fails (i.e. when the $\underline{\tau}$ actions are performed), a new process waits for the next *coin*. The objects $\underline{\overline{c}}, \underline{\overline{b}}, \underline{\overline{d}}$ produced by the rule are also frozen until next clock $\overline{\underline{\underline{tick}}}$. The global clock can expressed by a low priority loop:

$$!\ clock:3.\underline{\underline{worked}}.\underline{\overline{bcast}}\langle worked\rangle.\tau.\underline{\overline{bcast}}\langle tick\rangle.\overline{clock:3} \mid \overline{clock:3} \mid \overline{\underline{\underline{worked}}}$$

The first instruction, executed with the lowest priority, $clock:3$, is the guard for the global clock loop, working in the same way of the one previously seen: its *coin* is represented by $\overline{clock:3}$. As soon as this low priority guard fires, the presence of at least one $\overline{\underline{\underline{worked}}}$ process is checked and the potential duplicates are burned by a subsequent broadcast on the same name. Finally, all the processes waiting for the next clock tick (rules waiting for their coin, or frozen objects) are awakened by another broadcast.

Recalling previous considerations about localisation and movement of objects between membranes, the encoding function for rules can finally be formalised:

Definition 5.5 *Given an evolution rule $E \in R_{Dom}$, the function*

$$[\![\ ]\!]_R : R_{Dom} \times I\!N \longrightarrow \mathcal{P}$$

is defined as

$$[\![\ E, n\ ]\!]_R \stackrel{def}{=} (\nu\ coin)(!\ coin.(\tau.\underline{\underline{tick}}.\overline{coin} + [\![\ E, n\ ]\!]'_R) \mid \underline{\underline{tick}}.\overline{coin}) \mid \mid CLOCK \mid BCAST$$

where

$$CLOCK \equiv !\ clock\!:\!3.\underline{\underline{worked}}.\overline{\underline{\underline{bcast}}}\langle worked\rangle.\tau.\overline{\underline{\underline{bcast}}}\langle tick\rangle.\overline{clock\!:\!3} \mid \mid \overline{clock\!:\!3} \mid \overline{\underline{\underline{worked}}}$$

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

The function

$$[\![\ ]\!]'_R : R_{Dom} \times I\!N \longrightarrow \mathcal{P}$$

is defined as

$$[\![\ ca \to cv, n\ ]\!]'_R \stackrel{def}{=} \overline{\underline{\underline{c@n}}}.(\tau.(\overline{\underline{\underline{c@n}}} \mid \underline{\underline{tick}}.\overline{coin}) + [\![\ a \to (c, here)v, n\ ]\!]'_R)$$

$$[\![\ a \to v_1 \ldots v_k, n\ ]\!]'_R \stackrel{def}{=} \overline{\underline{\underline{a@n}}}.(\overline{coin} \mid \overline{\underline{\underline{worked}}} \mid \mid [\![\ v_1, n\ ]\!]''_R \mid \ldots \mid [\![\ v_k, n\ ]\!]''_R)$$

where $a \in V \setminus C$, $c \in C$, $v \in (V \times \{here, out, in\})^*$, $v = v_1 \ldots v_n$, V alphabet of the P system, C set of catalysts.

Finally, the function

$$[\![\ ]\!]''_R : (V \times \{here, out, in\}) \times I\!N \longrightarrow \mathcal{P}$$

is defined as

$$[\![\ (a, here), n\ ]\!]''_R \stackrel{def}{=} \underline{\underline{tick}}.\overline{\underline{\underline{a@n}}}$$

$$[\![\ (a, out), n\ ]\!]''_R \stackrel{def}{=} \underline{out@n}(x).\underline{\underline{tick}}.\overline{\underline{\underline{a@x}}}$$

$$[\![\ (a, in), n\ ]\!]''_R \stackrel{def}{=} \underline{in@n}(x).\underline{\underline{tick}}.\overline{\underline{\underline{a@x}}}$$

The definition of $CLOCK$ and $BCAST$ does not depend on any rule, so an immediate optimisation would be insert them in the encoding of the function $[\![\ ]\!]$ in order to avoid useless duplicates, but this affects only the compilation phase.

The above encoding function $[\![\cdot]\!]$ is summarised in Table 5.7.

$$
\begin{array}{rcl}
[\![\ \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} \\
[\![\ !\ 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 \\
 & & \quad [\![\ P\ ]\!]^{\alpha}_{K\cup\{b\},a,pa} \mid [\![\ P\ ]\!]^{\alpha}_{K\cup\{b\},a,pa} \mid \\
 & & \quad !\ \underline{\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 \Big(\sum_{i\in I,\ I\neq\emptyset} [\![\ \xi_i.P_i\ ]\!]^{\alpha *}_{K,na,npa} \\
 & & \quad\quad + TREE(s,na,npa)\Big) \mid \\
 & & \quad \sum_{i\in I,\ I\neq\emptyset} [\![\ \xi_i.P_i\ ]\!]^{\alpha *}_{K,a,pa} + TREE(s,a,pa)) \\
 & & \\
BANG(b,a,pa) & \triangleq & !\ \underline{b}(na,npa).SUBBANG(b,na,npa) \\
 & & \quad \mid SUBBANG(b,a,pa) \\
SUBBANG(b,na,npa) & \triangleq & \underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle na,npa\rangle.\overline{\underline{b}}\langle na,npa\rangle + \\
 & & \quad TREE(b,na,npa) \\
TREE(b,na,npa) & \triangleq & \underline{\underline{npa@na}}(x).\overline{\underline{b}}\langle na,x\rangle + \underline{\underline{merge@npa}}(x).\overline{\underline{b}}\langle na,x\rangle + \\
 & & \underline{\underline{merge@na}}(x).\overline{\underline{b}}\langle x,npa\rangle \\
BCAST & \triangleq & !\ \underline{bcast}(x,y,z).(\overline{\underline{\underline{x@y}}}\langle z\rangle.\overline{\underline{bcast}}\langle x,y,z\rangle + \underline{\tau})
\end{array}
$$

Table 5.1: Encoding of BioAmbients processes in $\pi@$.

$$
\begin{array}{rcl}
[\![\ enter\ n.P\ ]\!]^{\alpha*}_{K,a,pa} & \triangleq & enter@n@pa(x).\overline{\underline{\underline{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).\overline{\underline{\underline{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 \overline{\underline{\underline{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)
\end{array}
$$

$$
\begin{array}{rcl}
\Pi_K & \triangleq & \overline{\underline{\underline{unfold@k_1}}} \mid \cdots \mid \overline{\underline{\underline{unfold@k_n}}} \quad , \\
 & & \text{with } K = \{k_1,\ldots,k_n\}
\end{array}
$$

Table 5.2: Encoding of BioAmbients communications and capabilities in $\pi@$.

$$
\begin{array}{rcl}
[\![\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} \\
[\![\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} \\
[\![! P]\!]^{\beta}_{K,pc} & \triangleq & (\nu\ b)(BANG'(b,pc) \mid [\![P]\!]^{\beta}_{K\cup\{b\},pc} \mid \\
 & & \quad [\![P]\!]^{\beta}_{K\cup\{b\},pc} \mid ! \underline{\underline{new@b}}(npc).[\![P]\!]^{\beta}_{K\cup\{b\},npc}) \\
[\![! \sigma]\!]^{\beta}_{K,c,pc} & \triangleq & (\nu\ b)(BANG(b,c,pc) \mid [\![\sigma]\!]^{\beta}_{K\cup\{b\},c,pc} \mid \\
 & & \quad [\![\sigma]\!]^{\beta}_{K\cup\{b\},c,pc} \mid ! \underline{\underline{new@b}}(nc,npc).[\![\sigma]\!]^{\beta}_{K\cup\{b\},nc,npc}) \\
BANG(b,c,pc) & \triangleq & ! \underline{b}(nc,npc).SUBBANG(b,nc,npc) \mid \\
 & & SUBBANG(b,c,pc) \\
BANG'(b,npc) & \triangleq & ! \underline{b}(npc).SUBBANG'(b,npc) \mid SUBBANG'(b,pc) \\
SUBBANG(b,nc,npc) & \triangleq & \underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle nc,npc\rangle.\overline{\underline{b}}\langle nc,npc\rangle + TREE(b,nc,npc) \\
SUBBANG'(b,npc) & \triangleq & \underline{\underline{unfold@b}}.\overline{\underline{\underline{new@b}}}\langle npc\rangle.\overline{\underline{b}}\langle npc\rangle \ + \ \underline{\underline{exo@npc}}(x).\overline{\underline{b}}\langle x\rangle \\
TREE(b,nc,npc) & \triangleq & \underline{\underline{npc@nc}}(x).\overline{\underline{b}}\langle nc,x\rangle + \underline{\underline{exo@npc}}(x).\overline{\underline{b}}\langle nc,x\rangle + \\
 & & \underline{\underline{exo@nc}}(x).\overline{\underline{b}}\langle npc,x\rangle
\end{array}
$$

Table 5.3: Encoding of Brane processes in $\pi@$

$$
\begin{array}{rcl}
[\![\, a.\sigma \,]\!]^{\beta}_{K,c,pc} & \triangleq & BCAST \mid (\nu\ s)(!\ \underline{s}(nc,npc). \\
 & & ([\![\, a.\sigma \,]\!]^{\beta*}_{K,nc,npc} + TREE(s,nc,npc)) \mid \\
 & & [\![\, a.\sigma \,]\!]^{\beta*}_{K,c,pc} + TREE(s,c,pc)) \\
[\![\, phago_n.\sigma \,]\!]^{\beta*}_{K,c,pc} & \triangleq & phago@n@pc(x).\overline{\underline{bcast}}\langle pc,c,x\rangle.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,x} \mid \Pi_K) \\
[\![\, phago^{\perp}_n(\rho).\sigma \,]\!]^{\beta*}_{K,c,pc} & \triangleq & (\nu\ x)\big(\overline{phago@n@pc}\langle x\rangle.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,pc} \mid [\![\, \rho \,]\!]^{\beta}_{\emptyset,x,c} \mid \Pi_K)\big) \\
[\![\, exo_n.\sigma \,]\!]^{\beta*}_{K,c,pc} & \triangleq & exo@n@pc(x).\overline{\underline{bcast}}\langle exo,c,x\rangle.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,pc,x} \mid \Pi_K) \\
[\![\, exo^{\perp}_n.\sigma \,]\!]^{\beta*}_{K,c,pc} & \triangleq & \overline{exo@n@c}\langle pc\rangle.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,pc} \mid \Pi_K) \\
[\![\, pino(\rho).\sigma \,]\!]^{\beta*}_{K,c,pc} & \triangleq & (\nu\ x)\tau.([\![\, \sigma \,]\!]^{\beta}_{\emptyset,c,pc} \mid [\![\, \rho \,]\!]^{\beta}_{\emptyset,x,c} \mid \Pi_K) \\
 & & \\
\Pi_K & \triangleq & \overline{\underline{unfold@k_1}} \mid \cdots \mid \overline{\underline{unfold@k_n}} \quad , \\
 & & K = \{k_1,\ldots,k_n\} \\
BCAST & \triangleq & !\ \underline{bcast}(x,y,z).(\overline{\underline{x@y}}\langle z\rangle.\overline{\underline{bcast}}\langle x,y,z\rangle + \underline{\tau})
\end{array}
$$

Table 5.4: Encoding of Brane actions in $\pi@$

$$
\begin{array}{rcl}
[\![\diamond]\!]^{\beta'} & \triangleq & \mathbf{0} \\
[\![P \circ Q]\!]^{\beta'} & \triangleq & [\![P]\!]^{\beta'} \mid [\![Q]\!]^{\beta'} \\
[\![! P]\!]^{\beta'} & \triangleq & [\![! P]\!]^{\beta'}_{\emptyset,\ddot{o}\ddot{c}} \\
[\![\sigma(\!|P|\!)]\!]^{\beta'} & \triangleq & [\![\sigma(\!|P|\!)]\!]^{\beta'}_{\emptyset,\ddot{o}\ddot{c}} \\
[\![\diamond]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} & \triangleq & \mathbf{0} \\
[\![P \circ Q]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} & \triangleq & [\![P]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} \mid [\![Q]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} \\
[\![\sigma(\!|P|\!)]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} & \triangleq & (\nu\ \ddot{c})\big([\![\sigma]\!]^{\beta'}_{K,\ddot{c},\ddot{p}\ddot{c}} \mid [\![P]\!]^{\beta'}_{K,\ddot{c}}\big) \\
[\![\mathbf{0}]\!]^{\beta'}_{K,\ddot{c},\ddot{p}\ddot{c}} & \triangleq & \mathbf{0} \\
[\![\sigma \mid \rho]\!]^{\beta'}_{K,\ddot{c},\ddot{p}\ddot{c}} & \triangleq & [\![\sigma]\!]^{\beta'}_{K,\ddot{c},\ddot{p}\ddot{c}} \mid [\![\rho]\!]^{\beta'}_{K,\ddot{c},\ddot{p}\ddot{c}} \\
[\![! P]\!]^{\beta'}_{K,\ddot{p}\ddot{c}} & \triangleq & (\nu\ b)\big(BANG'(b,\ddot{p}\ddot{c}) \mid [\![P]\!]^{\beta'}_{K\cup\{b\},\ddot{p}\ddot{c}} \mid \\
 & & [\![P]\!]^{\beta'}_{K\cup\{b\},\ddot{p}\ddot{c}} \mid ! \underline{new}@b(n\ddot{p}\ddot{c}).[\![P]\!]^{\beta'}_{K\cup\{b\},n\ddot{p}\ddot{c}}\big) \\
[\![! \sigma]\!]^{\beta'}_{K,\ddot{c},\ddot{p}\ddot{c}} & \triangleq & (\nu\ b)\big(BANG(b,\ddot{c},\ddot{p}\ddot{c}) \mid [\![\sigma]\!]^{\beta'}_{K\cup\{b\},\ddot{c},\ddot{p}\ddot{c}} \mid \\
 & & [\![\sigma]\!]^{\beta'}_{K\cup\{b\},\ddot{c},\ddot{p}\ddot{c}} \mid ! \underline{new}@b(n\ddot{c},n\ddot{p}\ddot{c}).[\![\sigma]\!]^{\beta'}_{K\cup\{b\},n\ddot{c},n\ddot{p}\ddot{c}}\big) \\
BANG(b,\ddot{c},\ddot{p}\ddot{c}) & \triangleq & ! \underline{cycle}@b(n\ddot{c},n\ddot{p}\ddot{c}).SUBBANG(b,n\ddot{c},n\ddot{p}\ddot{c}) \mid \\
 & & SUBBANG(b,\ddot{c},\ddot{p}\ddot{c}) \\
BANG'(b,n\ddot{p}\ddot{c}) & \triangleq & ! \underline{cycle}@b(n\ddot{p}\ddot{c}).SUBBANG'(b,n\ddot{p}\ddot{c}) \mid SUBBANG'(b,\ddot{p}\ddot{c}) \\
SUBBANG(b,n\ddot{c},n\ddot{p}\ddot{c}) & \triangleq & \underline{unfold}@b().\overline{\underline{new}}@b\langle n\ddot{c},n\ddot{p}\ddot{c}\rangle.\overline{\underline{cycle}}@b\langle n\ddot{c},n\ddot{p}\ddot{c}\rangle + \\
 & & TREE(b,n\ddot{c},n\ddot{p}\ddot{c}) \\
SUBBANG'(b,n\ddot{p}\ddot{c}) & \triangleq & \underline{unfold}@b().\overline{\underline{new}}@b\langle n\ddot{p}\ddot{c}\rangle.\overline{\underline{cycle}}@b\langle n\ddot{p}\ddot{c}\rangle + \\
 & & \quad \underline{exo}@npc_b(nnpc_b,\ddot{x}).\overline{\underline{cycle}}@b\langle\ddot{x}\rangle \\
TREE(b,n\ddot{c},n\ddot{p}\ddot{c}) & \triangleq & \underline{npc}@nc_b(nnc_b,\ddot{x}).\overline{\underline{cycle}}@b\langle nc,nnc_b,\ddot{x}\rangle + \\
 & & \underline{exo}@npc_b(nnpc_b,\ddot{x}).\overline{\underline{cycle}}@b\langle n\ddot{c},\ddot{x}\rangle + \\
 & & \underline{exo}@nc_b(nnc_b,\ddot{x}).\overline{\underline{cycle}}@b\langle n\ddot{p}\ddot{c},\ddot{x}\rangle
\end{array}
$$

Table 5.5: Encoding of Brane processes in core-$\pi@$

$$
\begin{array}{rcl}
[\![\, a.\sigma \,]\!]^{\beta'}_{K,\ddot{c},\ddot{p}c} & \triangleq & BCAST \mid (\nu\ s)\big(!\ \underline{cycle}@s(\ddot{n}c, n\ddot{p}c).\\
& & ([\![\, a.\sigma \,]\!]^{\beta'*}_{K,\ddot{n}c,n\ddot{p}c} + TREE(s,\ddot{n}c,n\ddot{p}c)) \mid\\
& & [\![\, a.\sigma \,]\!]^{\beta'*}_{K,\ddot{c},\ddot{p}c} + TREE(s,\ddot{c},\ddot{p}c)\big)\\
[\![\, phago_n.\sigma \,]\!]^{\beta'*}_{K,\ddot{c},\ddot{p}c} & \triangleq & phago\_n@pc(\ddot{x}).\\
& & (\nu\ nc_b)(\overline{\underline{bc}}@bcast\langle pc, c_b, nc_b, \ddot{x}\rangle.([\![\, \sigma \,]\!]^{\beta'}_{\emptyset,c,nc_b,\ddot{x}} \mid \Pi_K)\\
[\![\, phago^{\perp}_n(\rho).\sigma \,]\!]^{\beta'*}_{K,\ddot{c},\ddot{p}c} & \triangleq & (\nu\ \ddot{x})\big(\overline{phago\_n}@pc\langle\ddot{x}\rangle.([\![\, \sigma \,]\!]^{\beta'}_{\emptyset,\ddot{c},\ddot{p}c} \mid [\![\, \rho \,]\!]^{\beta'}_{\emptyset,\ddot{x},\ddot{c}} \mid \Pi_K\big)\\
\\
[\![\, exo_n.\sigma \,]\!]^{\beta'*}_{K,\ddot{c},\ddot{p}c} & \triangleq & exo\_n@pc(\ddot{x}).\\
& & (\nu\ nc_b)(\ \overline{\underline{bc}}@bcast\langle exo, c_b, nc_b, \ddot{x}\rangle.[\![\, \sigma \,]\!]^{\beta'}_{\emptyset,\ddot{p}c,\ddot{x}} \mid \Pi_K)\\
[\![\, exo^{\perp}_n.\sigma \,]\!]^{\beta'*}_{K,\ddot{c},\ddot{p}c} & \triangleq & \overline{exo\_n}@c\langle\ddot{p}c\rangle.([\![\, \sigma \,]\!]^{\beta'}_{\emptyset,\ddot{c},\ddot{p}c} \mid \Pi_K)\\
[\![\, pino(\rho).\sigma \,]\!]^{\beta'*}_{K,\ddot{c},\ddot{p}c} & \triangleq & (\nu\ \ddot{x})\tau.([\![\, \sigma \,]\!]^{\beta'}_{\emptyset,\ddot{c},\ddot{p}c} \mid [\![\, \rho \,]\!]^{\beta'}_{\emptyset,\ddot{x},\ddot{c}} \mid \Pi_K)\\
\\
\Pi_K & \triangleq & \overline{\underline{unfold}}@k_1\langle\rangle \mid \cdots \mid \overline{\underline{unfold}}@k_n\langle\rangle\\
& & \text{with} \quad K = \{k_1, \ldots, k_n\}\\
BCAST & \triangleq & !\ \underline{bc}@bcast(x, y, ny_b, \ddot{z}).\\
& & (\tau + \underline{x}@y\langle ny_b, \ddot{z}\rangle.\overline{\underline{bc}}@bcast\langle x, y, ny_b, \ddot{z}\rangle)
\end{array}
$$

Table 5.6: Encoding of Brane actions in core-$\pi@$

$$[\![\ \Pi\ ]\!] \stackrel{def}{=} [\![\ \mu\ ]\!]_{MS} \mid [\![\ w_1^0, 1\ ]\!]_S \mid \ \ldots\ \mid [\![\ w_d^0, d\ ]\!]_S \mid$$
$$[\![\ R_1, 1\ ]\!]_R \mid \ \ldots\ \mid [\![\ R_d, d\ ]\!]_R$$
$$\text{with } \Pi = (V, C, \mu, w_1^0, \ldots, w_d^0, R_1, \ldots, R_d, i_0)$$

$$[\![\ [_1\ \mu_{i_1}, \ldots, \mu_{i_k}\ ]_1\ ]\!]_{MS} \stackrel{def}{=} !\ \overline{\underline{out@1}}\langle outside\rangle \mid [\![\ \mu_{i_1}, 1\ ]\!]'_{MS} \mid \ \ldots\ \mid [\![\ \mu_{i_k}, 1\ ]\!]'_{MS}$$

$$[\![\ [_n\ \mu_{i_1}, \ldots, \mu_{i_k}\ ]_n,\ p\ ]\!]'_{MS} \stackrel{def}{=} !\ \overline{\underline{out@n}}\langle p\rangle \mid !\ \overline{\underline{in@p}}\langle n\rangle \mid$$
$$[\![\ \mu_{i_1}, n\ ]\!]'_{MS} \mid \ \ldots\ \mid [\![\ \mu_{i_k}, n\ ]\!]'_{MS}$$

$$[\![\ s, n\ ]\!]_S \stackrel{def}{=} \overline{\underline{\underline{s_1@n}}} \mid \ \cdots\ \mid \overline{\underline{\underline{s_p@n}}} \qquad s = s_1 \ldots s_p$$

$$[\![\ E, n\ ]\!]_R \stackrel{def}{=} (\nu\ \ coin)(!\ coin.(\underline{\tau}.\underline{\underline{tick}}.\overline{coin} + [\![\ E, n\ ]\!]'_R) \mid \underline{\underline{tick}}.\overline{coin}) \mid$$
$$\mid CLOCK \mid BCAST$$

$$[\![\ ca \to cv, n\ ]\!]'_R \stackrel{def}{=} \overline{\underline{\underline{c@n}}}.(\underline{\tau}.(\overline{\underline{\underline{c@n}}} \mid \underline{\underline{tick}}.\overline{coin}) + [\![\ a \to (c, here)v, n\ ]\!]'_R)$$

$$[\![\ a \to v_1 \ldots v_k, n\ ]\!]'_R \stackrel{def}{=} \overline{\underline{\underline{a@n}}}.(\overline{coin} \mid \overline{\underline{\underline{worked}}} \mid [\![\ v_1, n\ ]\!]''_R \mid \ \ldots\ \mid [\![\ v_k, n\ ]\!]''_R)$$

$$[\![\ (a, here), n\ ]\!]''_R \stackrel{def}{=} \underline{\underline{tick}}.\overline{\underline{\underline{a@n}}}$$

$$[\![\ (a, out), n\ ]\!]''_R \stackrel{def}{=} \underline{out@n}(x).\underline{\underline{tick}}.\overline{\underline{\underline{a@x}}}$$

$$[\![\ (a, in), n\ ]\!]''_R \stackrel{def}{=} \underline{in@n}(x).\underline{\underline{tick}}.\overline{\underline{\underline{a@x}}}$$

$$CLOCK \equiv !\ clock{:}3.\underline{\underline{worked}}.\overline{\underline{bcast}}\langle worked\rangle.\underline{\tau}.\overline{\underline{bcast}}\langle tick\rangle.\overline{clock{:}3} \mid$$
$$\mid \overline{clock{:}3} \mid \overline{\underline{\underline{worked}}}$$

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

Table 5.7: Encoding of catalytic P systems in $\pi@$

Part II

The Stochastic π@ calculus

Chapter 6

The Stochastic π@ Calculus

The Sπ@ language can be considered in first approximation as the stochastic version of the core-π@ introduced in Sect. 4.3.2, which is limited to two levels of priority and two names for each channel. The ability to give infinite rates to reactions [55] replaces the two priority levels of this core-π@, while the two names denoting each action assume different meanings, since the first represents the type of (chemical) reaction, while the second the compartment where the reaction takes place. The syntax is decorated with information of the *volume* occupied by each chemical species, which is used to calculate the effective rate of reactions inside each compartment: this allows a unique definition of each chemical reaction independently of the number of compartments, but it also requires a specific extension of the SSA to the multi-compartment setting. After a short introduction to the original Gillespie's simulation algorithm this extension is presented in Sect. 6.2, while in Sect. 6.3 the Sπ@ language is formalised.

6.1 Gillespie Stochastic Simulation Algorithm

The original SSA considers a fixed volume V containing a mixture of N chemical species $S_1, \ldots S_N$ interacting through M reaction channels $R_1, \ldots, R_M$. $X_1, \ldots, X_N$ represent respectively the number of molecules for each of the S_i chemical species. The state of the system at any time t is characterised by the state vector $X(t) =$

$\big(X_1(t), \ldots, X_N(t)\big) = x$ where $X_i(t)$ is the number of molecules of species S_i at the given time. $\nu_\mu = (\nu_{1\mu}, \ldots, \nu_{N\mu})$ is the *state change vector*, which contains all the information of the number and species of reactant molecules and reaction products for each reaction R_μ. For a reaction of the kind

$$R_\mu : \quad S_i + S_j \rightarrow S_k \qquad\qquad i \neq j$$

we have that $\nu_{i\mu} = \nu_{j\mu} = -1$, while $\nu_{k\mu} = +1$ and $\nu_{l\mu} = 0$ for every other index l. The probability that an R_μ reaction will occur inside V in the next infinitesimal time interval $(t, t + \mathrm{d}t)$ is calculated as

$$a_\mu(x)\,\mathrm{d}t = X_i(t)X_j(t)c_\mu\,\mathrm{d}t \tag{6.1}$$

where $c_\mu\,\mathrm{d}t$ is the probability that a particular molecular pair of the involved chemical species will react according to R_μ inside V in the next infinitesimal interval $(t, t + \mathrm{d}t)$ and $a_\mu(x)$ is the *propensity function* of R_μ. In general, $a_\mu(x)$ is calculated as

$$a_\mu(x) = h_\mu(x)c_\mu \tag{6.2}$$

where $h_\mu(x)$ represents the number of distinct R_μ molecular reactant combinations available at some time t inside V.

The dynamics of the system obeys the *chemical master equation* (CME) [44, 45, 16]

$$\delta P(x, t|x_0, t_0)/\delta t = \textstyle\sum_{\mu=1}^{M}[a_\mu(x - \nu_\mu)P(x - \nu_\mu, t|x_0, t_0) - a_\mu(x)P(x, t|x_0, t_0)] \tag{6.3}$$

where $P(x, t|x_0, t_0)$ is the probability that $X(t)$ will be x, given that $X(t_0) = x_0$. The CME is hard to solve except for very simple systems. The SSA provides a stochastic simulation method rigorously equivalent to the CME. Starting from an initial state, the SSA allows the system to evolve stochastically by providing the next state reached after a single firing of one of the M molecular reactions, chosen according to the CME. The aim of the algorithm is to find *when the next reaction*

fires (i.e. the value of the time variable τ) and *which of the M reactions it is* (the index μ of the next reaction R_μ).

The function $P(\tau, \mu|x)\,\mathrm{d}\tau$ represents the probability that, given the state x at some time t, the next reaction in V will occur in the infinitesimal time interval $(t+\tau, t+\tau+\mathrm{d}\tau)$ and will be an R_μ reaction. It can be calculated as the product of the probability $P(\tau|x)$ that, given the state x at some time t, no reaction will occur in the time interval $(t, t+\tau)$, times the probability $a_\mu(x)\,\mathrm{d}\tau$ that an R_μ reaction will occur in the time interval $(t+\tau, t+\tau+\mathrm{d}\tau)$:

$$P(\tau, \mu|x)\,\mathrm{d}\tau = P(\tau|x)a_\mu(x)\,\mathrm{d}\tau$$

Since $[1-\sum_j a_j(x)\,\mathrm{d}\tau]$ is the probability that no reaction will occur in time $\mathrm{d}\tau$ from the state x,

$$P(\tau+\mathrm{d}\tau|x) = P(\tau|x)\cdot[1-\sum_j a_j(x)\,\mathrm{d}\tau]$$

from which

$$P(\tau|x) = \exp(-\sum_{j=1}^{M} a_j(x)\tau)$$

Hence, the function $P(\tau, \mu|x)$ is given by

$$P(\tau, \mu|x) = \begin{cases} a_\mu(x)\exp(-a_0(x)\tau) & 0 \le \tau < +\infty, \\ & \mu = 1, \ldots, M \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

where $a_0(x) = \sum_{j=1}^{M} a_j(x)$.

In order to generate a random pair (τ, μ) according to Expr. (6.4) by a pair (z_1, z_2) obtained by a unit-interval uniform random number generator, the following resampling is evaluated in the SSA for τ

$$\tau = \frac{1}{a_0(x)}\log\frac{1}{z_1} \tag{6.5}$$

while μ is calculated as the smallest integer satisfying

$$\sum_{j=1}^{\mu} a_j(x) > z_2 a_0(x) \tag{6.6}$$

The SSA can be summarised as follows:

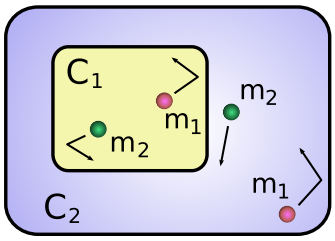


Figure 6.1: Two equivalent molecule pairs floating in compartments of different volume.

Algorithm 1 *(Gillespie Stochastic Simulation Algorithm)*

1. *calculate* $a_i, i = 1, \ldots, M$ *and* a_0 *from Expr.* (6.2)*;*
2. *generate uniformly two random numbers* z_1, z_2 *in the interval* $(0, 1)$*;*
3. *calculate* τ *from Expr.* (6.5) *and* μ *from Expr.* (6.6)*;*
4. *update the states of the species to reflect the execution of reaction* μ *and set* $t = t + \tau$*;*
5. *go to step* (1).

For details we refer to [45].

6.2 Stochastic simulation with multiple compartments

Gillespie's SSA [44, 45] constitutes one of the more exploited chemical abstractions for the effective simulation of bio-systems expressed in terms of concurrent calculi, but its original formulation is limited to systems composed of a single, fixed-size volume.

The most intuitive way to adapt the SSA to multiple compartments [26] is to consider each compartment as an isolated system evolving in parallel with the others, so that each compartment is implemented as an instance of the SSA. Even if

correct, the above approach is based on strong assumptions which severely limit its application.

In fact, the distinctive behaviour of biological systems is the continuous interaction of their constituting elements. In the case of multi-compartment systems, this behaviour introduces the necessity of modelling interaction between adjacent compartments, which consists of the interaction of their respective elements or exchange of material. A refined model should also be able to represent a dynamic compartment structure. When the simulation of compartments is obtained by independent instances of the SSA under its original assumptions, the interaction between compartments must obey the hypothesis of volume invariance but, for a biological liquid-state system, this means that the exchanged elements must be negligible with respect to the dimension of the whole compartments. This approximation can be tolerated while very few molecules are exchanged during the time of the simulation, but becomes unreasonable when the aim is the modelling of compartments with dynamic structure (i.e. the system is subject to complex structural changes like, for example, the movement, merging, splitting of compartments), a necessary step towards a satisfactory representation of biological systems.

Besides the limit of static compartment structure, the above approach presents another relevant drawback. Consider the system in Fig. 6.1, composed of two compartments with different volume. According to the model of the SSA, the molecules m_1 and m_2 have different probability of collision when floating in compartment C_1 compared with C_2. In fact, since C_2 is characterised by a larger volume, the probability to collide is lower than in C_1. Since the SSA requires the modeller to specify the reaction rate (more precisely, the "reaction probability per unit time") for each reaction, the same kind of reaction (like, for example, the one between m_1 and m_2) may be characterised by a different rate for each compartment. In other words, in the presence of K compartments, the same reaction rule (performed by the same kind of reactants, with the same physical properties) will have K different rates depending on the compartment the reaction is localised into, hence K different representations, one for each instance of the SSA.

The above approach has been usually adopted when providing a stochastic formulation of bio-oriented calculi equipped with explicit compartment semantics such as BioAmbients [87, 12], where the same reaction in different compartments may be associated with a unique rate only if all the volumes are assumed constant and equal, or if compartments are exploited at other abstraction levels (e.g. formation of protein complexes) such that they cannot be associated with any volume information. Some kind of parameterisation of reaction rules can be adopted to overcome the remarked drawback, but the result is almost useless if the dynamic nature of compartments in the calculus is taken into proper account.

We follow a different approach to the extension of the SSA to the multi-compartment model: here we show that by introducing in the model the information pertaining to the volumes, the SSA can be transparently adapted to the exact representation of multiple compartments. The extended model – multi-compartment SSA, MSSA for short – is then shown to be consistent with the kinetic hypotheses of the original SSA, but closer to the biological scenario and simpler with respect to the approach discussed above, since it allows us to give unique description of elements and reactions while keeping their rates consistent as a function of the enclosing compartment.

In order to find an expression for the probability of occurrence of each reaction in a given system, the original SSA [45] considers a gas-phase system of fixed volume V and, by simple kinetic deductions, calculates the probability $c_p\ \mathrm{d}t$ of collision of two molecules m_1 of species S_i and m_2 of species S_j (modeled as hard spheres of radii d_1 and d_2) in the infinitesimal time interval $(t, t + \mathrm{d}t)$ given by

$$c_p\ \mathrm{d}t = \overline{\mathrm{d}V_{coll}/V} = V^{-1}\pi d_{12}^2\overline{v_{12}}\ \mathrm{d}t$$

where $\mathrm{d}V_{coll}$ is the average "collision volume", $d_{12} = d_1 + d_2$ and $\overline{v_{12}}$ is the average relative velocity of the two molecules. In the same way, the probability $c_\mu\ \mathrm{d}t$ of *reactive collision* of two molecules (according to reaction R_μ) is expressed by

$$c_\mu\ \mathrm{d}t = c_p\, c_r\ \mathrm{d}t = V^{-1}\pi d_{12}^2\overline{v_{12}}\, c_r\, \mathrm{d}t$$

where c_r represents the probability that a given collision is actually reactive.

We can now introduce a constant r_μ, which depends only on the physical properties of the two molecules and the temperature of the system, such that

$$c_\mu \ \mathrm{d}t = V^{-1} \, r_\mu \ \mathrm{d}t$$

Furthermore, if X_i and X_j are the number of molecules of type S_i and S_j respectively, the probability $a_\mu \ \mathrm{d}t$ that an R_μ reaction will occur somewhere inside V in the infinitesimal time interval $(t, t + \mathrm{d}t)$ at the state $x = X(t)$ can be expressed as

$$a_\mu(x) \ \mathrm{d}t = X_i(t) X_j(t) c_\mu \ \mathrm{d}t = V^{-1} X_i(t) X_j(t) r_\mu \ \mathrm{d}t \tag{6.7}$$

The value a_μ captures all the information pertaining to the *concentration* of the reactants S_i and S_j. In the case of a single compartment of fixed volume, a_μ depends only on the quantity of reactants inside V, while in the case of multiple compartments or variable volume, expression (6.7) allows us to calculate the effective reaction rates as a function of the number of involved molecules inside each compartment and the volume of the compartment itself.

The effect of compartments is to *separate* the enclosed elements, that is to *prevent the interaction between elements placed in different compartments*. If m_{11}, m_{21} are two molecules of species S_i and S_j inside compartment C_1, and m_{12}, m_{22} are two molecules of species S_i and S_j inside compartment C_2, even if m_{11} may interact with m_{22} by an R_μ reaction, their collision is prevented because of the separation granted by compartment boundaries. Even if the pairs m_{11}, m_{21} and m_{12}, m_{22} may undergo the same reaction of type R_μ, the reaction R_μ of elements inside compartment C_1 is completely independent of the same reaction R_μ of elements inside compartment C_2. Therefore, in a multi-compartment environment, each *reaction channel* should be denoted not only by the *type* of reaction but also by the *compartment* the reaction happens in.

By following these intuitions, the SSA can be transparently adapted to fit the multi-compartment model without affecting its original kinetic hypotheses. This can be achieved by expressing the propensity function a_j of each reaction as a function of the (variable) volume V of the compartment. In the case of reactions fired by

two reactant molecules, we have:

$$a_\mu(x) = h_\mu(x)c_\mu = V^{-1}(x)h_\mu(x)r_\mu \tag{6.8}$$

The $r_\mu \, \mathrm{d}t$ value represents the probability that an R_μ reaction fires in the infinitesimal time $\mathrm{d}t$ inside a unit-size volume containing a single molecule pair undergoing reaction R_μ.

The price of this generalisation is the insertion, in the implementation of the model, of the information pertaining to the volume size of each compartment, which can be accomplished in different ways. The most immediate method would be the definition of a function $\mathrm{Vol} : \mathcal{C} \to \mathbb{R}$ returning the volume size of each compartment, but this would not allow us to model variable volumes.

The approach chosen here is to allow (but not require) the specification of the volume $v(S_j)$ for each molecule m of species S_i, which represents *the (average) increment of volume of a compartment C needed for including the additional element m*. In the case of constant volume, $v(S_i) = 0$ for each i. In fact, the additional volume needed to include any further element is 0 if the volume is constant. Conversely, at constant and homogeneous temperature and pressure, for a single chemical species S occupying the entire volume V, $v(S)$ can be calculated as

$$v(S) = \frac{V}{X} = \frac{m}{X} \cdot \frac{1}{D} = \frac{w_S}{D}$$

where X is the number of molecules of S inside V, m is the total mass, D is the density of S, w_S its molecular weight.

In an ideal system at constant and uniform temperature and pressure, thermal and chemical equilibrium, the volume $v(S_i)$ is constant and depends only on the species S. On the contrary, in a biological system at constant temperature and pressure, not in chemical, nor structural equilibrium, $v(S_i)$ is a function depending on the kind of the element S_i but also on the compartment C, since the chemical composition of C may vary the average distance between the inner elements, because of atomic-level forces like van der Waals interactions and hydrogen bonds. Under the assumption (tolerable if the discussed variation is reasonably small or the chemical

composition of the compartments is almost the same) of constant $v(S_i)$ for each i, the volume V of each compartment C can be easily calculated as the sum of $v(S_i)$ for each i inside C, even in the case of exchange of molecules or reorganisations in the compartment structure.

Hence, the volume $V(x)$ of the compartment at some time t in the state $x = X(t) = \big(X_1(t), \ldots, X_N(t)\big)$ can be calculated as the sum of the volumes occupied by each of the molecules located inside V. Given the function $v : \{S_1, \ldots, S_M\} \to \mathbb{R}$ which returns the volume occupied by one molecule of each chemical species, V is calculated as

$$V(x) = \sum_{j=1}^{M} v(S_j) X_j(t) \tag{6.9}$$

In the case of aeriform systems or systems composed of one (or few) chemical species, the function $v(S_j)$ can be easily estimated by knowing the molecular weight of the species and their density. In the case of real systems composed of thousands of different species, $v(S_j)$ may only be estimated by formulating a specific kinetic model.

In the presence of more than one compartment, each of the R_j reactions must be considered with respect to the compartment the reaction occurs in. This means that each R_μ reaction is characterised by C different propensity functions, one for each of the C compartments. In the case of reactions fired by two reactant molecules

$$a_\mu^k(x) = h_\mu^k(x) c_\mu^k = V_k^{-1}(x) h_\mu^k(x) r_\mu \qquad k = 1, \ldots, C \tag{6.10}$$

while for monomolecular (decay) reactions, which are independent of the volume

$$a_\mu^k(x) = h_\mu^k(x) c_\mu^k = X_\mu^k(t) r_\mu \qquad k = 1, \ldots, C \tag{6.11}$$

where

$$x = X(t) = \big(\mathbf{X}_1(t), \ldots, \mathbf{X}_C(t)\big), \quad \mathbf{X}_k(t) = \big(X_1^k(t), \ldots, X_N^k(t)\big) \tag{6.12}$$

Additional reactions which model interaction between distinct compartments (i.e. the presence of chemical products inside compartments different from the one where

the reaction initially fired) are still expressed by one of the above two kinds of reactions.

Each $X_j^k(t)$ represents the number of molecules of the S_j chemical species inside compartment k at time t. The value $a_\mu^k(x)\,\mathrm{d}t$ represents the probability that the reaction R_μ will happen inside compartment k in the next infinitesimal interval $\mathrm{d}t$. Each volume V_k is calculated as

$$V_k(x) = \sum_{j=1}^{M} v(S_j) X_j^k(t) \tag{6.13}$$

The value a_0 of Expr. (6.4) becomes

$$a_0(x) = \sum_{k=1}^{C} \sum_{j=1}^{M} a_j^k(x) = \sum_{k=1}^{C} a^k(x) \tag{6.14}$$

where $\sum_{j=1}^{M} a_j^k(x) = a^k(x)$. Expr. (6.6) is then unchanged:

$$\tau = \frac{1}{a_0(x)} \log \frac{1}{z_1} \tag{6.15}$$

where a_0 is calculated according to Expr. (6.14).

In order to identify both the compartment ψ and the reaction μ by a single generation of a unit-interval random number z_2, Expr. (6.6) is modified so that (ψ, μ) is the smallest pair of indexes satisfying

$$\sum_{k=1}^{\psi} \sum_{j=1}^{\mu} a_j^k(x) > z_2 a_0(x) \tag{6.16}$$

where $(\psi, \mu) < (\psi', \mu')$ according to the standard lexicographic ordering.

The multi-compartmental simulation algorithm (MSSA) can be finally expressed as a slight variation of the SSA:

Algorithm 2 *(Multi-compartmental Stochastic Simulation Algorithm)*

1. *calculate* a_i^k *with* $k = 1, \ldots, C, \quad i = 1, \ldots, M$ *from Expr.* (6.10), (6.11), (6.13) *and* a_0 *from Expr.* (6.14)*;*

2. *generate uniformly two random numbers* z_1, z_2 *in the interval* $(0, 1)$*;*

3. *calculate* τ *from Expr.* (6.15) *and* (ψ, μ) *from Expr.* (6.16)*;*

4. *update the states of the species to reflect the execution of reaction* (ψ, μ) *and set* $t = t + \tau$*;*

5. *go to step* (1)*.*

The original Gillespie's propensity functions are recovered when the system is composed of a single compartment of unitary volume: this may be achieved by setting $v(S_j) = 0\ \forall j$ and adding a fictitious element of volume 1 not participating in any reaction. Although immediate, this expedient seems quite artificial. A more faithful model would be obtained by specifying the total (not null) volume of the products of each reaction equal to the total volume of the respective reactants. Further elements not taking part in the reactions but influencing their rates (such as water, which may dilute reactants and slow down reactions even without direct chemical interaction) should also be specified, because they actually determine the total volume of the compartment.

It is worth remarking that the MSSA closely follows the original SSA master equation (6.3). As reported in [45], the key element of the master equation formalism is the "grand probability function"

$$\begin{aligned} P(x,t) \ \equiv\ & \text{probability that there will be, in the single volume } V \text{ of the system} \\ & \text{at time } t,\ X_1 \text{ molecules of species } S_1,\ X_2 \text{ of species } S_2,\ \ldots,\ X_N \text{ of} \\ & \text{species } S_N, \text{ with } x = X(t) = \big(X_1(t), \ldots, X_N(t)\big) \end{aligned}$$

whose knowledge would provide a complete characterisation of the system at any time. The evolution of the system is then considered in the discrete infinitesimal time interval $t + \mathrm{d}t$, in which at most one reaction occurs. Hence, given M different possible chemical reactions, the state $X(t + \mathrm{d}t)$ can be reached in $M + 1$ ways: either $X(t) = X(t + \mathrm{d}t)$ and no reaction occurs, or one of the M reactions actually happens. The grand probability function can be then expressed as follows:

$$\begin{aligned} P(x; t + \mathrm{d}t) \ =\ & P(x;t)\ P(\text{no reaction occurs over } \mathrm{d}t) \\ & + \textstyle\sum_{\mu=1}^{M} P(x - \nu_\mu; t)\ P(\text{reaction } R_\mu \text{ occurs over } \mathrm{d}t) \end{aligned} \tag{6.17}$$

where ν_μ is the state change vector associated with reaction R_μ, and

$$\begin{aligned} P(\text{no reaction occurs over } \mathrm{d}t) &= 1 - \sum_{\mu=1}^{M} a_\mu(x)\,\mathrm{d}t \\ P(\text{reaction } R_\mu \text{ occurs over } \mathrm{d}t) &= a_\mu(x-\nu_\mu)\,\mathrm{d}t \end{aligned}$$

in perfect agreement with Expr. (6.1), (6.2), (6.4). Expr. (6.17) becomes the CME (6.3) when $\mathrm{d}t$ tends to zero.

We can now follow the same steps in order to find the expression for the chemical master equation of the MSSA. Here, we have

$$\begin{aligned} P(x,t) \equiv\ & \text{probability that there will be in the system at time } t,\ X_i^k \\ & \text{molecules of each species } S_i \text{ inside each compartment } k, \\ & \text{with } x \text{ corresponding to Expr. (6.12) and} \\ & i = 1,\ldots,N \qquad k = 1,\ldots C \end{aligned}$$

The discrete evolution of the system to the state $X(t+\mathrm{d}t)$ is again expressed as a function of the $M \cdot C + 1$ ways it can be reached, where M is the number of possible reactions and C the number of compartments.

It is worth remarking that the number of possible reactions in the MSSA is even greater than $M \cdot C$, since an unbounded number of inter-compartment reactions can be introduced to model the exchange of molecules between compartments. In general, a system can be composed of

- M reactions describing the behaviour of chemical species inside any compartment, corresponding to $M \cdot C$ rules in the MSSA;
- M_{ic} inter-compartment reactions.

The total number of reactions in the system is then $M \cdot C + M_{ic}$. The reactants of each of these M_{ic} rules must be located in the same compartment. Consequently, it is ideally possible to place each inter-compartment rule inside a specific compartment in relation to the location of its reactants. For example, the simple inter-compartment rule

$$R : S_1^1 \rightarrow S_1^2$$

which moves a molecule of species S_1 from compartment C_1 to compartment C_2 can be thought of as being located inside compartment C_1, where its reactants reside. Although conceptually different, these inter-compartment rules can be treated as standard reactions: they are formally indistinguishable, except for the fact that the products in their state change matrix are not in the same row (or rows) as the reactants.

To be rigorous, we should then consider in general a distinct number of reactions $M_1, \ldots M_C$ for each of the C compartments. In particular, $M_j = M + M_{ic}^j$ where M_{ic}^j is the number of inter-compartment reactions whose reactants reside in C_j. However, since this level of detail does not bring any further insight, for the sake of readability we avoid to index explicitly inter-compartment reactions hereinafter.

We have that

$$\begin{aligned} P(x; t + \mathrm{d}t) \;=\; & P(x;t)\; P(\text{no reaction occurs over } \mathrm{d}t) \\ & + \textstyle\sum_{\mu,k} P(x - \nu^{k\mu}; t) \cdot \\ & \qquad P(R_\mu \text{ occurs in compartment } k \text{ over } \mathrm{d}t) \end{aligned} \tag{6.18}$$

with $\mu = 1, \ldots, M$ and $k = 1, \ldots, C$, while $\nu^{k\mu}$ is the state change vector of reaction R_μ firing inside compartment k and

$$\begin{aligned} P(\text{no reaction occurs over } \mathrm{d}t) &= 1 - \sum_{k=1}^{C} \sum_{\mu=1}^{M} a_\mu^k(x)\, \mathrm{d}t \\ P(R_\mu \text{ occurs in compartment } k \text{ over } \mathrm{d}t) &= a_\mu^k(x - \nu^{k\mu})\, \mathrm{d}t \end{aligned}$$

in agreement with Expr. (6.10), (6.14). When $\mathrm{d}t$ tends to zero, we obtain the CME for the MSSA:

$$\begin{aligned} & \delta P(x, t | x_0, t_0)/\delta t = \\ & \textstyle\sum_{k=1}^{C} \sum_{\mu=1}^{M} [a_\mu^k(x - \nu^{k\mu}) P(x - \nu^{k\mu}, t | x_0, t_0) - a_\mu^k(x) P(x, t | x_0, t_0)] \end{aligned} \tag{6.19}$$

which closely resembles the CME of the SSA.

A clever indexing, which separates the M_{ic} inter-compartment reactions from the M standard reactions, would give Expr. (6.19) almost the same shape as the reaction-diffusion master equation of the Next Subvolume Method [41], of which the CME of the MSSA constitutes a generalisation.

6.3 Sπ@ syntax and semantics

We formalise now the Sπ@ language.

Definition 6.1 *Let $\mathcal{N}, \mathcal{C}$ be distinct sets of names on a finite alphabet, with m, n ranging over $\mathcal{N}$, a, b over $\mathcal{C}$ and x, y over $\mathcal{X} = N \cup C$. Also let v range over $\mathbb{R}$ within the interval $]0, +\infty[$. The syntax of the Sπ@ language is defined as*

$$\begin{array}{rcl} P & ::= & \mathbf{0} \;\Big|\; \displaystyle\sum_{i \in I} \pi_i.P_i \;\Big|\; P \mid Q \;\Big|\; !\,\pi.P \;\Big|\; (\nu\, x)P \\ \pi & ::= & \tau_r \;\Big|\; n@a : v(\mathbf{x}) \;\Big|\; \overline{n}@a : v\langle \mathbf{x} \rangle \end{array}$$

where $\mathbf{x}$ represents zero or more names $x_1, \ldots, x_i$ ranging over $\mathcal{X}$.

The meaning of the above syntax closely follows that of the standard π-calculus:

- $\mathbf{0}$ is the null process, capable of doing nothing;
- $\sum_{i \in I} \pi_i.P_i$, written also $\pi_1.P_1 + \pi_2.P_2$ in the case $|I| = 2$, represents the guarded choice between different actions;
- $P \mid Q$ means that P and Q are two processes executing in parallel;
- $!\,\pi.P$ represents guarded replication, which allows the expression of recursive behaviour in π-like calculi;
- $(\nu\, x)P$ allows the scope restriction of the name x: the restriction of compartment names allows the creation of new compartments, while the restriction of reaction names is used in several ways, such as for representing bindings between different elements;
- τ_r represents an internal transition (silent action) characterised by exponential rate $r \in \mathbb{R} \cup \{\infty\}$.

The expressions $n@a : v(\mathbf{x})$ and $\overline{n}@a : v\langle \mathbf{x} \rangle$ represent respectively the polyadic input and output capabilities of a process, where

- n is the kind of reaction the process is ready to perform: in Expr. (6.10) it corresponds to the index μ denoting the reaction R_μ;
- a is the compartment where the reaction may take place, corresponding to k in Expr. (6.10);
- v corresponds to $v(S_\mu)$ in Expr. (6.13) and represents the volume occupied inside compartment a by the process ready to perform the input or output action.

The Sπ@ syntax allows the easy specification of processes which are located (and hence may occupy volume) in more than one compartment. For example, the process P

$$P \equiv n@a : v_1.Q_1 + m@b : v_2.Q_2 + p@a : v_3.Q_3$$

occupies some space both in compartment a and in compartment b. This capability allows us to represent neatly biological elements like transmembrane proteins, whose action takes place in the two compartments adjacent to the membrane or even in three compartments, if the membrane itself needs to be modelled as a compartment. [1] Since Sπ@ syntax does not force the association of a unique volume value with each action name, P may be written as well as

$$P \equiv n@a : v_{13}.Q_1 + m@b : v_2.Q_2 + p@a : 0.Q_3$$

with $v_{13} = v_1 + v_3$. In fact the volumes occupied in compartments a, b are the same in both cases. This kind of overloading may be avoided by changing the syntax of the choice operator, for example by specifying the volume occupied in each compartment in a list (associative array):

$$P \equiv [a : v_{13}, b : v_2]n@a.Q_1 + m@b.Q_2 + p@a.Q_3$$

[1] As long as the physical hypotheses of the SSA hold (random movement and stirring of molecules on the surface of membranes), the MSSA can handle both two- and three-dimensional compartments. In the case of two dimensions, as for membrane surfaces, the spatial information of elements would not represent their volume but their area.

Even if in this way the syntax is more rigorous, it loses readability, so Def. (6.1) is still preferable.

Definition 6.2 *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:*

$$\begin{aligned} (\nu\, x)P \mid Q &\equiv (\nu\, x)(P \mid Q) && \text{if } x \notin fn(Q) \\ (\nu\, x)P &\equiv P && \text{if } x \notin fn(P) \\ !\,\pi.P &\equiv \pi.(!\,\pi.P \mid P) && \text{if } \mathrm{fn}(\pi) \cap \mathrm{bn}(\pi) = \emptyset \end{aligned}$$

where the function fn *is defined as*

$$\begin{aligned} \mathrm{fn}(n@a\!:\!v(\mathbf{x})) &\triangleq \{n, a\} & \mathrm{fn}(\overline{n}@a\!:\!v\langle\mathbf{x}\rangle) &\triangleq \{n, a, \mathbf{x}\} \\ \mathrm{fn}(\mathbf{0}) = \mathrm{fn}(\tau_r) &\triangleq \emptyset & \mathrm{fn}((\nu\, x)P) &\triangleq \mathrm{fn}(P) \setminus \{x\} \\ \mathrm{fn}(\pi.P) &\triangleq \mathrm{fn}(\pi) \cup \mathrm{fn}(P) \setminus \mathrm{bn}(\pi) & \mathrm{fn}(\sum_{i\in I} \pi_i.P_i) &\triangleq \bigcup_i \mathrm{fn}(\pi_i.P_i) \\ \mathrm{fn}(P \mid Q) &\triangleq \mathrm{fn}(P) \cup \mathrm{fn}(Q) & \mathrm{fn}(!\,\pi.P) &\triangleq \mathrm{fn}(\pi.P) \end{aligned}$$

with $\mathrm{bn}(\pi) \triangleq \{\mathbf{x}\}$ *if* $\pi = @n()\!:\!avx$ *or* $\mathrm{bn}(\pi) \triangleq \emptyset$ *otherwise.*

Definition 6.3 *S$\pi@$ semantics is given in terms of the following reduction system:*

$$(S)\ \frac{r = \infty \ \vee \ M \overset{\infty}{\not\to} M'}{\tau_r.P + M \overset{r}{\to} P}$$

$$(C)\ \frac{\mathrm{rate}(n) = \infty \ \vee \ M \mid N \overset{\infty}{\not\to} S}{(n@a\!:\!v_1(\mathbf{x}).P + M) \mid (\overline{n}@a\!:\!v_2\langle\mathbf{y}\rangle.Q + N) \xrightarrow{\mathrm{rate}(n)} P\{\mathbf{y}/\mathbf{x}\} \mid Q}$$

$$(R)\ \frac{P \overset{r}{\to} P'}{(\nu\, x)P \overset{r}{\to} (\nu\, x)P'} \qquad (P)\ \frac{P \overset{r}{\to} P' \qquad r = \infty \ \vee \ P \mid Q \overset{\infty}{\not\to} S}{P \mid Q \overset{r}{\to} P' \mid Q}$$

$$(E)\ \frac{P \equiv Q \quad P \overset{r}{\to} P' \quad P' \equiv Q'}{Q \overset{r}{\to} Q'}$$

The rule (S) models the internal, silent transition of a process while the rule (C) allows the communication of the names $\mathbf{x}$ from process P to Q, where they are properly substituted to names $\mathbf{y}$. The function

$$\text{rate} : \mathcal{N} \to (\mathbb{R} \cup +\infty) \tag{6.20}$$

is an external function which permits us to associate the correct rate with each reaction, where the rate corresponds to the value r_μ of Expr. (6.10). Rules $(R), (P), (E)$ allow the transition of processes in the presence of restriction and of parallel operator, or by exploiting structural equivalence.

The introduction of the rates in the reduction relation of Def. 6.3 allows a more compact definition of Sπ@ semantics, with respect to Def. 4.5 of core π@. In fact, each rule in Def. 6.3 corresponds to two in Def. 4.5: for example, the precondition

$$r = \infty \qquad \vee \qquad M \stackrel{\infty}{\not\to} M'$$

on rule (S) allows the process $\tau_r.P$ to reduce if *either* of the following conditions hold:

- $r = \infty$, i.e. the rate r of the silent action is infinite, or
- $M \stackrel{\infty}{\not\to} M'$, that is any other reduction in the system is characterised by a finite rate.

In this way, infinite rate reductions are possible if any of the two conditions is true, while finite rate reductions can be fired only when the second one holds, that is when there is no infinite rate reduction in the system at that moment.

As for the π@ calculus, infinite-rate transitions can help the modelling of complex atomic operations but can also cause, if not correctly handled, an indefinite suspension of the time flow in the stochastic simulation, which corresponds to the hanging of the system.

Definition 6.4 *A Sπ@ system S is said to be in* standard form *if*

$$S = (\nu\, x)\big(P_1 \;\big|\; \cdots \;\big|\; P_j \;\big|\; !\, P_{j+1} \;\big|\; \cdots \;\big|\; !\, P_k\big)$$

and each P_i is a non-empty sum.

The standard form constitutes a more readable way to write (and an easier way to handle) systems in π-like calculi: restricted names are all collected on the left and replicated processes are listed after the non-replicated ones.

Proposition 6.1 *For every $S\pi@$ system S, there exists a system S' such that $S \equiv S'$ and S' is in standard form.*

In order to calculate the value $h_{\mu c}$ of Expr. (6.10) and (6.11), we introduce, according to [80], the function Act which permits us to know the number of possible combinations of inputs and outputs on a reaction channel inside a given compartment or the number of silent actions of a given rate.

Definition 6.5 *The activity* Act *of an action π is defined as*

$$\mathrm{Act}_\pi(S) = (\mathrm{In}_{n@a}(S) \cdot \mathrm{Out}_{n@a}(S)) - \mathrm{Mix}_{n@a}(S)$$

if $\pi = n@a$, corresponding to channel n inside compartment a in the system S, and

$$\mathrm{Act}_\pi(S) = \mathrm{Num}_{\tau_r}(S)$$

if $\pi = \tau_r$. S is in standard form, $\mathrm{In}_{n@a}(S)$ and $\mathrm{Out}_{n@a}(S)$ are the number of unguarded inputs and outputs on channel n inside compartment a, and $\mathrm{Mix}_{n@a}(S)$ is the sum of $\mathrm{In}_{n@a}(\sum_i) \cdot \mathrm{Out}_{n@a}(\sum_i)$ for each summation $\sum_i$ in S. $\mathrm{Num}_{\tau_r}(S)$ is the number of silent transitions of rate r in S.

For example, for the system

$$S \triangleq n@a.P_1 + \overline{m}@a.P_2 + m@a.P_3 \mid \overline{n}@a.Q \mid \overline{n}@a.R$$

we have three unguarded summations corresponding to the three parallel processes in S. The values of the functions previously defined are in this case

- $\mathrm{In}_{n@a}(S) = 1$, for the input guard before P_1;
- $\mathrm{Out}_{n@a}(S) = 2$, for the output guards before Q, R;

- $\mathrm{Mix}_{n@a}(S) = 0$, since there is no summation with both input and output actions on $n@a$;
- $\mathrm{Act}_{n@a}(S) = \mathrm{In}_{n@a}(S) \cdot \mathrm{Out}_{n@a}(S) - \mathrm{Mix}_{n@a}(S) = 2$.

For $m@a$ we have instead

- $\mathrm{In}_{m@a}(S) = 1$, for the input guard before P_3;
- $\mathrm{Out}_{m@a}(S) = 1$, for the output guards before P_2;
- $\mathrm{Mix}_{m@a}(S) = 1$, since the input and output actions on $m@a$ appear both in the first summation;
- $\mathrm{Act}_{m@a}(S) = \mathrm{In}_{m@a}(S) \cdot \mathrm{Out}_{m@a}(S) - \mathrm{Mix}_{m@a}(S) = 0$.

The activities on $n@a$ and $m@a$ are then 2 and 0 respectively, in agreement with the intuition that the two possible reductions on $n@a$ increase the potential activity on this channel, while on $m@a$ no reduction can be performed.

The function chan returns all the active channels inside each compartment in a given system S.

Definition 6.6 *Given a S$\pi@$ system S in standard form*

$$S = (\nu\, x)\big(P_1 \mid \cdots \mid P_j \mid !\, P_{j+1} \mid \cdots \mid !\, P_k\big)$$

the function chan *is defined recursively as follows:*

$$\begin{aligned}
\mathrm{chan}(S) &= \bigcup_{i=1}^{k} \mathrm{chan}(P_i) \\
\mathrm{chan}(\sum_{i\in I} \pi_i.P_i) &= \bigcup_{i\in I} \mathrm{chan}(\pi_i) \\
\mathrm{chan}(n@a : v(\mathbf{x})) &= \{n@a\} \\
\mathrm{chan}(\overline{n}@a : v\langle \mathbf{x}\rangle) &= \{n@a\} \\
\mathrm{chan}(\tau_r) &= \{\tau_r\}
\end{aligned}$$

Definition 6.7 *Given a Sπ@ system S in standard form*

$$S = (\nu\ x)\big(P_1 \mid \cdots \mid P_j \mid !\,P_{j+1} \mid \cdots \mid !\,P_k\big)$$

the volume Vol_a *of the compartment a in the system S is calculated as follows:*

$$\begin{aligned}
\mathrm{Vol}_a(S) &= \sum_{i=1}^{k} \mathrm{Vol}_a(P_i) \\
\mathrm{Vol}_a(\sum_{i\in I} \pi_i.P_i) &= \sum_{i\in I} \mathrm{Vol}_a(\pi_i) \\
\mathrm{Vol}_a(\tau_r) &= 0 \\
\mathrm{Vol}_a(\overline{n}@a\,{:}\,v\langle\mathbf{x}\rangle) &= \mathrm{Vol}_a(n@a\,{:}\,v(\mathbf{x})) \\
\mathrm{Vol}_a(n@b\,{:}\,v(\mathbf{x})) &= \begin{cases} v & a = b \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

If $\mathrm{Vol}_a(S) = 0$*, then a is given the default volume value* 1*.*

Each molecule is represented by a choice $\sum_{i\in I} \pi_i.P_i$, which may occupy volume in more than one compartment: Vol_a considers only that part of the molecule falling inside compartment a.

The default volume value of 1 is introduced as syntactic facility to allow the omission of any volume information when not required by the model. The main drawback of this choice is that it is not possible to catch the undesirable situation of compartments whose volume assumes value zero at some point during the simulation, a situation which probably reflects some error in the Sπ@ model. However, a smart implementation of the language may still spot this occurrence at runtime and notify it to the user.

The definition of Vol_a reflects the additive properties of Def. (6.13): its linear dependence on molecular volumes allows its almost immediate calculation in terms of modification of its value at the previous step of the algorithm. A definition of compartment volumes as non-linear functions of molecular volumes, or any dependence on molecular volumes upon the chemical composition of the surrounding

environment inhibits such additive expression of Vol_a and increases significantly the computational cost of its calculation at each step of the simulation.

The following algorithm corresponds to each repetition of the loop of Alg. (2).

Algorithm 3 *Given a S$\pi@$ system S in standard form, the selection of the next reaction* $\mathrm{Next}(S)$ *and of the delay* $\mathrm{Delay}(S)$ *relative to the MSSA are described by the following algorithm:*

1. *For each channel c_i in* $\mathrm{chan}(S)$*, with* $\mathrm{chan}(S) = \{c_1, \ldots, c_j\}$*, calculate*

$$a_i = \mathrm{Act}_{n@b}(S) * \mathrm{rate}(n) / \mathrm{Vol}_b(S)$$

   if $c_i = n@b$ for some $n \in \mathcal{N}, b \in \mathcal{C}$ or

$$a_i = \mathrm{Act}_{\tau_r}(S) * r$$

   if $c_i = \tau_r$.

2. *Calculate $a_0 = \sum_{i=1}^{j} a_i$*

3. *Generate two random numbers $z_1, z_2 \in [0, 1]$ and calculate τ, λ such that*

$$\tau = (1/a_0) \ln(1/z_1) \qquad \sum_{i=1}^{\lambda-1} a_i < z_2 a_0 \leq \sum_{i=1}^{\lambda} a_i$$

4. $\mathrm{Next}(S) = c_\lambda$ *and* $\mathrm{Delay}(S) = \tau$.

The value $c_\lambda = \tau_r$ for some r or $c_\lambda = n@b$ for some n, b denotes the rate of the silent action or the reaction channel n (corresponding to μ in Alg. (2)) and the compartment b (corresponding to ψ in Alg. (2)) of the next reaction happening after τ time. The process performing the silent transition or the two processes performing the synchronisation step on c_λ are then randomly chosen as for SPiM [80].

6.4 Efficient formulation of the simulation algorithm

In the original formulation of the SSA [45] each transition of the system from one state x to a subsequent state x' requires at most M operations needed to know which of the M reactions will happen next, as a function of the random number generated in the second step of the algorithm. In fact, in order to find the index μ in Expr. (6.6), M summations are required in the worst case. Several improvements or alternatives to the SSA have been proposed (e.g. [43, 16, 41]) which reduce the computational complexity of each transition to $O(\log M)$ or optimise it as a function of reaction rates.

The MSSA inherits the complexity order of the original SSA, linear in the number of distinct reactions. However in this case the number of independent reactions is $M \cdot C$, where C is the number of compartments. As noted in [41], the simulation becomes computationally unfeasible if C grows significantly. Unfortunately, none of the proposed improvements can be directly applied to the MSSA with appreciable gain. The main reason is that the propensity function a_j^k of each bimolecular reaction depends of the volume V_k of the enclosing compartment. Each reaction firing may change the volume of one or more compartments, so that all the propensity functions of bimolecular reactions located into the involved compartments should be recalculated. This would cause the complexity of the algorithm to be still linear in the number of the M reactions even after the optimisations proposed in [43, 41]. Also the optimised direct method (ODM) formulated in [16] would provide no substantial gain in the (not unusual) case that the chemical composition of the compartments is almost the same: in this situation the complexity would be almost linear in the number of compartments in the best case.

Nevertheless, the computational complexity order of the MSSA can be reduced to $O(\log M + \log C)$ by exploiting the same data structures proposed in [43] for enhancing directly the SSA. These structures are justified by two observations. The first is that only few propensity functions change at each transition and these can

be easily identified by building a dependency graph. The second is that the linear search in step (3) can be improved by exploiting (twice) a binary search tree. The definitions of the needed data structures follow.

Definition 6.8 *Let $\nu^{k\mu}$ be the state change vector of reaction μ firing inside compartment ψ, $\nu^{\psi\mu} = (\nu_1^{\psi\mu}, \ldots, \nu_C^{\psi\mu})$, with $\nu_j^{\psi\mu} = (\nu_{j1}^{\psi\mu}, \ldots, \nu_{jN}^{\psi\mu})$.*

Let $\mathrm{Re}(R_\mu) \subseteq \{S_1, \ldots, S_N\}$ be the chemical species needed to fire reaction R_μ.

Let $G(V_G, E_G)$ be a directed graph with vertex set $\{0, \ldots, (C \cdot M) - 1\}$. For each vertex pair $(n, n'), n = f(\psi, \mu), n' = f(\psi', \mu')$, where $f(\psi, \mu) = (\psi * M + \mu)$, the edge $e = (n, n')$ is in E_G iff $\exists j \in \{1, \ldots, N\} : S_j \in \mathrm{Re}(R_{\mu'}) \wedge \nu_{\psi' j}^{\psi\mu} \neq 0$.

G represents the dependency graph of the system. Each vertex n of G represents a reaction R_μ inside a compartment ψ. Every edge from n to n' indicates that reaction R_μ inside ψ influences reaction $R_{\mu'}$ inside ψ' by changing the concentration in ψ' of at least one of the reactants of $R_{\mu'}$. The dependency graph evidences the only propensity functions which need to be updated after each transition.

We now define the structure of *non-cumulative complete binary search tree*. This definition represents the formalisation of the data structure proposed in [43] for the enhancement of the SSA as alternative of the Next Reaction Method.

Definition 6.9 *A binary tree T is recursively defined as* **0** *(the empty tree) or (n, T_l, T_r) where $n = (v, D)$ is the node, $v \in \mathbb{R}^+$ is its value and D the associated data, T_l and T_r are binary trees.*

A binary tree T is complete *if it is empty, or if all its levels are full, except for the last which presents all the remaining leaves on the left hand side.*

A complete binary tree T is a non-cumulative binary search tree (NCBST) if it is empty, or if $T = ((v, D), T_l, T_r)$ and

- *v corresponds to the sum of the values of the root nodes of T_l and T_r;*
- *T_l and T_r are NCBSTs as well.*

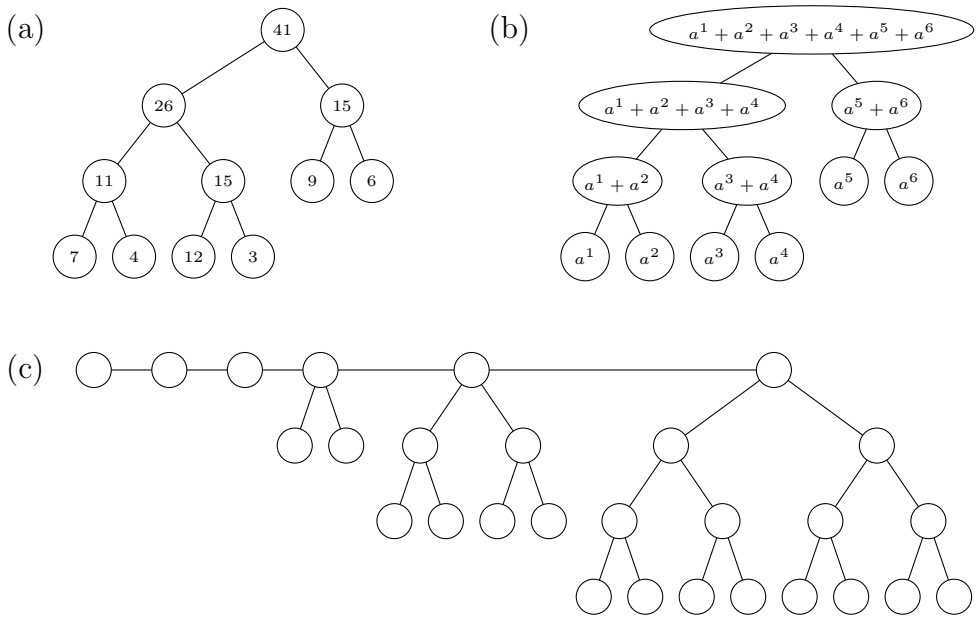


Figure 6.2: (a) (b) Non-cumulative complete binary search trees. (c) List of non-cumulative complete binary search trees with increasing heights.

A binary tree is a non-cumulative search tree if each node value is equal to the sum of the values of its offspring. The definitions are illustrated in Fig. 6.2. A non-cumulative binary search tree can be transformed into a binary search tree by recursively adding the value of each non-leaf node to all the nodes of its right subtree. In the NCBST of Fig. 6.2 (a), the value 41 of the root should be added to the three nodes on its right, with values 15, 9, 6. In this same subtree, the value 15 should be added to the leaf with value 6. In the same way, the values 26, 11, 15 of the non-leaf nodes in the left subtree should be added to their respective right subtrees.

We define now the function which implements the search in a NCBST.

Algorithm 4 *Let $T = ((v, D), T_l, T_r)$ be a non-empty NCBST and $p \in \mathbb{R}, p \in [0, v[$, with T, p formal parameters of the function:*

1. *if $T_l = T_r = \mathbf{0}$ (i.e. T is a leaf) then return $(p/v, D)$, else*

2. *let $T_l = ((v^l, D), T_l^l, T_r^l)$; if $p < v^l$ then set $T \leftarrow T_l$ and go to (1), else*

3. set $p \leftarrow (p - v^l), T \leftarrow T_r$ *and go to* (1).

Given a NCBST $T = ((v, D), T_l, T_r)$ and a random number $p = r \cdot v$, with $r \in [0, 1[$, the function returns the data associated with the leaf i, where

$$\sum_{j<i}(\text{leaf j value}) < p < \sum_{j\leq i}(\text{leaf j value})$$

and leaves are numbered from left to right, as in Fig. 6.2 (b). The function returns also the remainder of p scaled to the interval $[0, 1[$ (denoted in the algorithm by p/v in the first step), which avoids the generation of a further random number when the algorithm is called for the second time in the enhanced MSSA. A short example of execution follows.

Example 6.1 Consider the tree in Fig. 6.2 (a). Let $p = 20$. The root node is not a leaf, so it must be checked if $p < v_l = 26$, where v_l is the value associated with the left sub-tree. The condition is true, so the loop cycle must be repeated starting from T_l. The value of the left sub-sub-tree is $v_{ll} = 11 < p$, hence $p \leftarrow 20 - 11$ and the search continues in T_{lr}. Now $p = 9 < v_{lrl} = 12$ and T_{lrl} is a leaf, so the algorithm ends and returns $(0.75, D)$. □

The computational complexity of the algorithm is $O(\log K)$, where K is the number of nodes of the tree. Given a list of l non-negative real numbers, it is also possible to build a corresponding NCBST in $O(l)$ with all the elements of the list appearing as leaves of the tree. This is the technique exploited for executing step (3) of the MSSA in logarithmic time.

We can now define the improved variant of the MSSA. We first consider reindexing of monomolecular and bimolecular reactions, and also distinct expressions of their propensity functions, in order to keep them separated according to their (in)dependence on compartment volumes:

$$a_0(x) = m_0(x) + b_0(x) = \sum_{k=1}^{C} m^k(x) + \sum_{k=1}^{C} b^k(x) \tag{6.21}$$

where

- m_0 represents the sum of the propensity functions of monomolecular reactions;
- b_0 is the sum of the propensity functions of bimolecular reactions;
- M_m corresponds to the number of monomolecular reactions R_μ, with $\mu = 1, \ldots, M_m$;
- M_b corresponds to the number of bimolecular reactions R_μ, with $\mu = M_m + 1, \ldots, M$ so that $M_m + M_b = M$;
- m^k and b^k (with $k = 1, \ldots, C$) are the partial sums of the propensity functions of monomolecular and bimolecular reactions respectively, inside each of the k compartments.

The values a^k in Expr. (6.14) for bimolecular reactions can be written as

$$\begin{aligned} b^k(x) = \sum_{j=M_m+1}^{M} b_j^k(x) \quad &= \sum_{j=M_m+1}^{M} V_k^{-1}(x) h_j^k(x) r_\mu \\ = V_k^{-1}(x) \sum_{j=M_m+1}^{M} h_j^k(x) r_\mu \quad &= V_k^{-1}(x) \beta^k(x) \end{aligned} \tag{6.22}$$

where

$$\beta^k(x) = \sum_{j=M_m+1}^{M} h_j^k(x) r_\mu = \sum_{j=M_m+1}^{M} \beta_j^k(x) \tag{6.23}$$

with

$$\beta_j^k(x) = h_j^k(x) r_\mu \tag{6.24}$$

Furthermore, the summation of Expr. (6.16) for bimolecular reactions can be expressed as

$$\begin{aligned} \sum_{k=1}^{\psi} \sum_{j=M_m+1}^{\mu} b_j^k(x) = &\sum_{k=1}^{\psi-1} b^k(x) + \sum_{j=M_m+1}^{\mu} b_j^\psi(x) = \\ &\sum_{k=1}^{\psi-1} b^k(x) + V_\psi^{-1} \sum_{j=M_m+1}^{\mu} \beta_j^\psi(x) \end{aligned} \tag{6.25}$$

For monomolecular reactions, we have simply that

$$m^k(x) = \sum_{j=1}^{M_m} m_j^k(x) = \sum_{j=1}^{M_m} X_j^k(t) r_\mu \tag{6.26}$$

The enhanced MSSA (EMSSA) is defined as follows.

Algorithm 5 *(Enhanced Multi-compartmental Stochastic Simulation Algorithm)*

1. *calculate* V_k *from Expr.* (6.13), β_j^k *from Expr.* (6.24), β^k *from Expr.* (6.23), b^k *from Expr.* (6.22), m^k *from Expr.* (6.26), *with* $k = 1, \ldots, C, \quad j = M_m + 1, \ldots, M$, *and* b_0, m_0, a_0 *from Expr.* (6.21)*;*

2. *build the dependency graph of the system according to Def.* (6.8)*;*

3. *build the NCBST* T_b *such that its leaves are* $((b^k, k), \mathbf{0}, \mathbf{0})$, *then build the NCBST* T_m *such that its leaves are* $((m_j^k, j + (k-1) * M_m), \mathbf{0}, \mathbf{0})$, *with* $k = 1, \ldots, C$ *and* $j = 1, \ldots, M_m$*;*

4. *build* C *NCBSTs such that* $((\beta_j^k, j), \mathbf{0}, \mathbf{0})$ *are the leaves of the* k*-th NCBST* T^k, *with* $j = M_m + 1, \ldots, M$ *and* $k = 1, \ldots, C$*;*

5. *generate uniformly two random numbers* z_1, z_2 *in the interval* $(0, 1)$*;*

6. *calculate* τ *from Expr.* (6.15) *and set* $t = t + \tau$*;*

7. *if* $a_0 z_2 < m_0$ *then go to step* (13)*;*

8. *set* $z_2 \leftarrow (a_0 z_2 - m_0)/b_0$*;*

9. *let* (v, D) *be the value returned by Alg.* (4) *called with parameters* $(T_b, b_0 z_2)$*; set* $\psi \leftarrow D$, *according to Expr.* (6.25)*;*

10. *let* (v', D) *be the value returned by Alg.* (4) *called with parameters* $(T^\psi, \beta^\psi \cdot v)$*; set* $\mu \leftarrow D$, *according to Expr.* (6.25)*;*

11. *update the states of the species and* $T_m, T_b, T^1, \ldots, T^C$ *to reflect the execution of the bimolecular reaction* (ψ, μ) *by updating only the propensity functions, volumes and sub-trees indicated by the dependency graph built at step* (2)*;*

12. *go to step* (5).

13. *set* $z_2 \leftarrow a_0 z_2 / m_0$;

14. *let* (v, D) *be the value returned by Alg.* (4) *called with parameters* $(T_m, m_0 z_2)$; *set* $\psi \leftarrow d_1$ *and* $\mu \leftarrow d_2$, *where* $d_2 + (d_1 - 1) * M_m = D$;

15. *update the states of the species and* $T_m, T_b, T^1, \ldots, T^C$ *to reflect the execution of monomolecular reaction* (ψ, μ) *by updating only the propensity functions, volumes and sub-trees indicated by the dependency graph built at step* (2);

16. *go to step* (5).

The initialisation of the algorithm includes the building of the dependency graph and of *three kinds* of NCBSTs. The first, constituted by T_m, is the tree of all the monomolecular reactions of the system. The second, T_b, is the only one that contains information about the V_k volumes of the compartments. The third kind, represented by $\{T^1, \ldots, T^C\}$, contains the information about the propensity functions of the bimolecular reactions inside each compartment. This multi-tree organisation allows us to express the propensity functions independently of their respective volumes, so that they do not need to be recalculated as the volumes change.

The height of T_b is $\lceil \log(2 \cdot C) \rceil$, while the height of each T^k is $\lceil \log(2 \cdot M) \rceil$. The search of steps (9) and (10) are consequently executed in $O(\log C)$ and $O(\log M)$ respectively. The same cost of $O(\log C + \log M)$ applies to the search in T_m. The most expensive operations are steps (11) and (15): if Max_V is the maximum number of compartments influenced by some reaction R_μ and Max_R is the maximum number of propensity functions modified by some reaction R'_μ, the number of operations is bounded by $(\mathrm{Max}_V \log C + \mathrm{Max}_R \log M)$, because each update of the leaf of a NCBST T requires a number of updates of the ancestor nodes proportional to the height of the tree. Since in most cases $\mathrm{Max}_V \ll C$ and $\mathrm{Max}_R \ll M$, the computational complexity of the algorithm is usually $O(L(\log C + \log M))$, where L is the number of transitions of the systems (in practice limited by the detection of some steady

state condition, or by the availability of time of the user, as for the SSA), each corresponding to one reaction firing and one execution of the loop in Alg. (5).

It is worth remarking that the introduction of infinite-rate reactions breaks the correspondence between execution steps of the algorithm and temporal evolution of the simulated system. In other words, the above complexity analysis considers as *transitions* both the *time-elapsing steps* (caused by finite-rate reactions) and the *immediate steps* (corresponding to infinite-rate reactions), even if only the first kind produces some observable temporal progression in the simulation.

6.4.1 Further enhancements

The number of operations to perform step (11) can be considerably reduced by grouping together the indexes of the compartments and reactions whose volumes and propensity functions change simultaneously. This can be achieved by a proper analysis of the dependency graph of the system and may lead in the best case to $(2\,\mathrm{Max}_V + 2\,\mathrm{Max}_R + \log C + \log M)$ operations. Steps (9), (10) and (13) can be improved by adapting the enhancements to the SSA discussed in [16] to NCBSTs. The NCBST structure may be changed as shown in Fig. 6.2 (c). Here the leaves of NCBSTs of increasing height linked in a list contain the propensity function values in increasing order of firing frequency of the corresponding reactions. The frequencies can be calculated by previous benchmarking, as in [16].

Chapter 7

Implementation

The multi-compartment stochastic simulation algorithm previously described has been implemented as a prototype tool (the Multi-compartment Stochastic Simulator, MSS) written in Java [47]. The tool supports the Systems Biology Markup Language (SBML) [52] and can be integrated into the Systems Biology Workbench (SBW) [51], an open source framework which allows the interconnection of heterogeneous software applications oriented to the analysis and simulation of biochemical and biological models.

In the following we discuss the main features of the MSS and its integration with SBML and SBW. The chapter is structured as follows. First a short introduction to SBML is given in Sect. 7.1, in order to provide the reader with the minimal background necessary to follow the description of how SBML has been extended for the storage of all the quantitative information needed by the MSSA. Then a short description of the SBW with few details about its architecture is presented in Sect. 7.2. Finally, in Sect. 7.3 the MSS is described.

7.1 The Systems Biology Markup Language

The Systems Biology Markup Language is a machine-readable format for the representation of biochemical and biological models: it is formalised through an XML [11] schema and allows the description of models characterised by an unlimited number

of chemical species obeying standard chemical reactions rules of the form

$$R: \quad m_1R_1 + m_2R_2 + \ldots + m_iR_i \quad \xrightarrow{f} \quad n_1P_1 + n_2P_2 + \ldots + n_jP_j$$

where $R_1, \ldots, R_i$ stand for the reactants, $P_1, \ldots, P_j$ for the products, $m_1, n_1, \ldots$ for their stoichiometric coefficients and f represents a reaction law with arbitrary kinetic which can be totally specified in SBML.

Each model definition consist of several sections, each one devoted to the listing of the different entities of interest (e.g. compartments, species, reactions, functions, events and so on):

```
<model id="SingleReaction" ... >
    <listOfCompartments>
    ...
    </listOfCompartments>
    <listOfSpecies>
    ...
    </listOfSpecies>
    <listOfReactions>
    ...
    </listOfReactions>
</model>
```

Each list of entities contains all the elements of the same kind that appear in the model, each one described by optional attributes. The list of compartments, for example, contains all the compartments, each one characterised by qualitative or quantitative attributes such as name (or identifier) and size:

```
<listOfCompartments>
   <compartment name="Comp" size="1">
</listOfCompartments>
```

The list of species contains the information about each species in the model, with all the related values of interest (identifier, initial concentration, identifier of the compartment where the species is located, and so on):

```
<listOfSpecies>
  <species compartment="Comp" id="r"
           initialConcentration="10" />
    ...
</listOfSpecies>
```

The description of each reaction usually contains several subsections:

- the list of its reactants (with respective stoichiometry);
- the list of products (with stoichiometry as well);
- an additional section which specifies the kinetic law of the reaction by means of a MathML [24] element.

For example, a simple reaction of the kind $R : r \rightarrow p$ denoted by mass-action kinetics with rate k may be written as:

```
<reaction id="R">
  <listOfReactants>
    <speciesReference species="r" stoichiometry="1" />
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="p" stoichiometry="1" />
  </listOfProducts>
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times />
        <ci> k </ci>
        <ci> r </ci>
      </apply>
    </math>
  </kineticLaw>
</reaction>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml level="2" version="1" xmlns="http://www.sbml.org/sbml/level2">
  <model id="SingleReaction" name="SingleReaction">
    <listOfCompartments> <compartment id="Comp" size="0" /> </listOfCompartments>
    <listOfSpecies>
      <species boundaryCondition="false" compartment="Comp" id="r" initialConcentration="10" />
      <species boundaryCondition="false" compartment="Comp" id="p" initialConcentration="0" />
    </listOfSpecies>
    <listOfParameters>   <parameter id="k" value="0.5" />   </listOfParameters>
    <listOfReactions>
      <reaction id="R" reversible="false">
        <listOfReactants   <speciesReference species="r" stoichiometry="1" />   </listOfReactants>
        <listOfProducts>   <speciesReference species="p" stoichiometry="1" />   </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>      <times />     <ci> k </ci>    <ci> r </ci>
            </apply>
          </math>
        </kineticLaw>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

Figure 7.1: Systems Biology Markup Language: representation of a simple model composed of a single reaction $R : r \rightarrow p$ with standard mass-action kinetic law.

Each element can contain an optional section *annotation*, which allows the insertion of arbitrary information about the element:

```
<species compartment="Comp" id="r"
         initialConcentration="10">
    <annotation>
    ...
    </annotation>
</species>
```

This peculiarity will turn out to be useful in order to store all the information needed by the MSSA for the stochastic simulation with variable compartment volumes.

A simple but complete example of SBML model specification is given in Fig. 7.1.

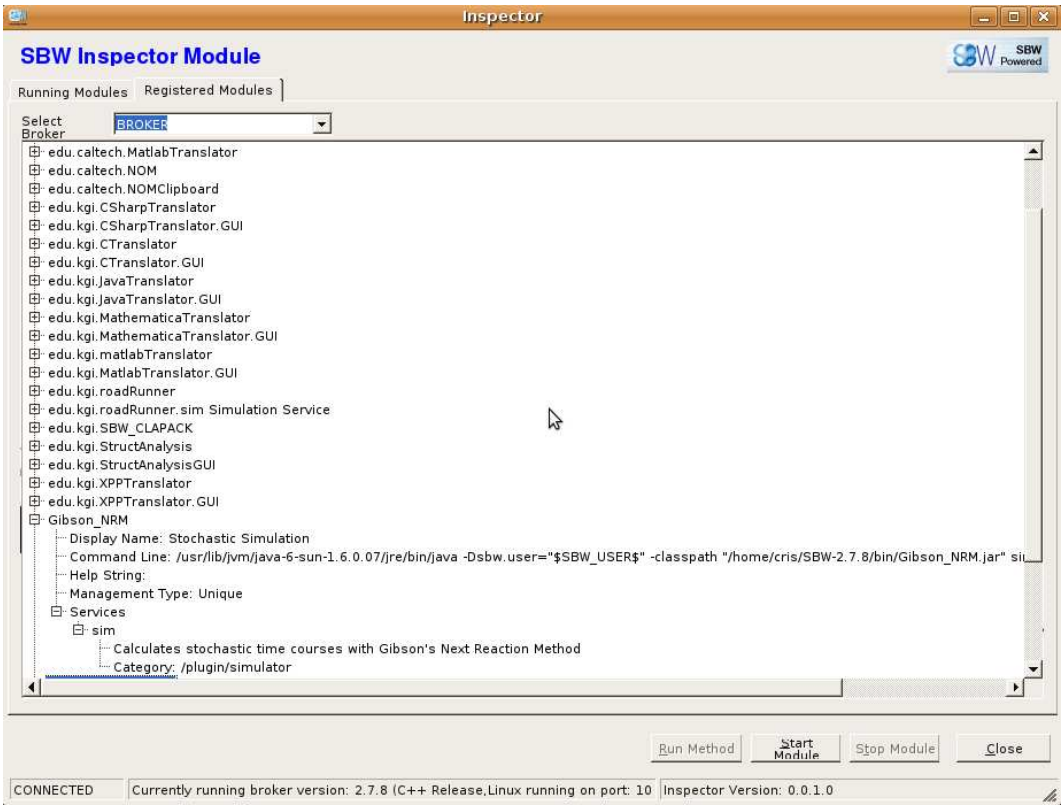


Figure 7.2: Screenshot of the SBW Broker, the central application of the framework. In the picture, the list of registered applications (modules) and their services.

7.2 The Systems Biology Workbench

The Systems Biology Workbench is an open source, multi-platform framework which supports the interaction of heterogeneous applications, written in different programming languages (C/C++, Java, Perl, Delphi, ...). SBW provides a core infrastructure providing a basic application programming interface (API) through which the *SBW-enabled* applications can cooperate. The modularity of the framework is achieved by the abstraction of *service*: each application provides one or more services, and every application can query the SBW Broker (the central process of the framework, which keeps track of all the applications and services, as shown in Fig. 7.2) in order to know which services are provided by other applications.

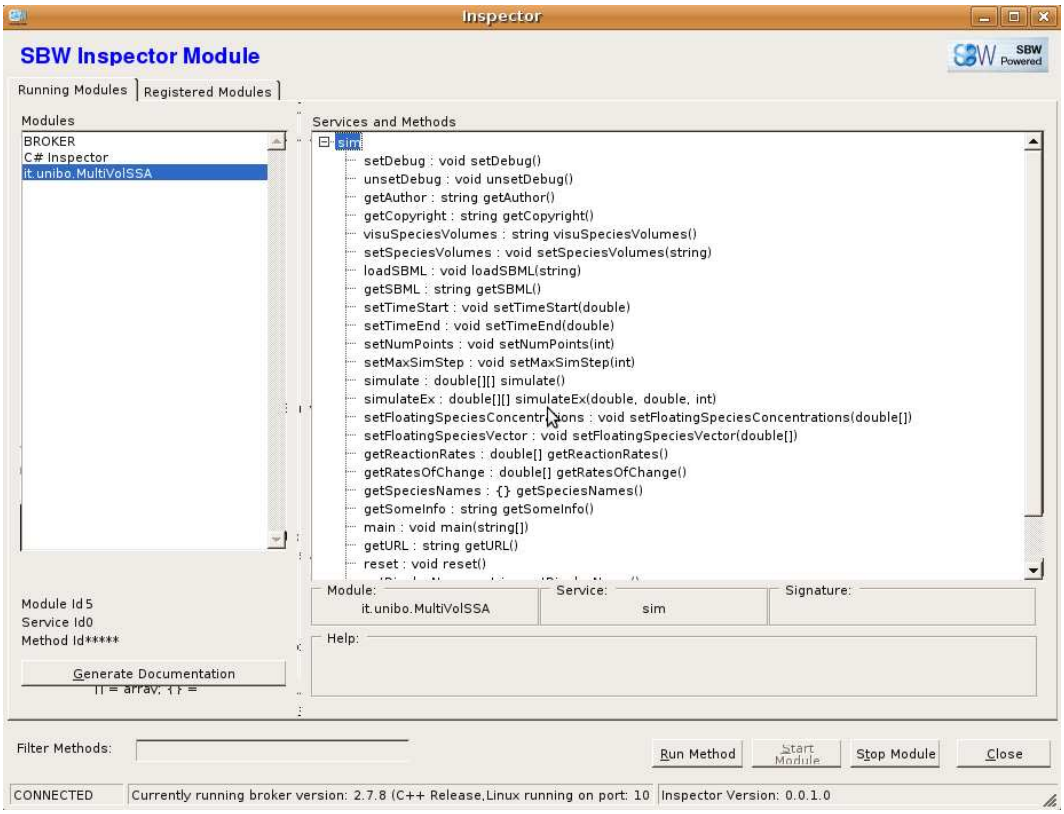


Figure 7.3: Visualisation in the SBW Broker of the service and methods provided by the Multi-compartment Stochastic Simulator.

The same service can be provided (in different ways) by one or more applications. For example, the *simulation* service consist in the possibility of loading arbitrary SBML code and retrieve the result of its related simulation. No constraints are given on the *kind* of simulation, so that it may consist of any kind of stochastic or deterministic simulation, performed with any algorithm.

This modularity helps the reuse of code, since new features can be added (such as new simulation algorithms) without the need to update the remaining part of the framework (for example post-processing or graphics tools for the elaboration and visualisation of the simulation data).

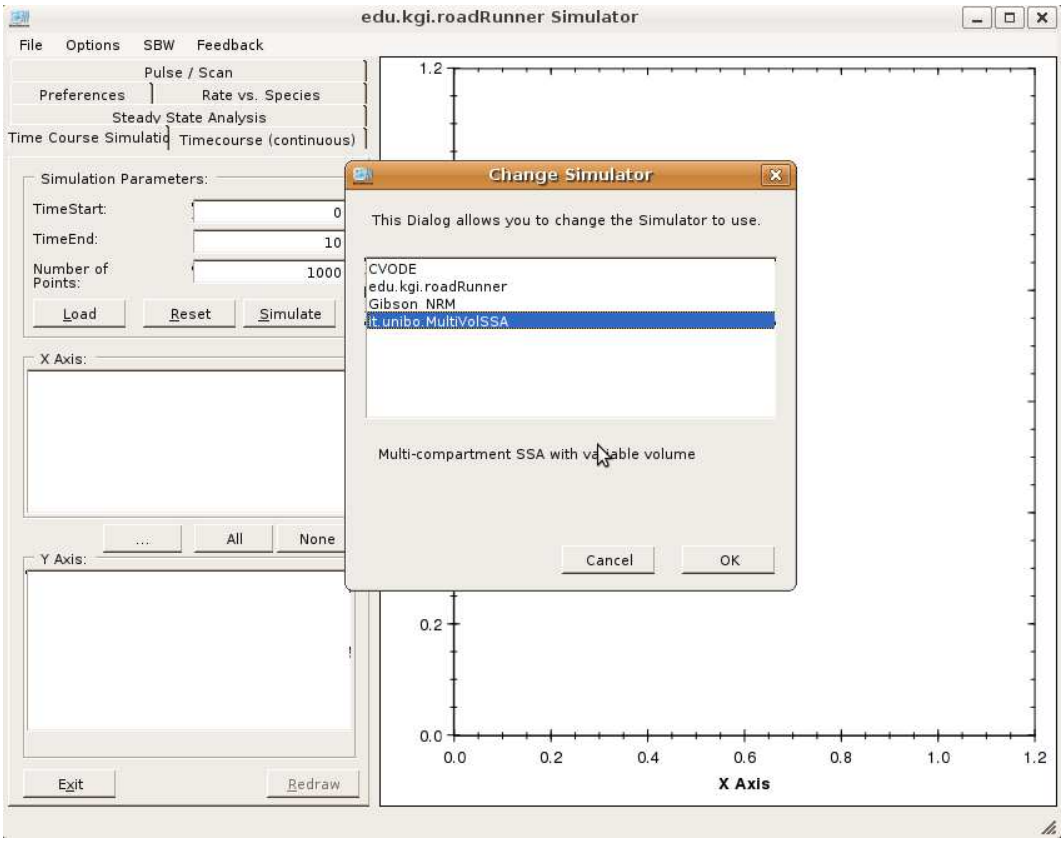


Figure 7.4: Selection of the Multi-compartment Stochastic Simulator from the Graphical User Interface for simulation included in the SBW.

7.3 The Multi-compartment Stochastic Simulator

The Multi-compartment Stochastic Simulator (MSS) is a prototype, multi-platform application written in Java which implements the MSSA described in the previous chapter. The MSS is SBML-capable, i.e. is able to read SBML models and perform their stochastic simulation according to the MSSA.

However, the species volumes $v(S_j)$ for each species S_j (needed by the MSSA for the simulation in the case of variable volumes) do not appear as attributes in the SBML description of chemical species. Fortunately, SBML allows the storage of additional information about any entity of the language in the *annotation* tag, so that the species volumes can be easily included in the SBML model for example in

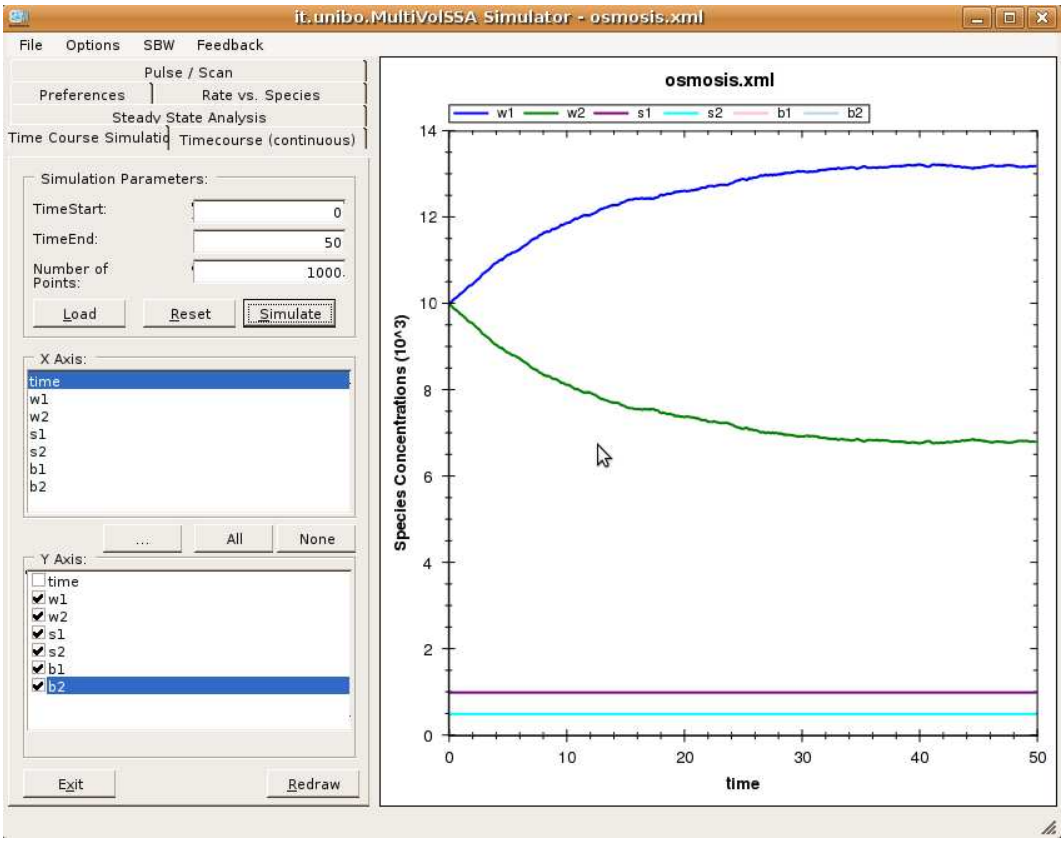


Figure 7.5: Visualisation in the SBW interface of a run of the MSS.

the following way:

```
<species compartment="Comp" id="r"
         initialConcentration="10">
    <annotation>
        <speciesVolume volume="0.001" />
    </annotation>
</species>
```

The annotation contains a *speciesVolume* tag with a single attribute *volume* whose value corresponds to $v(S_j)$.

The MSS can act as a standalone, non-interactive application which accepts as input a few parameters (in particular the name of an SBML file) and exports the

simulation data as a text file in CSV (comma-separated values) format.

In order to take full advantage from the MSS, its integration with the SBW must be taken into account. The MSS constitutes indeed an SBW-enabled tool which provides the *simulation* service (Fig. 7.3).

All the modules included in the SBW (or external, SBW-enabled applications) which can take advantage of the simulation service are consequently able to exploit the MSS. In particular, the graphical simulation interface included in the SBW provides a more user-friendly interaction with the MSS. Such graphical interface allows the user to select the module to be used for the simulation (as shown in Fig. 7.4) and to pick the SBML file to be simulated.

Fig. 7.5 shows the graphical result of a simple model simulated by the MSS.

Chapter 8

Case Studies

In this chapter we present two simple case studies demonstrating the conditions under which the SSA (and in general any algorithm based on the hypothesis of constant compartment volume) cannot provide valid simulation results. The first, in Sect. 8.1, describes a quite common phenomenon in Biology known as osmosis, which denotes the change of volumes in many cells as a function of the external concentration of particular solutes. The second example, in Sect. 8.2, considers the effect of volume variation in the case of cellular growth and division in a simple biochemical system.

8.1 Osmosis

The simple case study considered in this chapter allows us to observe the limits of the previous models and the need to introduce volume information specified in Expr. (6.10), in order to manage correctly the relative rates of reactions as a function not only of the number of elements for each reactant, but also of its concentration. The system in Fig. 8.1 depicts a compartment c of variable size containing a water solution placed in a hypotonic environment e. The compartment is bounded by a semipermeable membrane, i.e. a filter which allows only the water to move across. Experience shows that this situation causes a net movement of water towards compartment c, due to the so-called osmosis phenomenon. Biological occurrence of this

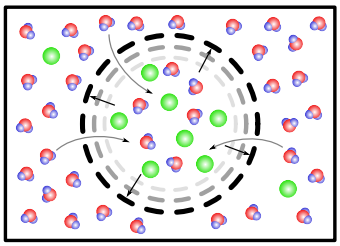


Figure 8.1: Osmosis by semipermeable membrane in hypotonic solution.

circumstance is common both in animal and vegetal cells. A typical animal cell swells when placed in hypotonic and shrinks when in hypertonic solution even if the cell membrane is poorly permeable to water, thanks to the presence of water-channel proteins which allow only water to flow, in either direction. The process A may describe a simple abstraction of water-channel protein acquaporin:

$$HOH(d) \equiv \overline{hoh}@d : v_{H_2O} \qquad S(d) \equiv \overline{s}@d : v_S$$
$$A(f,g) \equiv !\, hoh@f.HOH(g) \mid !\, hoh@g.HOH(f)$$

S is the solute, HOH represents a water molecule able to interact with the acquaporin A and move through the membrane of the cell. In this very simple layout, the acquaporin is completely symmetrical. The system in Fig. 8.1 may be written as

$$Sys \equiv \underbrace{HOH(e) \mid \cdots \mid HOH(e)}_{m_1} \mid \underbrace{S(e) \mid \cdots \mid S(e)}_{n_1} \mid A(c,e) \mid$$
$$\underbrace{HOH(c) \mid \cdots \mid HOH(c)}_{m_2} \mid \underbrace{S(c) \mid \cdots \mid S(c)}_{n_2}$$

where m_1 and n_1 represent the number of water molecules and salt ions outside, m_2 and n_2 the number of molecules and ions inside the cell membrane and only one acquaporin is present, for simplicity. If we discard the information about the volumes V_e and V_c of the two compartments in Expr. (6.10), we are first forced to introduce asymmetry in the encoding of the acquaporin A, in order to differentiate the rate r_e of molecules entering from the rate r_c of molecules leaving the cell. The system can be considered in equilibrium when the probability of a water molecule

entering is equal to the probability of a water molecule leaving the cell, that is when

$$h'_e r_e = h'_c r_c \tag{8.1}$$

where h'_e is the number of possible combinations of acquaporin–water molecules outside, h'_c the possible combinations inside the cell. Since only one acquaporin is present, we have that $h_e = m'_1$ and $h_c = m'_2$, where m'_1 and m'_2 are the number of water molecules at equilibrium. Hence Expr. (8.1) becomes

$$m'_2/m'_1 = r_e/r_c$$

meaning that the equilibrium depends on the rate of the molecules initially conveyed, which is not true. In reality, under the hypothesis of uniform temperature and pressure, the equilibrium is reached when the concentration of the two solutions is the same, that is when

$$m'_2/n'_2 = m'_1/n'_1 \tag{8.2}$$

where n'_1 and n'_2 are the number of salt ions in the respective compartments. Even by artificially setting $r_e = 1/n_1$ and $r_c = 1/n_2$, the model would be incorrect in the case of a variable number of salt ions. Conversely, this dependence is coherently modeled if we consider the right expressions for V_e and V_c. In fact, by Expr. (6.10), we have

$$V_e^{-1} h'_e r_e = V_c^{-1} h'_c r_c \tag{8.3}$$

Since the volumes are obtained as the sum of the average volumes occupied by each element, we have

$$V_e = \sum_e v_{H_2O} + \sum_e v_S = m'_1 v_{H_2O} + n'_1 v_S$$
$$V_c = \sum_c v_{H_2O} + \sum_c v_S = m'_2 v_{H_2O} + n'_2 v_S$$

which – under the initial symmetric assumption $r_e = r_c$ – properly substituted in (8.3) leads to

$$\frac{1}{v_{H_2O} + \frac{n'_1}{m'_1} v_S} = \frac{1}{v_{H_2O} + \frac{n'_2}{m'_2} v_S}$$

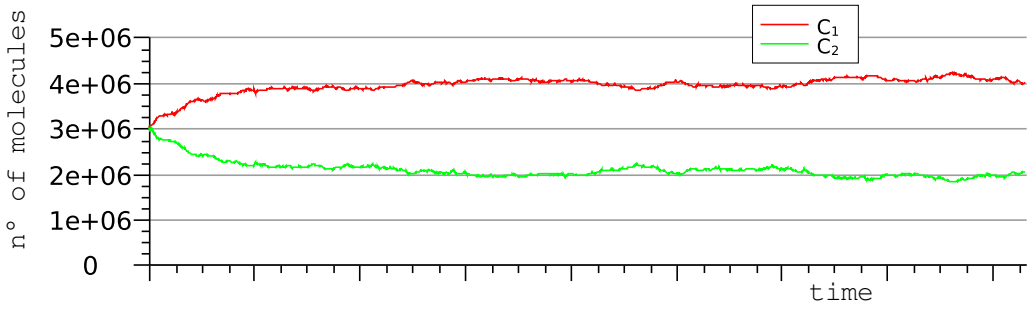


Figure 8.2: Simulation of the osmosis phenomenon in the case of two compartments C_1 and C_2 with the same initial number of H_2O molecules and different salt concentration: the lower (upper) line represents the number of water molecules in C_2 (C_1), with C_2 containing half the number of molecules of salt with respect to C_1.

that is clearly satisfied by expression (8.2). The inclusion of the volume of both water molecules and salt ions in the denominator of the above expression affects the net flux of water molecules through the acquaporin so that its rate smoothly decreases to zero, with a rapidity depending on the ratio between salt and water.

In Fig. 8.2, a MSSA simulation graph of the above example is reported.

Osmosis involves most living cells, and expression (8.2) describes only one of the possible equilibrium conditions. For example, plant cells are surrounded by rigid walls which prevent them from increasing their volume. Consequently, if placed in hypotonic solution, these cells absorb water until the pressure on cell walls equals the osmotic pressure, which depends on the absolute temperature T and the difference of salt ions/molecules concentration. This equilibrium condition cannot be expressed in Sπ@, since no pressure evaluation is present. Such information may be taken into account in the model by defining reaction rates as functions of the absolute temperature T and pressure p_c of the compartment, where p_c may be in turn calculated as a function of the compartment c, the absolute temperature and the elements surrounded by c. This would allow the introduction of temperature, pressure and some of the structural information pertaining to mechanical properties of compartment boundaries. Depending on the kind of function used for evaluating

	Reaction			Propensity function
R_1 :	$S_0 + S_{Act}$	$\xrightarrow{k_1}$	S_1	$k_1 \cdot [S_0] \cdot [S_{Act}]$
R_2 :	S_1	$\xrightarrow{k_{-1}}$	$S_0 + S_{Act}$	$k_{-1} \cdot [S_1]$
R_3 :	S_0	$\xrightarrow{\alpha_0}$	$S_0 + X$	$\alpha_0 \cdot [S_0]$
R_4 :	S_1	$\xrightarrow{\alpha_1}$	$S_1 + X$	$\alpha_1 \cdot [S_1]$
R_5 :	X	$\xrightarrow{k_x}$		$k_x \cdot [X]$

Table 8.1: Biochemical reactions for a simple system describing a constitutive promoter S.

p_c, the computational complexity of the algorithm may grow significantly.

The osmosis example shows a simple (and biologically common) situation where the SSA happens not to be faithful if taken "as it is", since the volume of each element (and not only compartment volumes) must be properly considered during the simulation in order to obtain the correct values for reaction rates as a function of reactants' concentration. The reason is that SSA already embodies the unique, fixed-size volume hypothesis, which is correct for fluid systems under the conditions stated in [45] and needs to be properly translated into other chemical or biological contexts.

8.2 Cellular growth and division

Another simple yet non-trivial example is the effect of the variation of volume as a consequence of cellular growth and division. In this section we consider such effect on the system in Table 8.1, which was firstly analysed in [53] under the hypothesis of constant volume, while in [62] a hybrid variant of Gillespie's SSA was formulated in order to take into account the variation of volume of the cell during the process of growth and subsequent division.

The system involves a single gene which fluctuates between two states S_0 and S_1.

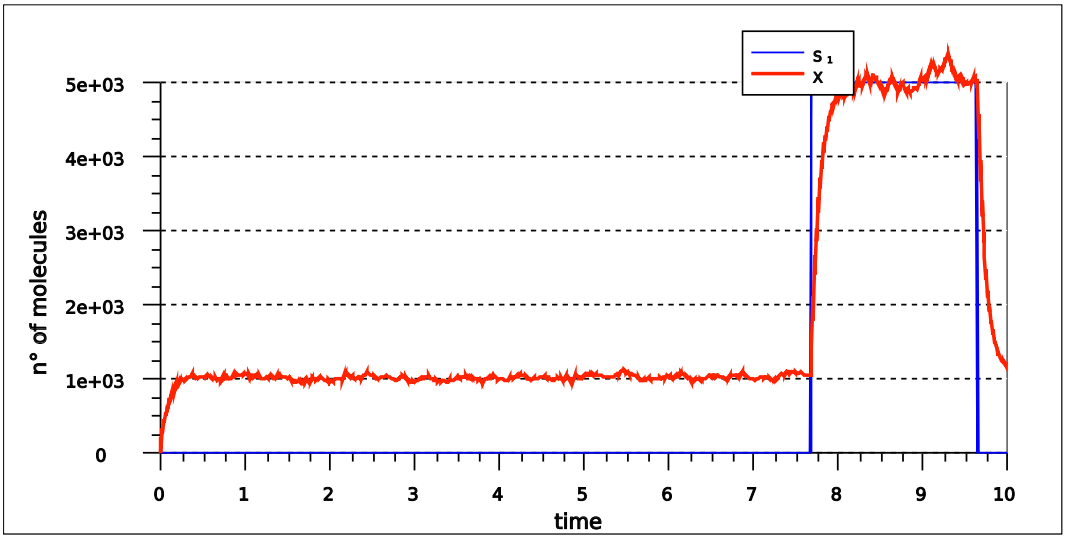


Figure 8.3: Simulation of the system specified in Table 8.1 in the case of a single molecule of the promoter S (in the graph its level is multiplied by $5 \cdot 10^3$ for the sake of readability of the graph) and constant volume $Vol = 1$, with $k_1 = k_{-1} = 0.1$, $\alpha_0 = 1 \cdot 10^4$, $\alpha_1 = 5 \cdot 10^4$, $k_x = 10$.

The transition $S_0 \rightarrow S_1$ occurs when one regulator protein S_{Act} binds to the gene's promoter, while the reverse transition $S_1 \rightarrow S_0$ is supposed to occur autonomously. The transcription of gene S leads to the production of the protein X at rate α_0 when its promoter is in the state S_0, and at at rate α_1 when it is in the state S_1 (with $\alpha_0 < \alpha_1$). The protein X spontaneously degrades at rate k_x.

Under standard conditions, the concentration of the protein X follows a bistable condition which depends on the state of the gene S. In the state S_0, the concentration level of X quickly reaches an equilibrium at α_0/k_x, while in S_1 at α_1/k_x, as shown in Fig. 8.3.

We shall now consider the same system in the case of volume variation as a consequence of the growth and division of the cellular compartment. In [62], such growth obeys a deterministic (exponential) function and the division occurs at fixed time steps. Here we are following an alternative approach, which totally sticks to the stochastic spirit of the SSA and allows a formulation of the system exclusively

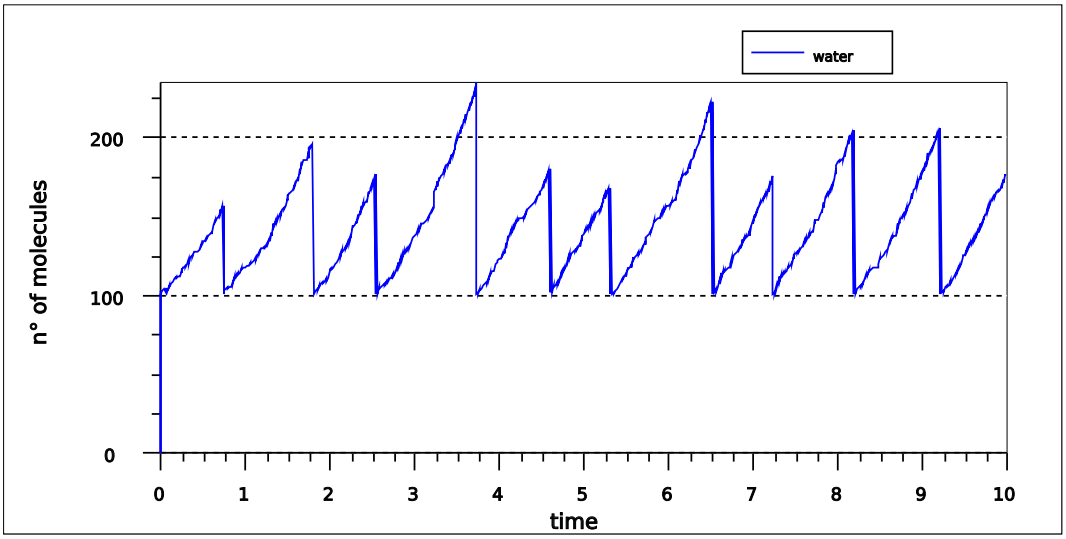


Figure 8.4: Simulation of stochastic volume variation in the case of growing and dividing cells. The number of water molecules exponentially grows until the division of the cell restores its initial amount.

within the standard MSSA.

The exponential growth of the cell compartment can be easily reproduced by means of the introduction of a new chemical species w with volume $v(w)$ which may represent the amount of water (and other substances) inside the cell itself. A reaction rule of the kind

$$R_w : \qquad w \quad \rightarrow \quad 2 \cdot w$$

seamlessly introduces an exponential growth of cellular volume (whose speed is a function of the rate of R_w), supposing that the volume occupied by all the other elements is negligible, that is $v(S_0) = v(S_1) = v(S_{Act}) = v(X) = 0$.

The event of cell division can be signalled by the stochastic appearance of a single molecule of species k, which dramatically changes the configuration of the system through a series of infinite-rate reductions. It first sets the volume of the cell

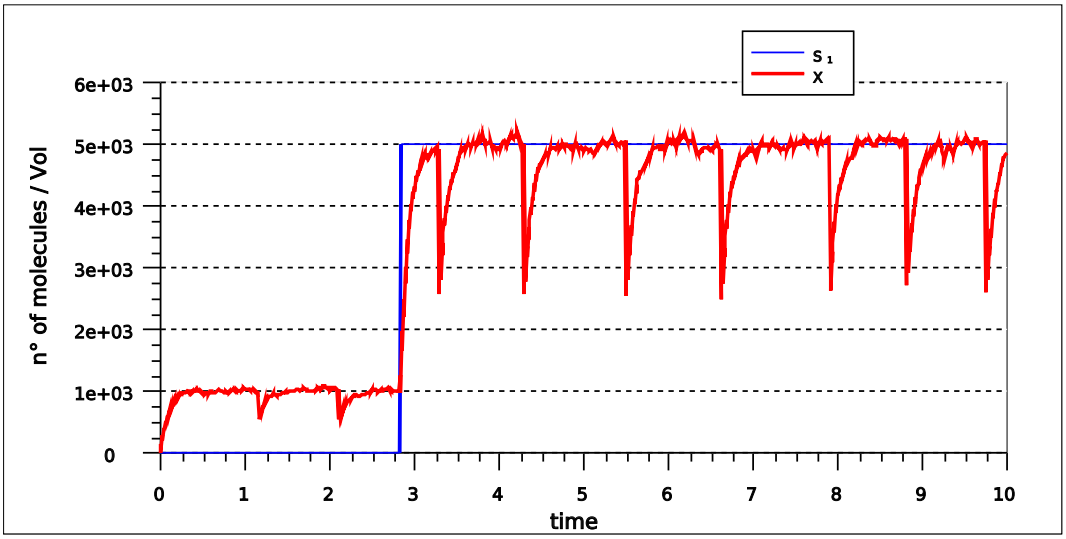


Figure 8.5: Simulation of cellular growth and division with complete neglect of volume variation: the concentration of protein X quickly reaches the equilibrium after each cell division, which happens approximately once per time unit.

to zero, then it "stochastically halves" the number of proteins of species X:

$$\begin{aligned} R_{1k}: &\quad k + w \quad \rightarrow \quad k \\ R_{2k}: &\quad k + X \quad \rightarrow \quad k \\ R_{2'k}: &\quad k + X \quad \rightarrow \quad k + X' \end{aligned}$$

with $\text{rate}(R_{1k}) = \text{rate}(R_{2k}) = rate(R_{2'k}) = \infty$. R_{1k} eliminates all the water molecules, while $R_{2k}, R_{2'k}$ simulate the division of the cytosol between the two budding cells: $R_{2'k}$ replace roughly half of the occurrences of X with X', which represents the protein X inside one of the two new cells, while R_{2k} ideally moves the protein inside the other budding cell, which is not explicitly modelled.

After a negligible time, the species k disappears according to the reaction

$$R_{kk}: \qquad k \quad \rightarrow \quad k_0 + n \cdot w \qquad\qquad n = Vol/v(w)$$

which also restores the initial volume Vol of the cell (to be precise, such volume should be "stochastically halved", but this expedient constitutes an effective way

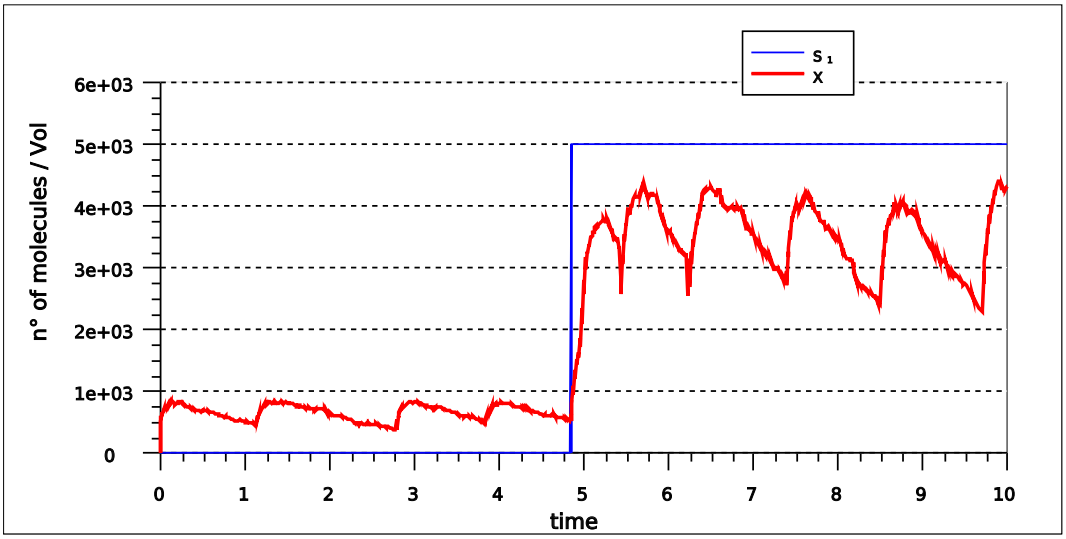


Figure 8.6: Level of concentration of the protein X during the process of growth and division: the division on average occurs once every one time unit, when the cell has approximately doubled in size. The concentration of protein X never reaches an equilibrium, following instead an oscillatory regime.

to limit its fluctuation, in the absence of the reliable control mechanisms typical of real biological cells) and allows the subsequent relabelling of the X' proteins:

$$R_{1r}: \qquad k_0 + X' \quad \rightarrow \quad X \qquad\qquad \text{rate}(R_{1r}) = \infty$$

A simulation of the fluctuation of the number of water molecules (and consequently of the volume of the system, according to the previous hypotheses) produced by such reaction rules is shown in Fig. 8.4.

As we have previously discussed, in a standard situation the concentration of the X protein in the cell reaches an equilibrium which depends on the state of the gene S. Upon cell division, the number of proteins is halved. If we disregard the variation of volume, the concentration of X is suddenly halved after each cell division, then quickly reaches the previous equilibrium level, according to the simulation in Fig. 8.5.

The behaviour of the system considerably changes if, conversely, the volume of the compartment is properly considered. Fig. 8.6 reports the simulation of the

system (in perfect agreement with the results presented in [62]) in the case that the division of the cell occurs once per time unit, after that (on average) the volume of the cell has doubled. In this situation the concentration of the protein X follows a quite different law: upon cell division, its concentration does not considerably change (both the volume and the number of proteins are approximately halved), while during each growth cycle it follows an oscillatory regime whose local minima coincide with the cell division (i.e. with the local maxima of cell volume).

The different behaviour of the systems in Figures 8.5 and 8.6 demonstrates the error deriving from the neglect of volume variations which heavily influences, in this case, the activity of the protein X.

Chapter 9

Conclusions

The application of concurrent languages to Systems Biology is still in its earliest stage of progress. Consistent efforts are going to be made in pursuit of increased biological faithfulness, better exploitation of techniques and development of software tools for analysis in silico of biological phenomena. To this aim, the evaluation of the expressiveness of current languages and the integration of further bio-oriented primitives will play a central role.

In this thesis we have presented $\pi@$, an extremely simple language for biological modelling whose expressive properties make it the keystone for research in this direction. The $\pi@$ language is obtained as a conservative extension of the π-calculus, extended with polyadic synchronisation and static, global priority.

Despite of its simplicity, $\pi@$ presents noteworthy modelling capabilities: beyond the description of basic chemical reactions, the calculus can model formation of molecular complexes, hierarchical organisation of the system in subcompartments, inter-compartment reactions (including exchange of elements between adjacent compartments) dynamic compartment structure (run-time creation or destruction of compartments, merging, splitting, migration of compartments into or out of other compartments).

Its stochastic counterpart, S$\pi@$, provides increased faithfulness in quantitative simulation of models thanks to the inclusion of additional physical properties. The simulation algorithm consists of a multi-compartment extension of Gillespie's one,

and handles the variation of compartment volumes consistently with the dynamic reorganisation of compartment structure. This extended simulation algorithm, of which an optimised version is also presented, is shown able to model effectively further biological phenomena like osmosis, cellular growth and division.

The remarkable expressiveness of $\pi@$ gives this calculus the ability of reproducing the exact behaviour of several bio-inspired formalisms. In this respect, the encodings of BioAmbients, Brane Calculi and catalytic P Systems have been provided here. Such encodings are *modular*: they allow independent translation into $\pi@$ of each process of the source language, so that the compilation can be parallel or incremental. In the case of BioAmbients and Brane Calculi, $\pi@$ can replicate the dynamic behaviour of all the operations related to compartments. Notably, $\pi@$ can also reproduce the semantics of catalytic P systems, which have a static compartment structure but are characterised by *maximal parallelism*. Also Beta binders [83] – another well known calculus denoted by explicit compartment semantics – have been encoded into $\pi@$ [17].

On the theoretical side, the first results on the expressiveness of the kind of priority exploited in $\pi@$ have been provided, in terms of *separation* between prioritised and non-prioritised formalisms. In particular, two languages denoted by two different kinds of static priority have been considered: the first one, FAP, consisting of a fragment of asynchronous CCS extended with *global* priority (the same used in $\pi@$, of which FAP is a fragment too, indeed); the second one, CPG, is denoted by a *local* kind of priority. The first result on the expressive gap between local and global priority has been proved here in terms of nonexistence of modular encoding of FAP into CPG. The most important results are anyway related to the impossibility of encoding in a modular way such prioritised languages into other two non-prioritised languages, that is the π-calculus and the bπ-calculus, characterised by point-to-point and broadcast communication respectively.

The results presented in this thesis establish the possible uses of $\pi@$ under different perspectives.

Its direct application to biological modelling provides the greatest flexibility for

the expression of systems denoted by dynamical structures at molecular and compartmental level. The possibility of encoding complex operations as atomic sequences of low-level steps provides the calculus with the capability of reproducing new biological operations at will, which can be then integrated with the existing ones in a seamless way. Such capability keeps $\pi@$ open to the inclusion of future biological primitives of interest.

The easy encoding of several bio-inspired formalisms (BioAmbients, Brane Calculi, a variant of P Systems presented here and also Beta binders) demonstrates the usefulness of $\pi@$ for the analysis and development of such formalisms: once translated, their encodings can be analysed and compared in order to understand their structural and semantic common points and differences. Furthermore, such encodings represent their correct implementation on top of $\pi@$: their conciseness evidence the minimum effort needed to provide such an implementation and to prove its correcness, once a proper implementation of $\pi@$ is given. These considerations support the choice of $\pi@$ as the optimal core of a framework for the quick implementation of bio-inspired formalisms.

The separation results between prioritised and non-prioritised languages points out the price to pay for such expressiveness and flexibility. The impossibility of encoding FAP and CPG in a modular way into π-calculus and bπ-calculus attests to the difficulties of providing a *distributed* implementation of $\pi@$. It is indeed shown that the semantics of priority (global as well as local) cannot be obtained by a purely parallel implementation even if powerful primitives such as broadcast are employed and deadlocks and divergence are tolerated. Consequently, any possible implementation of $\pi@$ is going to be *centralised*. The strength of the separation suggests also that even a partial parallelisation may likely cause significant overhead due to the frequent and spread additional synchronisations, so that the performance gain addressed with such parallelisation would be dramatically reduced if not totally lost.

As remarked before, these results on the impossibility of parallelisation of priority have wide validity. They can be applied in particular to the stochastic variants

of many other formalisms which would suffer from the same parallelisation issues deriving from the introduction of infinite-rate transitions in their implementations.

9.1 Future work

The analysis carried out in this thesis leaves much work to do and several open questions. The implementation of π@ and Sπ@ constitute just the first and easiest step to be achieved. Further efforts would be required to provide an *optimised* implementation, so that the encodings of formalisms on top of this language may have as good performance as in respective native implementations.

Moreover, π@ expressiveness needs to be investigated in many respects.

For example, the encoding of further formalisms (e.g. [32, 37]) into π@ is definitely one of the main directions to follow. This is intrinsically related to the determination of the expressiveness of priority in the reproduction of properties that are not native in π@ such as, for example, maximal parallelism in the presence of dynamical compartments, non-binary reactions, and so on.

Another relevant aspect is related to the properties (e.g. causality, stochastic semantics and observational equivalences) preserved by the encoding functions: the determination of such properties may shift the application of the related analysis techniques (e.g. causal analysis, stochastic simulation and even model checking) directly to the encodings. This would shorten the time required for software development by exploiting π@ as bridge towards the encoded formalisms.

With respect to priority, many issues need to be resolved. The relation between π@ and core-π@ is still unknown, i.e. how the number of priority levels affects the expressiveness, and also if polyadic synchronisation may be in turn encoded by priority and under what hypotheses this could be achieved. Moreover, it would be worth considering local priority in the style of CPG as an alternative: the potential gain of performance in the implementation and the likely loss of expressiveness should be evaluated.

Moreover, other extensions to the calculus should be taken into account. The inclusion of further physical parameters in Sπ@, such as pressure and temperature, should be considered in relation to the additional computational complexity required for their simulation.

Syntactic object-oriented facilities in the style of SpiCO [55] should be introduced in order to simplify the structuring of the code and increase the readability of the encodings.

References

[1] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, IX(2):127–168, 1986.

[2] Jos C. M. Baeten, Jan A. Bergstra, and Jan Willem Klop. Ready-trace semantics for concrete process algebra with the priority operator. *Comput. J.*, 30(6):498–506, 1987.

[3] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi, and A. Troina. Stochastic Calculus of Looping Sequences for the Modelling and Simulation of Cellular Pathways. *Transactions on Computational Systems Biology*, 2008.

[4] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. A Calculus of Looping Sequences for Modelling Microbiological Systems. *Fundamenta Informaticae*, 72(1):21–35, 2006.

[5] Marco Bernardo and Roberto Gorrieri. Extended Markovian Process Algebra. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 315–330. Springer, 1996.

[6] Marco Bernardo and Roberto Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theor. Comput. Sci.*, 202(1-2):1–54, 1998.

[7] C. Bodei. A Static Analysis for Beta-Binders. *Electronic Notes in Theoretical Computer Science*, 194(3):69–85, 2008.

[8] L. Bortolussi. Stochastic Concurrent Constraint Programming. *Electronic Notes in Theoretical Computer Science*, 164(3):65–80, 2006.

[9] L. Bortolussi and A. Policriti. Stochastic Concurrent Constraint Programming and Differential Equations. *Electronic Notes in Theoretical Computer Science*, 190(3):27–42, 2007.

[10] Mario Bravetti and Roberto Gorrieri. The theory of interactive generalized semi-Markov processes. *Theor. Comput. Sci.*, 282(1):5–32, 2002.

[11] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0. *W3C Recommendation*, 6, 2000.

[12] Linda Brodo, Pierpaolo Degano, and Corrado Priami. A Stochastic Semantics for BioAmbients. In Victor E. Malyshkin, editor, *PaCT*, volume 4671 of *Lecture Notes in Computer Science*, pages 22–34. Springer, 2007.

[13] Nadia Busi and Roberto Gorrieri. On the computational power of Brane Calculi. In Corrado Priami and Gordon D. Plotkin, editors, *T. Comp. Sys. Biology*, volume 4220 of *Lecture Notes in Computer Science*, pages 16–43. Springer, 2006.

[14] Muffy Calder, Stephen Gilmore, and Jane Hillston. Automatically deriving ODEs from process algebra models of signalling pathways. In Gordon Plotkin, editor, *Proceedings of Computational Methods in Systems Biology (CMSB 2005)*, pages 204–215, Edinburgh, Scotland, April 2005.

[15] Juanito Camilleri and Glynn Winskel. CCS with priority choice. *Inf. Comput.*, 116(1):26–37, 1995.

[16] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J Chem Phys*, 121(9):4059–4067, September 2004.

[17] I. Cappello and P. Quaglia. A translation of beta-binders in a prioritized pi-calculus. *Electronic Notes in Theoretical Computer Science*, 229(1):109–125, 2009.

[18] Marco Carbone and Sergio Maffeis. On the expressive power of polyadic synchronisation in pi-calculus. *Nord. J. Comput.*, 10(2):70–98, 2003.

[19] L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190–215, 2008.

[20] L. Cardelli and A. D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[21] L. Cardelli and S. Pradalier. Where Membranes Meet Complexes. In *Proceedings of Bio-CONCUR*, 2005.

[22] Luca Cardelli. Brane Calculi. In Danos and Schächter [39], pages 257–278.

[23] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *FoSSaCS*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.

[24] D. Carlisle, P. Ion, R. Miner, and N. Poppelier. Mathematical Markup Language (MathML) Version 2.0. *W3C Recommendation*, 21, 2001.

[25] Paolo Cazzaniga, Dario Pescini, Daniela Besozzi, and Giancarlo Mauri. Tau leaping stochastic simulation method in p systems. In Hendrik Jan Hoogeboom, Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Workshop on Membrane Computing*, volume 4361 of *Lecture Notes in Computer Science*, pages 298–313. Springer, 2006.

[26] Paolo Cazzaniga, Dario Pescini, Francisco-José Romero-Campero, Daniela Besozzi, and Giancarlo Mauri. Stochastic approaches in P systems for simulating biological systems. In Miguel Angel Gutiérrez-Naranjo, Gheorghe Paun, Agustín Riscos-Núñez, and Francisco José Romero-Campero, editors, *Fourth*

Brainstorming Week on Membrane Computing, Sevilla, January 30 - February 3, 2006. Volume I, pages 145–164. Fénix Editora, 2006.

[27] D. Chiarugi, Michele Curti, Pierpaolo Degano, and Roberto Marangoni. VICE: A VIrtual CEll. In Danos and Schächter [39], pages 207–220.

[28] F. Ciocchetta and J. Hillston. Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theor. Comput. Sci., to appear*, 2008.

[29] F. Ciocchetta and J. Hillston. Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks. *Electronic Notes in Theoretical Computer Science*, 194(3):103–117, 2008.

[30] Federica Ciocchetta, Stephen Gilmore, Maria Luisa Guerriero, and Jane Hillston. Integrated Simulation and Model-Checking for the Analysis of Biochemical Systems. In *Proceedings of Practical Application of Stochastic Methods (PASM 2008)*, 2008 (to appear on ENTCS).

[31] Federica Ciocchetta and Maria Luisa Guerriero. Modelling Biological Compartments in Bio-PEPA. In *Proceedings of the 2nd International Meeting on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2008)*, 2008 (to appear on ENTCS).

[32] Federica Ciocchetta and Jane Hillston. Bio-PEPA: a framework for the modelling and analysis of biological systems, 2008. Theoretical Computer Science.

[33] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebra. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 711–765. Elsevier Science Publishers, February 2001.

[34] Rance Cleaveland and Matthew Hennessy. Priorities in process algebras. *Inf. Comput.*, 87(1/2):58–77, 1990.

[35] A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, and G. Zavattaro.

Modelization and Simulation of Nano Devices in nanokappa Calculus. *LECTURE NOTES IN COMPUTER SCIENCE*, 4695:168, 2007.

[36] M. Curti, P. Degano, and C. Baldari. Causal pi-calculus for biochemical modelling, 2003.

[37] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.

[38] Vincent Danos and Sylvain Pradalier. Projective Brane Calculus. In Danos and Schächter [39], pages 134–148.

[39] Vincent Danos and Vincent Schächter, editors. *Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers*, volume 3082 of *Lecture Notes in Computer Science*. Springer, 2005.

[40] Frank S. de Boer and Catuscia Palamidessi. Embedding as a tool for language comparison. *Inf. Comput.*, 108(1):128–157, 1994.

[41] J. Elf and M. Ehrenberg. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems Biology*, 1(2):230–236, 2004.

[42] Cristian Ene and Traian Muntean. Expressiveness of point-to-point versus broadcast communications. In Gabriel Ciobanu and Gheorghe Paun, editors, *FCT*, volume 1684 of *Lecture Notes in Computer Science*, pages 258–268. Springer, 1999.

[43] M.A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[44] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976.

[45] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[46] I. Goryanin. DBsolve: Software for metabolic, enzymatic and receptor-ligand binding simulation. *Available via the World Wide Web at http://websites. ntl. com/igor. goryanin*, 2001.

[47] J. Gosling. *The Java Language Specification*. Addison-Wesley Professional, 2000.

[48] Maria Luisa Guerriero and Corrado Priami. Causality and Concurrency in Beta-binders. Technical Report TR-01-2006, CoSBi, 2006. Available at `http://www.cosbi.eu/php/get_paper.php?id=1`.

[49] M.L. Guerriero, J.K. Heath, and C. Priami. An Automated Translation from a Narrative Language for Biological Modelling into Process Algebra. *LECTURE NOTES IN COMPUTER SCIENCE*, 4695:136, 2007.

[50] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002.

[51] M. Hucka, A. Finney, HM Sauro, and H. Bolouri. Introduction to the Systems Biology Workbench. *Available via the World Wide Web at http://sbw.kgi.edu/caltechSBW/sbwDocs/docs/intro/intro.pdf*, 2001.

[52] M. Hucka, A. Finney, HM Sauro, H. Bolouri, JC Doyle, H. Kitano, AP Arkin, BJ Bornstein, D. Bray, A. Cornish-Bowden, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, 2003.

[53] T.B. Kepler and T.C. Elston. Stochasticity in Transcriptional Regulation: Origins, Consequences, and Mathematical Representations. *Biophysical Journal*, 81(6):3116–3136, 2001.

[54] Jean Krivine, Robin Milner, and Angelo Troina. Stochastic bigraphs. In *Porc. of MFPS'08, 24th Conference on the Mathematical Foundations of Programming Semantics*, volume to appear of *ENTCS*. Elsevier, 2008.

[55] Céline Kuttler, Cédric Lhoussaine, and Joachim Niehren. A stochastic pi calculus for concurrent objects. In Hirokazu Anai, Katsuhisa Horimoto, and Temur Kutsia, editors, *AB*, volume 4545 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 2007.

[56] Marek Kwiatkowski and Ian Stark. The continuous π-Calculus: a Process Algebra for Biochemical Modelling. In *CMSB*, volume 5307 of *Lecture Notes in Computer Science*. Springer, 2008.

[57] Cosimo Laneve and Fabien Tarissan. A simple calculus for proteins and cells. *Electr. Notes Theor. Comput. Sci.*, 171(2):139–154, 2007.

[58] Gérard Le Lann. Distributed systems - towards a formal approach. In *IFIP Congress*, pages 155–160, 1977.

[59] N. Le Novere and T.S. Shimizu. StochSim: modelling of stochastic biomolecular processes, 2001.

[60] Paola Lecca, Corrado Priami, Carlo Laudanna, and G. Constantin. Predicting cell adhesion probability via the biochemical stochastic pi-calculus. In Hisham Haddad, Andrea Omicini, Roger L. Wainwright, and Lorie M. Liebrock, editors, *Symposium on Applied Computing*, pages 211–212. ACM, 2004.

[61] L.M. Loew and J.C. Schaff. The Virtual Cell: a software environment for computational cell biology. *Trends in Biotechnology*, 19(10):401–406, 2001.

[62] T. Lu, D. Volfson, L. Tsimring, and J. Hasty. Cellular growth and division in the gillespie algorithm. In *Systems Biology, IEE Proceedings*, pages 121–128, 2004.

[63] P. Mendes. GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems. *Bioinformatics*, 9(5):563–571, 1993.

[64] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[65] R. Milner. The polyadic pi-calculus: a tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.

[66] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[67] Robin Milner. Functions as processes. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 167–180. Springer-Verlag, 1990.

[68] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, New York, NY, USA, 1999.

[69] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992.

[70] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Inf. Comput.*, 100(1):41–77, 1992.

[71] M.L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1967.

[72] Uwe Nestmann. What is a "good" encoding of guarded choice? *Inf. Comput.*, 156(1-2):287–319, 2000.

[73] Uwe Nestmann and Benjamin C. Pierce. Decoding choice encodings. *Inf. Comput.*, 163(1):1–59, 2000.

[74] F. Nielson, H.R. Nielson, C. Priami, and D. Rosa. Static analysis for systems biology. In *Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6. Trinity College Dublin, 2004.

[75] Denis Noble. *The Music of Life: Biology Beyond Genes*. OUP Oxford, February 2008.

[76] Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

[77] Catuscia Palamidessi and Oltea Mihaela Herescu. A randomized encoding of the pi-calculus with mixed choice. *Theor. Comput. Sci.*, 335(2-3):373–404, 2005.

[78] Gheorghe Păun. Introduction to membrane computing. In *First brainstorming Workshop on Uncertainty in Membrane Computing, Palma de Mallorca, Spain, November 2004*, 2004.

[79] Andrew Phillips. A correct abstract machine for the stochastic bioambient calculus. In *Membrane Computing and Biologically Inspired Process Calculi*, ENTCS, 2008. In press.

[80] Andrew Phillips and Luca Cardelli. A correct abstract machine for the stochastic pi-calculus. In *Bioconcur'04*. ENTCS, August 2004.

[81] Iain Phillips. CCS with priority guards. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2001.

[82] Iain Phillips. CCS with priority guards. *Journal of Logic and Algebraic Programming*, 75(2):139–165, 2008.

[83] Corrado Priami and Paola Quaglia. Beta binders for biological interactions. In Danos and Schächter [39], pages 20–33.

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

[85] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419(6905):343, 2002.

[86] A. Regev, W. Silverman, and E. Shapiro. Representing biomolecular processes with computer process algebra: π-calculus programs of signal transduction pathways. *Proceedings of the Pacific Symposium of Biocomputing*, 2000, 2000.

[87] Aviv Regev. *Computational Systems Biology: A Calculus for Biomolecular knowledge*. PhD thesis, Tel Aviv University, 2002.

[88] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Y. Shapiro. BioAmbients: an abstraction for biological compartments. *Theor. Comput. Sci.*, 325(1):141–167, 2004.

[89] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[90] H.M. Sauro. Jarnac: A system for interactive metabolic analysis. In *Animating the Cellular Map: Proceedings of the 9th International Meeting on Bio-ThermoKinetics. Stellenbosch University Press. ISBN 0-7972-0776-7*, 2000.

[91] M. Tomita, K. Hashimoto, K. Takahashi, T.S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J.C. Venter, et al. E-CELL: software environment for whole-cell simulation. *Bioinformatics*, 15(1):72–84, 1999.

[92] Cristian Versari. A core calculus for a comparative analysis of bio-inspired

calculi. In Rocco De Nicola, editor, *ESOP*, volume 4421 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2007.

[93] Cristian Versari. Encoding catalytic p systems in pi@. *Electr. Notes Theor. Comput. Sci.*, 171(2):171–186, 2007.

[94] Cristian Versari and Nadia Busi. Stochastic biological modelling in presence of multiple compartments. *Theoretical Computer Science*. To appear.

[95] Cristian Versari and Nadia Busi. Stochastic simulation of biological systems with dynamical compartment structure. In Muffy Calder and Stephen Gilmore, editors, *CMSB*, volume 4695 of *Lecture Notes in Computer Science*, pages 80–95. Springer, 2007.

[96] Cristian Versari and Nadia Busi. Efficient Stochastic Simulation of Biological Systems with Multiple Variable Volumes. *Electronic Notes in Theoretical Computer Science*, 194(3):165–180, 2008.

[97] Cristian Versari, Nadia Busi, and Roberto Gorrieri. An expressiveness study of priority in process calculi. *Mathematical Structures in Computer Science*. To appear.

[98] Cristian Versari, Nadia Busi, and Roberto Gorrieri. On the expressive power of global and local priority in process calculi. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 241–255. Springer, 2007.

[99] Cristian Versari and Roberto Gorrieri. π@: a π-based process calculus for the implementation of compartmentalised bio-inspired calculi. In Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro, editors, *SFM*, volume 5016 of *Lecture Notes in Computer Science*, pages 449–506. Springer-Verlag, 2008.

[100] M.G. Vigliotti, I. Phillips, and C. Palamidessi. Tutorial on separation results in process calculi via leader election problems. *Theoretical Computer Science*, 388(1-3):267–289, 2007.