Technical Report CoSBi 13/2006

Process Calculi Abstractions for Biology

Maria Luisa Guerriero
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
guerrier@dit.unitn.it

Davide Prandi
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
prandi@cosbi.eu

Corrado Priami
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
and
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
priami@cosbi.eu

Paola Quaglia
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
quaglia@dit.unitn.it

This is the preliminary version of a paper that will appear in
Algorithmic Bioprocesses
available at http://www.springer.com/computer/foundations/book/978-3-540-88868-0

Process Calculi Abstractions for Biology

Maria Luisa Guerriero
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
guerrier@dit.unitn.it
and
Davide Prandi
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
davide.prandi@dit.unitn.it
and
Corrado Priami
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
priami@cosbi.eu
and
Paola Quaglia
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
quaglia@dit.unitn.it

Abstract

Several approaches have been proposed to model biological systems by means of the formal techniques and tools available in computer science. To mention just a few of them, some representations are inspired by Petri Nets theory, and some other by stochastic processes. A most recent approach consists in interpreting the living entities as terms of process calculi where the behavior of the represented systems can be inferred by applying syntax-driven rules.

A comprehensive picture of the state of the art of the process calculi approach to biological modeling is still missing. This paper goes in the direction of providing such a picture by presenting a comparative survey of the process calculi that have been used and proposed to describe the behavior of living entities.

1 Introduction

The recent progress of biology is rapidly producing a huge number of experimental results and it is becoming impossible to coherently organize them using only human power. Abstract models to reason about biological systems is becoming an indispensable conceptual and computational tool for biologists, so calling for computer science.

The biological approach clarifies components, for example proteins and cells. It also gives graphical and very readable representations of the interactions among

the above entities. However, all these aspects are far from being formally defined. Formal foundations of descriptions are mandatory requirements for enhancing the understanding of complex biological systems and for automatic simulation and analysis. Computer science modeling is specifically designed to meet the above requirements, but it heavily uses mathematical symbolism that is not easy to read for a neophyte. Therefore we need an approach that hides as many technical details as possible from users.

A further aspect to consider is the abstraction level that one wants to implement. Biology tries to answer a wide set of questions that are distributed on an (imaginary) scale. For example, classical genetic analysis uses the *gene* as elementary unit, ignoring (i.e. abstracting from) the biochemical properties of its elements. Abstraction is a powerful technique in computer science, where researchers often face undecidable problems. An abstraction has to capture the essential properties of the phenomenon under consideration, and, at the same time, it has to be computable, to allow automatic analysis, and extensible, to permit the addition of further details [65].

The models and the computational tools developed over the last years focused on molecular biology. Research in bioinformatics started from the observation that biological molecules in real systems participate in very complex networks, like regulatory networks for gene expression, intracellular metabolic networks and intra/inter-cellular communication networks. Due to the (relatively) recent studies in molecular biology and the omics disciplines, there is an accurate description of the fundamental components of the living systems, especially of proteins and cells, but there is not a complete knowledge on how these individual components are related and interact to form complex systems.

To cope with the complexity of these systems various computational approaches have been developed and used. Among them we mention the following ones:

- *biochemical kinetic models* (see, e.g., [3, 77, 71]);
- *generalized models of regulation* (see, e.g, [74, 1, 2, 20]);
- *functional object-oriented databases* (see, e.g., [76, 4, 38];
- *integrated frameworks with GUI* (see, e.g., [75, 70];
- *exchange languages* (see, e.g., [39, 23]).

In recent times, a paradigmatic shift occurred in biology. Researchers started trying to build system visions rather than component visions, and the focus is now rapidly moving from structure to function. This process leads to the so-called *Systems Biology* [40] that is mostly interested in the behavior of cellular processes and in the description of the interactions among components. Seen from a computer science point of view, the methods and the techniques that could be best suited to face the challenge of systems biology are those related to the description and simulation of interacting distributed systems.

Indeed, formal methods have gained increasing attention. Notable examples are those that use the graphical formalism of *Petri Nets* [67]. A Petri Net is an automaton whose states are sets of distributed components. A transition may transform some elements of a state, and more than one transition can be allowed to

occur at the same time. Thanks to their intuitive graphical representations, some variants of Petri Nets have been successfully developed to model biological systems (see, e.g., [27, 30, 63, 55, 34, 42]). More sophisticated models like self-modified Petri Nets [34], Hybrid Petri Nets [46], Stochastic Activity Networks [49] and MetaNets [42] have been used, too. Recent studies also used *Statecharts* and *Live Sequence Charts* [36, 22, 37] for biological modeling. Both formalisms are visual languages originating from the theory of reactive systems and software engineering. Moreover, Live Sequence Charts allow, through a specific methodology, the automated analysis of the biological data they represent [29]. Finally, *membrane systems* [54] (also called P *Systems*) are computational models based upon the notion of *membrane* structure. The model is founded on the observation that complex biological systems are composed by independent computing processes separated by and communicating through membranes. Membranes delimit regions and comprise *objects* and *evolution rules*. A computation is obtained starting from an initial configuration of membrane and objects and then applying evolution rules. The research in this area is currently very active and a comprehensive bibliography includes hundreds of papers [51], some of them aiming at finding formal properties (e.g., [53, 52]) and some of them working on systems biology applications (e.g., [24, 56]).

The above modeling examples witness that the use of formal methods for systems biology is actually promising, but the conceptual tools used are either limited in compositionality or in their ability to handle quantitative data. Another approach to biological modeling is based on process calculi [47, 11, 48, 69]. Processes are the basic units of these languages: they have internal states and interaction capabilities. When a process receives an input its behavior is based on its internal state and on the content of the input. A direct consequence of interaction can be the modification of the internal state and of the interaction capabilities of the interacting processes. In the setting of process calculi, complex entities, like protein complexes, can be described hierarchically, and this allows either top/down or bottom/up analysis. Moreover, process calculi typically come equipped with well-assessed equivalence relations which could be powerful tools for biology. For example, the equivalence of the same functional unit in different organisms could be used as a measure of behavioral and structural similarity.

The process calculi approach to the formal modeling of biological systems has gained more and more attention over the last few years, particularly since the publication on Nature of the landmark paper by Regev and Shapiro [65]. A comprehensive picture of the state of the art, however, is still missing. This paper goes in the direction of providing such a picture and presents a survey of the process calculi that have been proposed to describe the behavior of living entities. We will also point out the available tools based on the calculi we describe.

This survey paper is mainly intended for computer scientists who are interested in understanding how formal techniques from process calculi theory can be used to model biological systems. The reader who is not familiar with process calculi descriptions of concurrent tasks can find in Sect. 2 a short and high-level presentation that outlines the main features of the process calculi approach to concurrency and provides references to basic literature in the field.

The rest of the paper is organized as follows. Sect. 3 sets up some biological notions, providing a common background that allows the comparison of the pro-

posed calculi. Sect. 4 introduces the basic abstraction principle that relates biology and process calculi. Sect. 5 presents a particular biological phenomenon relative to the immune system. Such phenomenon is used as running example in Sect. 6 to comment on various calculi for biology which have been proposed in the literature. In particular, we deal with *biochemical stochastic π-calculus* [66, 62], *BioAmbients* [64], *Brane calculi* [9], *CCS-R* [15], *Beta-binders* [61], *PEPA* [31] and *κ-calculus* [16]. The primitives of the above calculi and languages are analyzed by referring to the running example. Sect. 7 concludes the presentation with some final remarks on the set of calculi considered.

2 Overview on process calculi

Starting from the forerunner CCS, the 'Calculus of Communicating Systems' [47], process calculi have been defined with the primary goal of providing formal specifications of concurrent processes, namely of computational entities executing their tasks in parallel and able to synchronize over certain kinds of activities. The model of a system is typically given as a program or a term that defines the possible behaviors of the various components of the system. Calculi are then equipped with syntax-driven rules, the so-called *operational semantics* [58]. These rules, that can automatically allow the inference of the possible future of the system under analysis. For instance, they can specify that a certain process P evolves into process Q, written $P \longrightarrow Q$.

The basic entities of process calculi are *actions* and *co-actions* (complementary actions). In the most basic view, like e.g. in CCS, an action is seen as an input or an output over a channel. Input is complementary to output and vice-versa. Actions and co-actions can also transmit/receive names over the channel (e.g. the IP address of the Internet) on which input and output are supposed to take place. This is, indeed, the underlying assumption taken in the π-calculus [48]. As it will be clear in the rest of the paper, the actual interpretation of complementarity varies from one calculus to the other. The relevant fact to be pointed out here is that complementary actions are those that parallel processes can perform together to synchronize their (otherwise) independent behaviors.

A process is an elaboration unit that evolves by performing actions (a, b, ...) and co-actions (e.g. $\overline{a}$, $\overline{b}$, ...). To constraint the temporal order of the concurrent activities there is a limited set of operators.

Sequential ordering is rendered via the *prefix operator* written as an infix dot. For instance the term $a.\,\overline{a}.\,P$ denotes a process that may execute the activity a, then $\overline{a}$, and then all the activities modeled by P.

Two processes P and Q that run in parallel are represented by the infix *parallel composition* operator '|' as in $P \mid Q$. As anticipated, processes P and Q can either evolve independently or synchronize over complementary actions. For instance, the operational semantics of $\overline{a}.\,P \mid a.\,Q$ allows the inference of the following synchronizing transition:

$$\overline{a}.\,P \mid a.\,Q \longrightarrow P \mid Q\,.$$

The *restriction* operator is essential for the representation of encapsulation. In basic calculi like CCS, this operator, written (νa), is only meant to limit the

visibility of actions. For instance, is not possible to infer

$$\overline{a}.\, P \mid (\nu a)(a.\, Q) \longrightarrow P \mid (\nu a)Q$$

because a is a private resource of the right-hand process of the parallel composition and the left-hand process cannot interact on it. This fact guarantees, e.g., that the two processes R and S in $(\nu a)(R \mid S)$ have the opportunity to interact over a shared resource a without any interference by the external world.

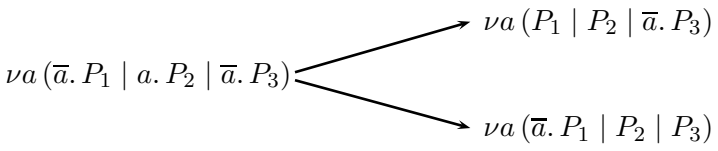
In more sophisticated calculi, as for instance in the π-calculus, the restriction operator ensures a relevant gain in expressiveness. The π-calculus allows channel names to be sent in interactions and hence the representation of *mobile* (i.e. dynamically changing) systems, because receiving new names means acquiring new interaction capabilities. This is what happens in biological networks, where the connections between nodes, and so the structure of the network, can change at any time.

In all the calculi for mobility, restricted names cannot be transmission media outside the scope of their definition implemented by the restriction operator. For example, no interaction over a can occur between P and Q or R in the process $P \mid (\nu a\,(Q \mid R))$, while Q and R can use their provate resource a to communicate. Restricted names can however be used as transmitted data and, once transmitted, become private resources shared by the sender and the receiver (hereafter we say public (private) names or channels to mean not restricted (restricted) names or channels). Suppose for instance that the restricted name a has been sent from Q to P in the above example. This is semantically rendered by a modification of the *scope* (i.e. the visibility) of the restricted name, yielding

$$P \mid \nu a\,(Q \mid R) \longrightarrow (\nu a)(P' \mid Q' \mid R)\,.$$

The peculiarity of the restriction operator of mobile process calculi has been extensively used in modeling biological behaviors. Since Q and R and then P', Q' and R can privately interact over a, if P, Q, P' are taken to represent molecules, then the processes $\nu a\,(Q \mid R)$ and $\nu a\,(P' \mid Q' \mid R)$ can be seen as the complexes of respectively two and three molecules.

We recalled only the fundamental operators which are common to various process calculi. Each calculus then adopts some specific operators and has a specific view about which activities must be considered complementary. For instance in CSP-like calculi (e.g. PEPA [31]) the interaction is not limited to be binary and the parallel composition is usually equipped with the set of channel names over which interaction can occur. Another common feature of process calculi is that their operational semantics allows the interpretation of process behaviors as graphs, called *transition systems*. The nodes of the graph represent processes, and there is an arc between the two nodes P and Q if P evolves into Q. For instance the immediate futures of $\overline{a}.\, P_1 \mid a.\, P_2 \mid \overline{a}.\, P_3$ is drawn as



The depicted transitions highlight that each of the processes at the left-hand and the right-hand sides can communicate with $a.\ P_2$. The evolution of the system depends upon the temporal order of the interaction. Since no assumption can be made on this, both transitions are reported in the graph, which is interpreted as a model of all the possible evolutions.

Process calculi are typically very simple, yet contain all the ingredients for the description of concurrent systems in terms of *what they can do* rather than of *what they are*.

Two main properties of process calculi are worth mentioning. First, the meaning (behavior) of a complex system is expressed in terms of the meaning of its components. A model can be designed following a bottom-up approach: one defines the basic operations that a system can perform, then the whole behavior is obtained by composition of these basic building blocks. This property is called *compositionality*. Second, the mathematical rules defining the operational semantics of process calculi allow both the automatic generation of the transition system of a given process by parsing the syntactic structure of the process itself and the simulation of a run of the system. So process calculi are specification languages that can be directly implemented and executed.

3 A few biological notions

Each of the languages that will be dealt with in the rest of this survey was adapted or developed to study a particular aspect, i.e. abstracts a specific set of features, of a biological system. In this section, we describe some biological notions in terms of a set of abstract 'biological primitives' that allow the relative comparison of the considered calculi.

3.1 Biochemical interactions

Living entities are constantly crossed by a flow of matter and energy. In this continuous random flow reactions take place whenever there is a sufficient quantity of kinetic energy [73]. For instance, a reaction between molecules A and B in Fig-

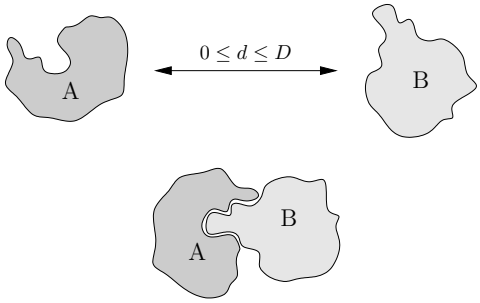


Figure 1: Molecular interaction

ure 1 may occur if A and B are close enough and correctly oriented. Normally the

frequency of reactions is quite low. By need, enzymes [12] may orient molecules in the right way favoring and speeding up the reaction.

Referring to the example in Figure 1, we observe that two molecules can bind if they possess complementary zones, called *domains*, and they have the right orientation (or, alternatively, the complementary domains are visible or available to each other).

The above conditions, however, are not enough. A domain of a molecule can be either *active* or *inactive*. An inactive domain can never bind, not even when a complementary domain with the right orientation is close to it. In order to activate a domain, a molecule needs to be involved in some reaction, for example in a phosphorylation (binding of a phosphate group to the protein). Active domains may further be free or bound. So domains are classified depending on three possible states: active bound, active free, and inactive. Figure 2 shows a schematic

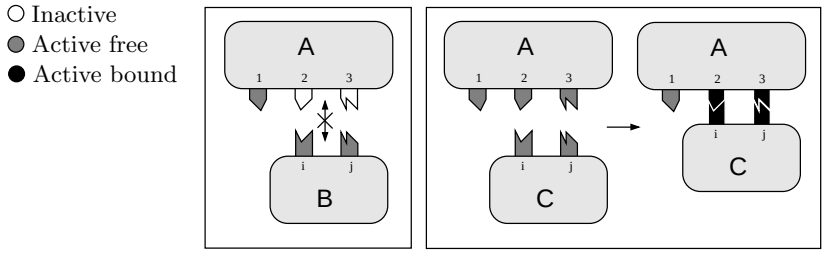


Figure 2: Interfaces, sites, and states

representation. Biological entities (named A, B, and C in the picture) possess an *interface* (the rounded box with colored hooks). Each interface has $n > 0$ sites (the hooks sticking out the rounded box), and each of them can be in one out of three states (the color of the hook) as said above. A site is an indivisible structure that can only join to a complementary site. In the scenario drawn in Figure 2, A cannot bind to B. In fact, sites 2 and i are complementary, just as 3 and j, but 2 and 3 are both inactive. On the contrary, A and C can bind together: sites i and 2, and j and 3 are pairwise complementary, and all of them are active free.

An interesting point is relative to the possibility of dynamically changing the number of sites available on a given interface or their states. In general, might be necessary to be able to add sites (see the running example in Sect. 5).

Finally, an important aspect of biological entities is their *shape*. Indeed two molecules can interact if they can get in touch accommodating their shapes. Consider for instance Figure 3(a) where entities A and B can interact through sites 1 and 2. The interaction can change the involved electrostatic forces. It can modify the shape of the new complex, and eventually make site 3 available for interaction (Figure 3(b)).

Concluding, an interaction site can be inactive either because a chemical reaction is needed to activate it, or because it is hidden by the three-dimensional structure (shape) of the hosting component.

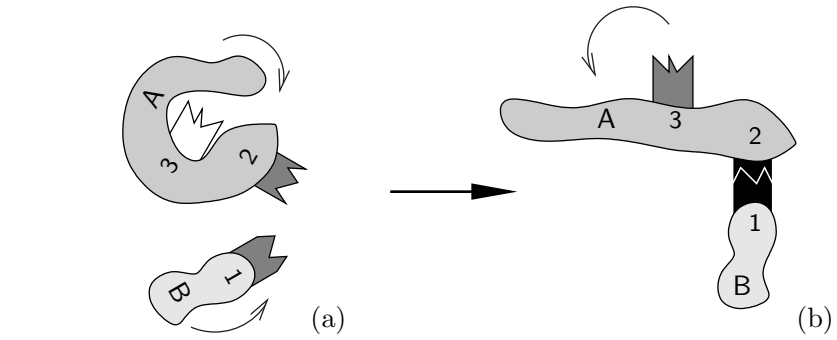


Figure 3: Shapes

3.2 Compartments

The same biochemical reaction in a different spatial context may have different outcomes. For instance, the bacterium Escherichia coli (E.coli) lives in the lower intestines of mammals performing useful functions involving the digestion of its host. If E.coli escapes the intestinal gap through a perforation (e.g. a hole from an ulcer), and enters the abdomen, it can cause an infection called 'peritonitis', fatal without immediate treatment. This example shows that it is important for a modeling language to allow the modeler to easily deal with compartments and with their possible modifications. These latest phenomena can be classified as endocytosis, exocytosis, break and merge.

- *Endocytosis and exocytosis.* Endocytosis consists in absorbing substances from the external environment. Endocytosis can be further distinguished in *pinocytosis* (assumption of liquids: no particle is absorbed except those contained in the liquid), *phagocytosis* (absorption of another component of comparable size), and *generalized endocytosis* (absorption of an arbitrary number of smaller components). Exocytosis is the opposite of endocytosis, i.e. the expulsion of sub-components.

- *Break and merge.* Break is used to model phenomena that imply a change in the boundary of a component. We consider *lysis*, *mitosis* and *meiosis*. Lysis is the disintegration of a cell following a damage of its plasma membrane. It makes free the biological material inside the membrane. Mitosis, typical of viruses, consists in the exact duplication of the cell. Meiosis, typical of reproductive cells, is the separation of the cell and of the contained genetic material and yields two new cells. The phenomenon opposite to break is called *merge*.

Figure 4 summarizes the four primitives which we define following the inspiration gained by the above observations. We call those primitives EXO, ENDO, BREAK and MERGE, respectively.

Finally, when dealing with the interaction of molecules within cells, the compartments might be thought of as the cellular compartments or the molecules. For

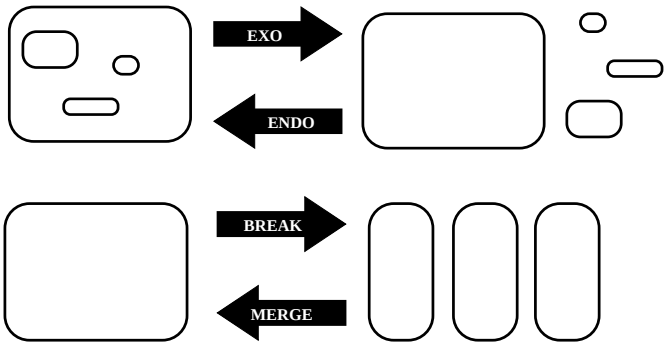


Figure 4: Compartment primitives

example, one important event that occurs within cells is the movement of small molecules across compartment membranes. Hence, we also consider the operations representing *uniport* (the movement of one molecule across a membrane), *symport* (the simultaneous movement of two molecules across a membrane in the same direction) and *antiport* (the simultaneous movement of two molecules across a membrane in opposite directions).

3.3 Further aspects

We now briefly sketch a set of interesting features of biomolecular processes which are independent on the concept of biochemical interactions and compartment and hence have not been presented yet. More details about the representability of these features in process calculi will be given later, when commenting on the various approaches used to face, respectively: reversibility; handling of quantitative information; and equivalence relations.

Reversibility

In nature many reactions are reversible. Reversibility is primarily governed by the kind of bonds that one wants to destroy and the available energy. Consider for instance the case when two proteins A and B are competing to bind to C (Figure 5). The system may evolve in three distinct ways. The case when both A and B bind to C gives raise to an unstable complex: sooner or later A or B will leave it. If A and B are, respectively, the activator and the inhibitor of C, and the global system is a molecular switch, then it is fundamental to be able to reverse the unstable complex.

Reversibility is a basic regulation mechanism that, for example, prevents deadlocks. It can be specified in process calculi either in an explicit way, by means of *ad hoc* behaviors, or implicitly, by means of backtracking mechanisms.

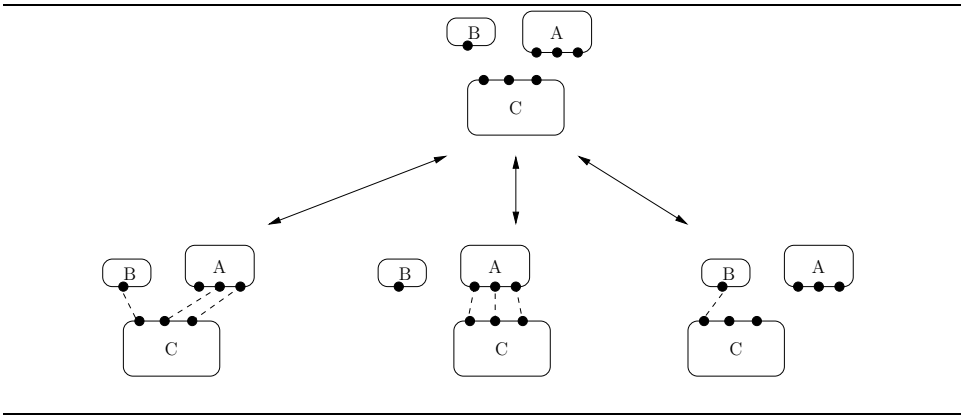


Figure 5: Reversible complexation

Quantitative information

The ability of reasoning about quantitative information plays a crucial role in biomolecular processes. For example, in order to correctly describe a reaction it is necessary to know the exact quantity of reactants involved, the affinity of the sites available for bonds, and the amount of energy which can be used. Representing and handling quantitative parameters typically results in both a formal and an implementation overhead. Nonetheless, this is a crucial point for building useful models of biological systems.

Equivalence

Two programs are usually considered equivalent in computer science if they exhibit the same behavior w.r.t. some chosen notion of observation. Different definitions of observation lead to distinct equivalence relations. The desired property is then that two equivalent components can be safely exchanged within a system without altering its overall behavior (if this property holds the equivalence turns out to be a congruence). An analogous situation is found in biology. Up to a certain equivalence relation, eukaryote cells and prokaryote cells might be seen as belonging to the same class of organisms. In order to relate distinct kinds of lymphocytes it would be surely necessary a finer grained notion of equivalence.

As far as systems described by terms of process calculi are concerned, equivalence relations are fundamental tools for both analysis and verification. It could well be the case that the techniques developed in concurrency theory may help in the formalization and the understanding of biomolecular relations. Further investigation is needed to relate the biological notions of relations (e.g., homology) and the computer science behavioral equivalences.

4 Process calculi abstraction principle

Abstraction is the mechanism of withdrawing information content from a knowledge domain in order to focus on the facts that seem most relevant for a particular purpose. Computer science deeply leans upon abstraction. For instance, computers work by simply checking presence or absence of electronic signals at physical level. With a similar capability it becomes even difficult to do simple operations like

$$var := var + 3 \qquad (1)$$

that increments the value of a variable var. Hence suitable abstractions are needed to make it easy give instructions to computers ans abstractions can be arranged hierarchically (firmware, assembler, operating systems, high-level programming languages, etc.). For instance, the low level steps required for (1) are quite complex: load the value of var into a registry, convert 3 to binary representation, decompose calculation into assembly instructions and so on. Moreover, the resulting program heavily depends on the underlying hardware architecture. High level languages (e.g. C++, Java) allow to forget (i.e. abstract away) implementation details and to focus on the programming activity. This approach boosted computer science over the last 40 year. The idea is to exploit the same principle in systems biology.

Biology	**Process calculi**
Entity	Process
Interaction capability	Channel
Interaction	Communication
Modification/evolution	State change

Table 1: Process calculi abstraction for systems biology

Assume to know the basic mechanisms that 'drive' life. If we are able to design a low level language that embeds the above mechanisms, we can explore biological hierarchies through compilation and then use living matter as our hardware infrastructure. In a similar vision, language theory is a conceptual formal tool that enables biologists to reassemble fragmented knowledge into a whole biological system via computational thinking. Process calculi play the role of low level languages, because their theory coincides at some extent with an abstraction of molecular interactions. Table 1 gives a concise picture of the map between biology and process calculi. A biological entity (e.g., a protein) is seen as a computation unit, a process, with interaction capabilities abstracted as channel names. Entities interact/react through complementary capabilities as processes communicate/synchronize on complementary actions. The change of a state after a communication represents the modification/evolution of molecules after a reaction.

The abstraction in Table 1 has four main properties [65]: (*i*) it captures an essential part of the phenomenon; (*ii*) it is computable, or better, it is executable, allowing computer aided analysis; (*iii*) it offers a formal framework to reason; (*iv*) it can be extended. In the rest of the paper we will show the main process calculi proposed for representing biological systems, and we will show how each one focuses on a particular extension of the abstraction principles in Table 1.

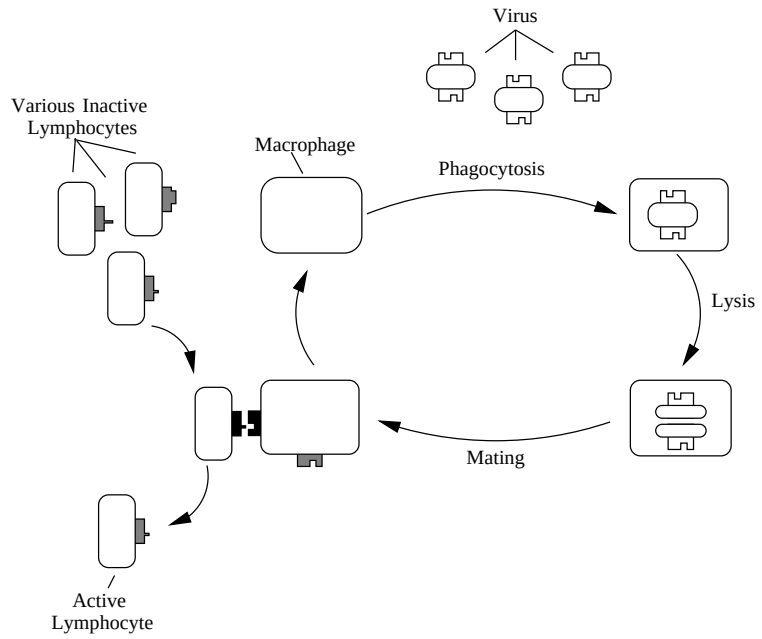


Figure 6: Lymphocyte T helper activation

5 Running Example

This section introduces the running example that will be used to present and compare the considered calculi in the biology applicative domain. The example comes from the biology of the immune system, and it is relative to the activation of the *lymphocyte T helper*.

The example has been chosen taking in mind two essential factors: (i) the example has to be sufficiently complex to be an interesting case study for representation issues; (ii) it must be abstract enough to allow independence from irrelevant biological details. These features ensure that a short description of the phenomenon can make it easily accessible to readers without any specific background in biology.

Lymphocytes T are eukaryote cells belonging to our immune system. There are three different sorts of lymphocytes T: lymphocytes T helper, lymphocytes T suppressor, and lymphocytes T cytotoxic. The lymphocytes of the first two classes are the main controllers of the immune system. Lymphocytes T cytotoxic work against foreign eukaryote cells and against cells of the body which have been infected by a virus.

Lymphocytes are normally inactive, and start their activity only after being triggered by special events. Each class of lymphocytes may be activated in many ways. We will focus on the activation of lymphocytes T helper performed by macrophages.

Macrophages are cells belonging to our immune system. They can engulf a virus by endocytosis, and, when this happens, the virus is degraded into fragments and a molecule (*antigen*) is displayed on the surface of the macrophage. The anti-

gen may be recognized by a lymphocyte T helper, and this activates the mechanisms of the immune reply, like, e.g., the duplication of the lymphocyte. Notice that the phenomenon includes the following relevant ingredients:

- pattern recognition, that allows the macrophage to distinguish malicious antigens and to activate the appropriate T cell;
- membrane interactions, that allow the macrophage to engulf viruses and to express antigens;
- internal pathways, that lead to the digestion of the engulfed viruses.

Figure 6 gives an abstract representation of the phenomenon described above. Viruses are modeled as entities with inactive sites which represent the viral antigens, i.e., the molecules characterizing the viruses. The process starts with the phagocytosis (ENDO) of the virus by the macrophage. The virus is then decomposed (BREAK), and eventually viral antigens are moved to the surface of the macrophage. So the macrophage acquires some active sites from the virus, and can wait for a lymphocyte with a complementary site. When the appropriate lymphocyte T helper binds to the macrophage, it becomes active and starts playing its role in the immune reply. Observe that lymphocytes have active sites even before binding to a macrophage. Indeed, even if in this state lymphocytes are inactive, as all in the immune reply, no binding would be possible without active sites.

6 Calculi for biology

In this section we survey the main calculi for biology which have been proposed in the literature. For each calculus, we first consider the representation of the compartments, and then we refine the model at the biochemical level. We will exploit short portions of code for the activation of lymphocytes T helper. Finally, for each calculus we will comment on the expressivity w.r.t. the biological requirements proposed in Sect. 3, and on the availability of software.

6.1 Biochemical stochastic π-calculus

The *biochemical stochastic π-calculus* [66, 62] represents biochemical systems of interacting molecules as mobile communicating processes of the π-calculus [48, 69]. Public channel names and co-names represent complementary sites and cellular compartments are rendered by the appropriate use of restrictions on channels. Molecular interaction is modeled as communication, and the stochastic extension of the π-calculus [60] is adopted to provide quantitative descriptions of systems.

6.1.1 Syntax and Semantics

The π-calculus is a name-passing process calculus where names are synonyms of both data and channels. Its biochemical stochastic extension represents molecules as computational processes. A molecular complex is a system of processes sharing a private name which is unknown outside the complex. In this way, a molecule

which is external to the complex can by no means have access to the complex. The scope of the private name represents the boundary of the complex. Movements between complexes and formations of new complexes are represented by transmitting private names (the so-called name extrusions).

The biochemical stochastic π-calculus represents interfaces by means of communication channels. Components interact by communicating on complementary sites, and interfaces may be possibly modified as a result of the communication.

6.1.2 Example

System specification

SYS::= MACROPHAGE | VIRUS | TCELL1 | TCELL2

MACROPHAGE::=(νMemM)($\overline{\mathsf{Tlr}}\langle$MemM$\rangle$. MemM(a). $!\,\overline{\mathsf{a}}\langle$str$\rangle$)

VIRUS::=(νMemV)(Tlr(y).$\overline{\mathsf{y}}\langle$Ant1$\rangle$)

System evolution

MACROPHAGE | VIRUS $\rightarrow$

(νMemM) (MemM(a).$!\,\overline{\mathsf{a}}\langle$str$\rangle$ | (νMemV)($\overline{\mathsf{MemM}}\langle$Ant1$\rangle$))$\rightarrow$

(νMemM)($!\,\overline{\mathsf{Ant1}}\langle$str$\rangle$)

Figure 7: Phagocytosis-Digestion-Presentation in π-calculus

Compartments. Figure 7 reports the a code fragment that encodes the antigen presentation phase. The global system SYS is given by the parallel composition of four processes: VIRUS, MACROPHAGE, TCELL1, and TCELL2. Figure 7 only presents the specifications of the first two elements. Here we just sketch the intuition of the behavior of the sub-system given by MACROPHAGE | VIRUS. The restriction on top of each component stands for its enclosing membrane. The macrophage phagocytes the virus by means of a communication on the public channel Tlr. Operationally, this communication involves the output action $\overline{\mathsf{Tlr}}\langle$MemM$\rangle$ and its complementary input action Tlr(y), and its effect is twofold: *(i)* the restricted name MemM undergoes a scope extrusion and becomes a private resource of both MACROPHAGE and VIRUS (thus modeling the engulfment of the virus); *(ii)* the name y in VIRUS is renamed into MemM (modeling the adaption of the internal machinery of the macrophage to start the lysis). The subsequent communication over the channel MemM is such that the datum Ant1 is transmitted to MACROPHAGE, which can make it available to lymphocytes T (TCELL1, TCELL2) by means of the latest action $!\,\overline{\mathsf{Ant1}}\langle$str$\rangle$. The operator *bang*, !, allows to model infinite behaviors. In particular, $!\,\overline{\mathsf{Ant1}}\langle$str$\rangle$ behaves as $\overline{\mathsf{Ant1}}\langle$str$\rangle$. ($!\,\overline{\mathsf{Ant1}}\langle$str$\rangle$), and therefore MACROPHAGE can activate many TCells expressing Ant1.

Biochemical interactions. Figure 8 shows the implementation of the activation of the appropriate lymphocyte T helper. Assume that the antigen presentation phase

already occurred, and hence that the macrophage is ready to communicate on channel Ant1 with whichever lymphocyte can execute a complementary action on the same channel. In the evolution drawn in Figure 8 this lymphocyte is TCELL1 which, after the synchronization on Ant1, can start its activities. Notice that the active form of the macrophage, MACROPHAGE', can activate another TCell because of the bang operator.

System specification

SYS::=MACROPHAGE' | TCELL1 | TCELL2

MACROPHAGE'::=(νMemM) (! $\overline{\text{Ant1}}\langle\text{str}\rangle$)

TCELL1::=(νMemT1)(Ant1(x).ACTIVITIES)

TCELL2::=(νMemT2)(Ant2(x).ACTIVITIES)

System evolution

SYS $\rightarrow$

MACROPHAGE' | (νMemT1)(ACTIVITIES) | (νMemT2)(Ant2(x).ACTIVITIES)

Figure 8: TCELL activation in π-calculus

6.1.3 Comments

The biochemical stochastic π-calculus represents biochemical interactions as communications, yielding models of biological pathways which are both detailed and concise. In the biochemical stochastic π-calculus there is no explicit concept of compartments. To represent the operations on compartments (EXO, ENDO, BREAK and MERGE), the non-intuitive concepts of restriction and name passing must be used.

There exist implementations of the biochemical stochastic π-calculus that make real in silico experiments possible. Two examples of simulation tools for biochemical stochastic π-calculus are BioSpi [72] and SPiM [57], both based on the Gillespie's algorithm [25]. Several complex models of real biochemical systems have been implemented and simulated by using these tools. Notably, the simulation of extra-vasation in multiple sclerosis reported in [45] showed to have a sort of predictive flavor: an unexpected behavior of leukocytes has been guessed by the results of in silico simulations, and proved a posteriori in lab experiments. Also, a whole virtual cell (VICE), with a basic prokaryote-like genome (about 180 different genes) is developed with interesting results: for instance, the distribution of metabolites along the glycolytic pathway of VICE significantly matches with those of real organisms [14].

To overcome the intrinsic difficult in π-calculus, due to its minimal sintax, some eorts are devoted to design higher-level languages that provides direct support for the concepts needed in modeling biological systems, as e.g. [21] that leads to a complex model of gene regulation [44].

6.2 BioAmbients

BioAmbients [64] focuses on compartments: the location of molecules within specific compartments is considered a key issue for regulatory mechanisms in biological systems. Biomolecular systems are organized in a hierarchical and modular way: a molecule can perform its task if and only if it is in the right compartment.

6.2.1 Syntax and Semantics

Ambients are the boundaries of a set of processes which can communicate with each other. Ambients can be nested and hence they are organized hierarchically.

As for the π-calculus, we do not provide a full description of the calculus. We rather focus on a few main features which are useful for our presentation. The reader is referred to [64] for full details.

Depending on the relative locations of the interacting processes, three kinds of communication are defined in BioAmbients:

- local, namely between two processes in the same ambient,
- s2s, namely between two processes located in sibling ambients,
- p2c / c2p, namely between processes located in ambients with a parent-child relation.

As far as the interpretation of movements is concerned, three pairs of primitives are defined as process actions:

- enter n / accept n, for entering an ambient and accepting the entrance, respectively,
- exit n / expel n, for exiting from a containing ambient and expelling a contained ambient, respectively,
- merge+ n / merge- n, for merging two ambients together.

6.2.2 Example

Compartments. Figure 9 shows a possible specification of the digestion of the virus by the macrophage. The two processes Infect and Digest abstract the infection capability of the virus and the digestion capability of the macrophage, respectively. Virus and macrophage synchronize on channel tlr, and the virus enters the macrophage by an enter / accept pair. Then the virus sends its antigen on channel tlr, and eventually the macrophage makes the antigen available to lymphocytes T helper.

Biochemical interactions. BioAmbients uses communication channels to implement 'interfaces' of biological entities (as the biochemical stochastic π-calculus). The BioAmbients implementation of the activation of the lymphocyte T helper by a macrophage is reported in Figure 10. Each lymphocyte reacts to a specific antigen and begins its task by means of a communication on a dedicated channel. After the right lymphocyte has been activated, the macrophage can activate other TCells, because of the bang operator !.

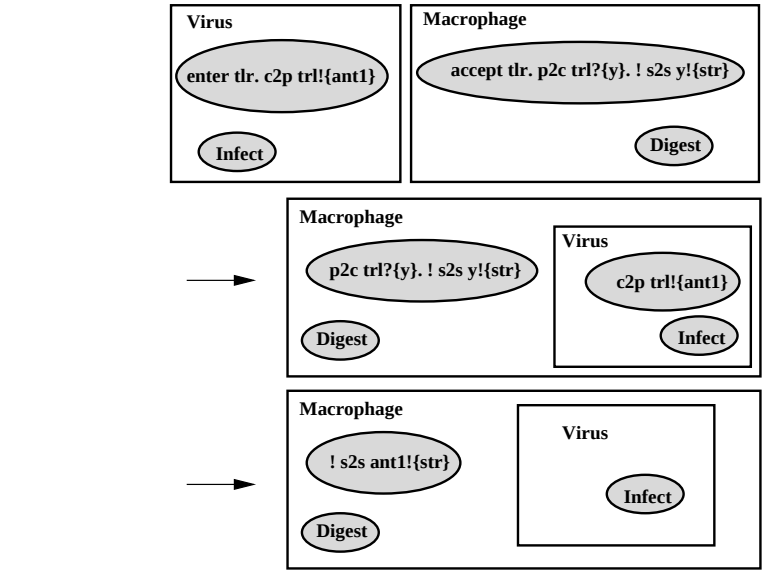


Figure 9: Phagocytosis-Digestion-Presentation in BioAmbients

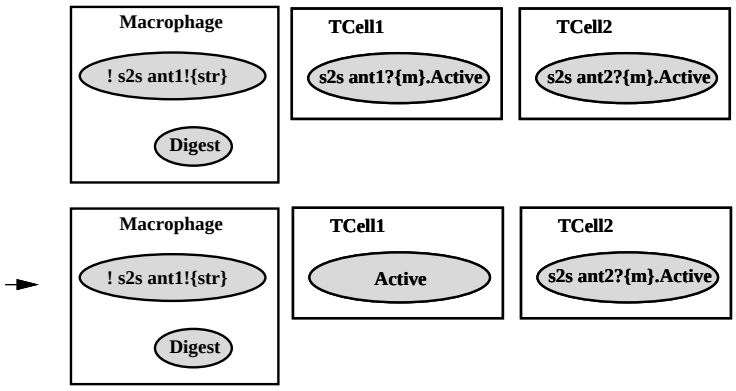


Figure 10: TCELL activation in BioAmbients

6.2.3 Comments

BioAmbients models biochemical interactions as communications on channels. It extends biochemical stochastic π-calculus communications with three kind of communications: interactions can occur between entities in the same compartment (*local* communication), between actions lying in ambients within the same ambient (*s2s* communication), and between father-child ambients (*c2p/p2c* communication). BioAmbients is the first process calculus for modeling biological systems in which an explicit and intuitive notion of compartments is considered. It is easy to model in BioAmbients operations as EXO, ENDO and MERGE. BioAmbients has no primitive to represent the splitting of environments (BREAK), and it is not straightforward to model, e.g., mitosis. Other operations that can be easily modeled are complex formation and transport of small molecules across compartments. Hence, BioAmbients may represent, in this respect, an improvement compared to biochemical stochastic π-calculus: many biological phenomena are represented much more easily in BioAmbients than in π-calculus.

A stochastic extension of the language has been defined, and a simulator is implemented as part of the BioSpi project [72] based on Gillespie's algorithm [25]. Control Flow Analysis, a static analysis technique that allows to analyze the description of the system to discover dynamic properties, is adapted to BioAmbients [50].

6.3 Brane Calculus and Projective Brane Calculus

Brane calculus [9, 10] is centered on membranes, and it is based on the observation that membranes are not just containers, but also active entities that take care of coordinating specific activities. Membranes can be highly dynamic: for example, they can shift or merge. Molecules can communicate using their membranes, and indeed large proteins are embedded in membranes which act like channels.

The main feature of Brane calculus is that membranes are considered active elements and hence the whole computation happens *on* membranes. In Brane calculus membranes can move, merge, split, enter into and exit from other membranes. Some constraints need to be satisfied when applying these operations. The most important one is that transformations need to be continuous (e.g. a membrane, except the case it represents a small molecule, cannot simply pass across another membrane). Another constraint is that the orientation of membranes need to be preserved, so merging of membranes cannot occur arbitrarily (e.g. membranes with a different orientation cannot merge).

6.3.1 Syntax and Semantics

A system is represented in Brane calculus as a set of nested membranes, and a membrane as a set of actions; actions carry out the mentioned membrane transformations. The Brane calculus primitives are inspired to membrane properties. Because of the constraints on membrane operations, Brane calculus primitives are more restrictive than those we presented in Sect. 3.2. The primitives related to movement to and from membranes are classified in two main groups, one for cytosis-like and the other for mitosis-like phenomena.

- Endocytosis, corresponding to the ENDO operation, is considered an uncontrollable process. Interesting interactions are usually more controllable, therefore two finer primitives are defined: phagocytosis (phago), for engulfing one external membrane, and pinocytosis (pino), for engulfing zero external membranes. Exocytosis (exo), which corresponds to the EXO operation, represents the expulsion of an internal membrane.

- Mitosis, corresponding to the BREAK operation, is also considered an uncontrollable process, because it can split a membrane at an arbitrary place. Hence, two finer primitives are defined: budding (bud), for splitting off one internal membrane, and dripping (drip), for splitting off zero internal membranes. Mating (mate), which corresponds to the MERGE operation, represents the controlled merging of two membranes.

For each action, a corresponding co-action is defined. Hence, as in BioAmbients, coordination between interacting components is always required. Communication can be *on-membrane* or *cross-membrane*, and they are associated with distinct pairs of primitives.

- *On-Membrane*. The primitives $\mathsf{p2p}_n$ / $\mathsf{p2p}_n^{\perp}$ are for on-membrane communications only.

- *Cross-Membrane*. The primitives $\mathsf{s2s}_n$ / $\mathsf{s2s}_n^{\perp}$, $\mathsf{p2c}_n$ / $\mathsf{p2c}_n^{\perp}$, and $\mathsf{c2p}_n$ / $\mathsf{c2p}_n^{\perp}$ are for communications between processes in distinct membranes and follow a BioAmbients-like style.

Projective brane calculus. This calculus, which has been introduced in [18], is a refinement of Brane calculus. Its authors observe that in real life biological membrane actions are directed; therefore they refine brane calculus by replacing actions with directed actions, so that interaction capabilities are specified as facing inwards or outwards. This refinement results in an abstraction which is closer to biological settings than the original language.

6.3.2 Example

Compartments. Figure 11 reports a specification of the running example in Brane calculus. The operator $\circ$ stands for parallel composition. The rounded parentheses $(\!|\ |\!)$ enclose the genetic content of the membrane, which is represented by a sequence of actions to the left of the symbol $(\!|$.

During the synchronization over phago, the virus communicates its antigen to the macrophage on the channel *trl*. Then the macrophage presents the antigen to the environment. As mentioned in Sect. 6.2, in real life a virus does not actually send its antigen to a macrophage.

Biochemical interactions. Figure 12 shows the Brane calculus code for the activation of the appropriate lymphocyte.

System specification

SYS::= VIRUS ∘ MACRO ∘ TCELL1 ∘ TCELL2

VIRUS::= phago_{trl}. $\mathsf{c2p}_n$(ant1).INFECT (|CAPSID|)

MACRO::= $\mathsf{phago}^{\perp}_{trl}$(DIGEST).$\mathsf{c2p}^{\perp}_n$(a).$\mathsf{s2s}_a$(str) (|CYTOSOL|)

DIGEST::= $\mathsf{c2p}^{\perp}_n$(a).$\mathsf{c2p}_n$(a).DIGEST

System evolution

MACRO ∘ VIRUS →

$\mathsf{c2p}^{\perp}_n$(a).$\mathsf{s2s}_a$(str) (|DIGEST (|$\mathsf{c2p}_n$(ant1).INFECT (|CAPSID|)|) ∘ CYTOSOL|) →

$\mathsf{c2p}^{\perp}_n$(a).$\mathsf{s2s}_a$(str) (|$\mathsf{c2p}_n$(ant1).DIGEST (|INFECT (|CAPSID|)|) ∘ CYTOSOL|) →

$\mathsf{s2s}_{ant1}$(str) (|DIGEST (|INFECT (|CAPSID|) |) ∘ CYTOSOL|)

Figure 11: Phagocytosis-Digestion-Presentation in Brane calculus

System specification

SYS := MACRO ∘ TCELL1 ∘ TCELL2

MACRO::= $\mathsf{s2s}_{ant1}$(str).PHAGO (|CYTOSOL|)

TCELL1::= $\mathsf{s2s}^{\perp}_{ant1}$(x).ACTIVITIES1(|CYTOSOL|)

TCELL2::= $\mathsf{s2s}^{\perp}_{ant2}$(x).ACTIVITIES1(|CYTOSOL|)

System evolution

SYS → PHAGO(|CYTOSOL|) ∘ ACTIVITIES1(|CYTOSOL|) ∘ TCELL2

Figure 12: TCELL activation in Brane calculus

6.3.3 Comments

In Brane calculus everything is interpreted as a membrane, which means that membrane-bound cellular compartments (e.g. cells and organelles) and molecular compartments (e.g. proteins) are modeled in the same way. The language does not take the internal structure of membrane-bound compartments into account, therefore it is not easy to describe biochemical events that are not directly related to cellular membranes, such as protein activation, phosphorylation, etc.

Brane calculus is inspired by BioAmbients, but it gives membranes an active role. The notion of membrane as an active entity and not just a simple container is surely relevant. In addition, Brane calculus primitives are realistic and provide a simple and intuitive way to model the most important membrane operations. Being Brane calculus primarily concerned on membrane interactions, it is possible (and also relatively easy) to model all kinds of operations involving compartments (EXO, ENDO, MERGE, BREAK) and also movements of small molecules across membranes. No software tool is available for this calculus.

6.4 CCS-R

CCS-R [15] explicitly deals with the issue of reversibility: most biochemical reactions are, indeed, reversible. Based on this observation, CCS-R is a CCS-like process algebra [47] with the peculiarity that reversibility is embedded in the syntax. No handling of energy and types of bonds is considered, although they are the driving forces of biomolecular reversibility.

6.4.1 Syntax and Semantics

CCS-R is a 'decoration' of CCS with the concept of reversibility. This feature of the language is relevant when considering biochemical scenarios. Regarding the description of compartments, instead, CCS-R may be considered the same as CCS: a process algebra that describes the interactions between processes in terms of binary synchronized communications and does not use either value or name-passing. In CCS-R it is not possible either to send a name on a channel or to dynamically change the scope of a restricted name. For this reason, compartments and information flows between processes cannot be represented.

CCS-R generalizes CCS duality between names and co-names to a binary complementation relation $\mathcal{C}$ between binding sites: the two sites x and x' can connect together only if $x\mathcal{C}x'$. Moreover, based on the observation that some protein interactions require a concurrent connection to different sites, the standard CCS syntax is extended to allow processes like C= (l'$_1$ | l'$_2$ | l'$_3$).0 to represent the fact that the sites of C must be activated simultaneously. Apart from these differences, CCS-R is the same as CCS. In particular, for what concerns our example, the system specifications in the two languages are identical.

6.4.2 Example

System specification

SYS::=(νTlr)(νAnt1)(νAnt2)(VIRUS | MACROPHAGE | TCELL1 | TCELL2)

VIRUS::=Tlr.$\overline{\text{Ant1}}$.INACT

MACROPHAGE::= $\overline{\text{Tlr}}$.DIGEST

System evolution

SYS $\rightarrow$ (νTlr)(νAnt1)(νAnt2)($\overline{\text{Ant1}}$.INACT | DIGEST | TCELL1 | TCELL2)

Figure 13: Phagocytosis-Digestion-Presentation in CCS-R

Compartments. As previously mentioned, compartments cannot be represented in CCS-R, and the information flow from the virus to the macrophage cannot be faithfully rendered. It is still possible, though, to specify the antigen presentation phase as a pathway activation. This kind of coding is used in the specification

of the running example in CCS-R, shown in Figure 13. Notice that, in order to overcome the fact that the position of restrictions cannot change at run-time, the relevant resources have to be declared as private channels of the top-level process SYS.

System specification

SYS::=(νTlr)(νAnt1)(νAnt2)(VIRUS | MACROPHAGE | TCELL1 | TCELL2)

MACROPHAGE::= DIGEST

VIRUS::=$\overline{\text{Ant1}}$.INACT

TCELL1::=Ant1.ACTIVITIES1

TCELL2::=Ant2.ACTIVITIES2

System evolution

SYS $\rightarrow$

(νTlr)(νAnt1)(νAnt2) (INACT | DIGEST | ACTIVITIES1 | Ant2.ACTIVITIES2)

Figure 14: TCELL activation in CCS-R

Biochemical interactions. CCS-R is particularly suitable to represent biochemical interactions. However, since it does not use a name-passing discipline, it is impossible to directly render the information flow between processes and its subsequent changing of the possible evolution of the communicating partners. With respect to the previous code fragments, let us consider the antigen passing from the virus to the macrophage and, finally, to the right TCELL: in the CCS-R specification for lymphocyte activation (Figure 14), the virus (rather than the macrophage) is responsible for activating the right TCELL.

6.4.3 Comments

CCS-R, being based on CCS, allows to represent biochemical pathways as a cascade of synchronized interactions. CCS-R does not allow name passing or a direct representation of biological bounds, therefore modeling compartment is not directly supported.

CCS-R has primarily been developed to implement reversibility in a process calculus for biology. The authors of CCS-R think of reversibility as the ability to backtrack from a reaction and claim that this is a common phenomenon in nature. This is true, however, only if the energy of the system is not considered. Indeed the second principle of thermodynamics states that going back exactly to the original system is not possible. This principle could become crucial if reversibility was investigated together with a quantitative analysis of the global energy of the biological system. No software tool is available for this calculus.

6.5 PEPA

Performance Evaluation Process Algebra (PEPA) [31] is a formal language for describing Markov processes. PEPA was introduced as a tool for performance analysis of large computer and communication systems to study in the same framework quantitative properties as throughput, utilization and response time and qualitative properties as deadlock freedom. With the advent of the systems biology era, the abstraction facilities of PEPA was exploited in biochemical signaling pathways analysis and simulation [8].

6.5.1 Syntax and Semantics

PEPA differs from the previous calculi because it adopts multiway synchronization on shared name in the style of the Communicating Sequential Processes (CSP) [33] rather than complementarity (as CCS and π-calculus). The cooperation operator $\underset{\mathcal{L}}{\bowtie}$ requires the "co-operands" to join for activities specified in the cooperation set $\mathcal{L}$. Consider a simple biochemical reaction r_1 where two protein $Prot1$ and $Prot2$ interact with a rate k_1 and form a protein $Prot3$. The system is specified by the following:

$$Prot1_H ::= (r1, k1).Prot1_L$$
$$Prot2_H ::= (r1, k1).Prot2_L$$
$$Prot3_L ::= (r1, \top).Prot3_H$$
$$Sys ::= Prot1_H \underset{\{r1\}}{\bowtie} Prot2_H \underset{\{r1\}}{\bowtie} Prot3_L$$

The subscript H and L stay for high and low level of protein. The three protein can synchronize on activity $r1$ enabling the transition:

$$Sys \xrightarrow{(r1,k1)} Prot1_L \underset{\{r1\}}{\bowtie} Prot2_L \underset{\{r1\}}{\bowtie} Prot3_H$$

where low levels of $Prot1$ and $Prot2$ are present and an high level of $Prot3$ is reached. The multi-way synchronization underlying PEPA allows the three processes to advance in one step. This is the main difference of PEPA w.r.t. CCS/π-calculus style of interaction.

6.5.2 Example

Compartments. PEPA cannot represent directly compartments or inderectly by means of scope extrusion. The information flow from the virus to the macrophage on Trl cannot be faithfully represented, but, by specifying the antigen presentation phase as a biochemical interaction, it is still possible . The approach is the same used in CCS-R, and Figure 15 shows the related PEPA code. Macrophage MACROP synchronize with VIRUS on activity Tlr enabling Ant1 presentation with a rate k.

Biochemical interactions. PEPA represent biochemical interactions as cooperation, meaning that processes jointly perform actions of the same type. However, PEPA has not name-passing features, and therefore PEPA does not allow to directly represent information flow and subsequent changing of the interaction capabilities. Figure 16 sketches PEPA code for TCell activation. The virus (rather than the

System specification

SYS::=VIRUS $\underset{\{Tlr\}}{\bowtie}$ MACROP

VIRUS::= (Tlr,k) . (Ant1,k_{A1}) . INACT

MACROP::= (Tlr,k) . DIGEST

System evolution

SYS $\xrightarrow{(\text{Tlr},k)}$ (Ant1,k_{A1}) . INACT $\underset{\{Tlr\}}{\bowtie}$ DIGEST

Figure 15: Phagocytosis-Digestion-Presentation in PEPA

System specification

SYS::= VIRUS' $\underset{\{Ant1,Ant2\}}{\bowtie}$ MACROP' $\underset{\{Ant1,Ant2\}}{\bowtie}$ TCELL1 $\underset{\{Ant1,Ant2\}}{\bowtie}$ TCELL2

MACROP'::= DIGEST

VIRUS'::=(Ant1,k_{A1}) . INACT

TCELL1::=(Ant1, k_{A1}) . ACTIVITIES1

TCELL2::=(Ant2, k_{A2}) . ACTIVITIES2

System evolution

SYS $\xrightarrow{(\text{Ant1},k_{A1})}$

INACT $\underset{\{Ant1,Ant2\}}{\bowtie}$ MACROP' $\underset{\{Ant1,Ant2\}}{\bowtie}$ ACTIVITIES1 $\underset{\{Ant1,Ant2\}}{\bowtie}$ TCELL2

Figure 16: TCELL activation in PEPA

macrophage) synchronize with the TCell with the right antigene. Notice that $Ant2$ is in the cooperation set $\{Ant1, Ant2\}$ because otherwise TCELL2 can proceed without recognizing the right antigene.

6.5.3 Comments

PEPA was introduced as a tool for performance analysis. Its application to systems biology allows to *quantitatively* model and analyze large pathway systems (e.g., [7]). However, PEPA lacks in expressivity of compartment primitives.

PEPA has two main characteristics that makes it very interesting, also for biology: *(i)* a large community supporting it [35] with a reach availability of software tools. For instance, the PEPA Workbench [26] allows to exploit Markov process analysis on PEPA specification. Moreover, external tools support PEPA. For instance, the PRISM model checker [32] accepts model descriptions in the PEPA formalism. *(ii)* PEPA is a language for describing Markov processes, and therefore PEPA is developed with a strong mathematical background. This enables the

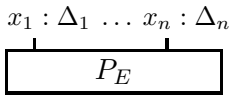
comparison and convergence of process calculi models and classical ODE modelling [5, 6].

6.6 Beta-binders

Beta-binders [61] is a bio-inspired process calculus that interprets biological entities as an internal "process unit" and an "interface" exposed to the external environment. By introducing the concept of *affinity*, the interaction approach extends the CCS notion of complementarity between action and coation. Beta-binders communication models is inspired by enzyme theory [43], where interactions between not perfectly matching components are allowed.

6.6.1 Syntax and Semantics

In Beta-binders, π-calculus processes are encapsulated into *boxes* with interaction capabilities. The π-calculus syntax is enriched by operations for manipulating interaction capabilities, that are represented by specialized binders. Any biological entity E is represent as a box B_E



Types Δ_i express the interaction capabilities of the box. The parallel composition of boxes, called *bio-process*, models a system of interaction biological entities. Two boxes can interact if they have complementary types up to a certain *user-defined* notion (see [59] for an example). Here we adopt the original interpretation, where types are sets of names and two types Δ_1 and Δ_2 are affine if $\Delta_1 \cap \Delta_2 \neq \emptyset$. The dynamic behavior of entity box $B_E$is specified through the internal pi-process P_E. A pi-process is a π-calculus process, extended for manipulating the interface of a box. For instance, *hide* and *unhide* actions make respectively invisible and visible an interaction site, allowing the direct representation of dephosphorilation and phosphorilation. Finally, two boxes can bring together (*join*) and one box in two can divide in two (*split*).

6.6.2 Example

Compartments. Figure 17 reports the Beta-binders fragment that encodes the antigen presentation phase. The global system SYS is given by the parallel composition of four boxes representing the VIRUS, the MACROPHAGE, the TCELL1, and the TCELL2, respectively. Figure 17 only presents the specifications of the first two elements and we just sketch an explanation of the behavior of the subsystem given by the MACROPHAGE and the VIRUS. The macrophage phagocytes the virus by means of a join operation. This results in a box whose interaction capabilities are inherited from MACROPHAGE, and whose body is essentially given by the parallel composition of the internal bodies of the original boxes P_{MACRO} and P_{VIRUS}. After that, virus and macrophage are in the same box, so they can communicate, and the name Ant_1 is transmitted to MACROPHAGE, which can make it available to lymphocytes T (TCELL1, TCELL2) by means of the latest output action $\overline{\mathsf{Ant1}}\langle \mathsf{str} \rangle$.

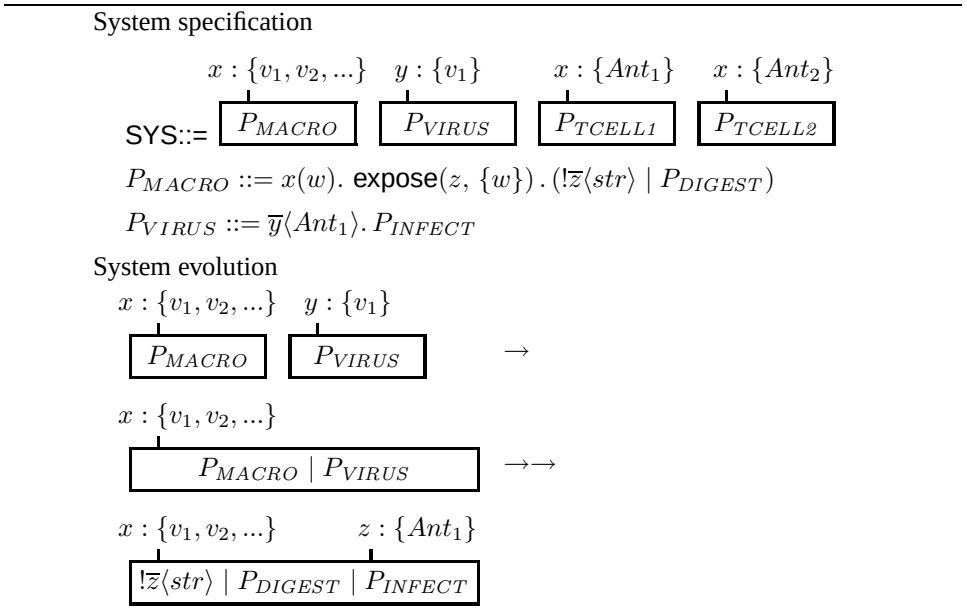


Figure 17: Phagocytosis-Digestion-Presentation in Beta-binders

Biochemical interactions. Figure 18 shows the implementation of the activation of the appropriate lymphocyte T helper. We imagine that the antigen presentation phase already occurred, and hence the macrophage is ready to execute an inter-communication on channel with type is $Ant1$ with whichever lymphocyte can execute a complementary action on a channel with a compatible type. In the system of Figure 18 this lymphocyte is TCELL1 which, after the interaction, can start its activities.

6.6.3 Comments

Beta-binders was specifically designed to model biological interactions. The main peculiarity of Beta-binders is the concept of affinity, which allows not perfectly matching components to interact. This is often the case in biology, where the interaction sites of proteins can be compatible even if not exactly complementary. Biochemical events that are not directly related to cellular membranes (e.g. protein activations, phosphorylations, etc.) can be easily modeled by Beta-binders communications and operations on box interfaces.

Another interesting feature of Beta-binders is that operations such as fusion of membranes and splitting of one membrane into two submembranes, can be easily modeled by means of the appropriate join and split primitives. However, when dealing with compartments one main drawback of Beta-binders arises: nesting of boxes is not allowed, so it is not intuitive to model hierarchies of entities. In [28] an extension of Beta-binders with an explicit notion of compartments is introduced. This extension permits to represent static hierarchical structures and the movement of components across compartments.

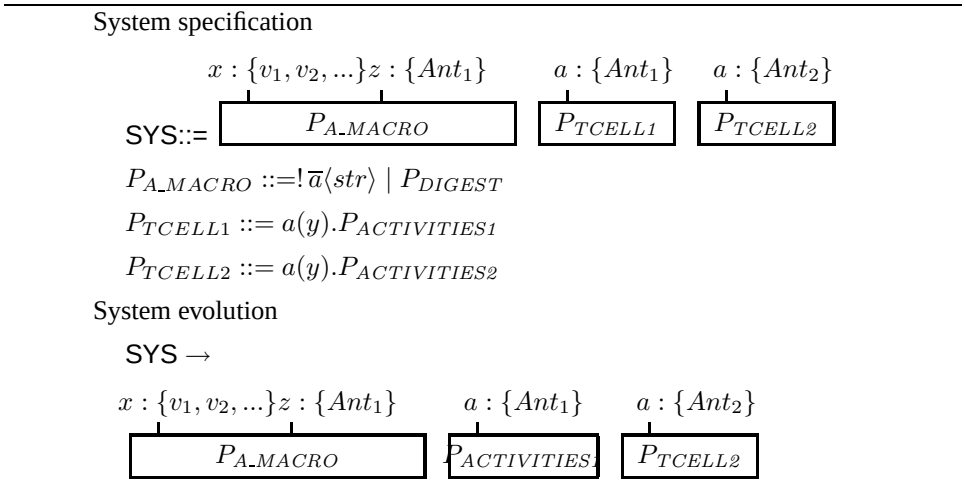


Figure 18: TCELL activation in Beta-binders

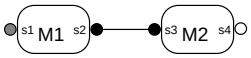
Finally, Beta-binders is equipped with a stochastic semantics [19] and the associated simulation environment [68] for the in-silico study of biochemical pathways.

6.7 κ-calculus

κ-calculus [16, 17] is a formal calculus of proteins interaction. It was conceived to represent complexation and decomplexation of proteins. The κ-calculus comes equipped with a very clear visual notation, and uses the concept of shared names to represents bonds.

6.7.1 Syntax and Semantics

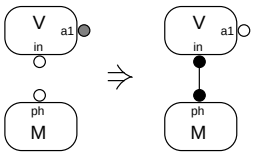
The units of κ-calculus are proteins, and operators are meant to represent creation and division of protein complexes. Proteins are drawn as boxes with sites on their boundaries. A site can be either visible, hidden or bound. For instance



represents two bounded molecules M1 and M2 on sites s2 and s3, respectively. Moreover, the site s1 of M1 is hidden and the site s4 is visible.

Besides the graphical representation, the κ-calculus provides a language in the style of process algebras. Expressions and boxes are given semantics by a set of basic reactions. Once the initial system has been specified and the basic reductions have been fixed, the behavior of the system is obtained by rewriting it after the reduction rules. This kind of reduction resembles pathway activation.

6.7.2 Example



(a)

System specification

$\mathbf{r}_1$: M(ph),V(in,$\overline{\text{a1}}$) $\rightarrow$(x)(M(ph^x),V(in^x,a1))

System evolution

M(ph),V(in,$\overline{\text{a1}}$) $\rightarrow \mathbf{r}_1$
(x)(M(ph^x),V(in^x,a1))

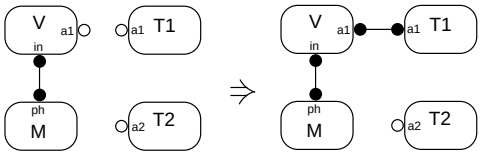
(b)

Figure 19: Phagocytosis-Digestion-Presentation in κ-calculus

Compartments. The calculus does not offer a natural support for the compartment layer. It is possible to represent the Phagocytosis-Digestion-Presentation example (see Figure 19(a)) as an activation pathway. The virus is rendered by the box V, which has a visible site in, used to enter a cell, and a hidden site a1, which represents the antigen. The macrophage is represented by the box M, which has a visible site ph, used to phagocytes a molecule.

Figure 19(b) shows the single reaction relevant to our running example. In this reaction rule, the superscript x in ph^x and in^x means that the sites in and ph are linked by the channel named x. This mechanism may resemble a possible handling of affinity between channels, although no quantitative measure is considered.

Biochemical interactions. Figure 20(a) shows the κ-calculus graphical representation of the activation of a lymphocyte T helper. After phagocytosis, the virus has a visible site a1, which represents its antigen: only the lymphocyte with the right site can bind it.



(a) Graphical representation

System specification

$\mathbf{r}_2$: M(ph^x),V(in^x,a1),T1(a1) $\rightarrow$ M(ph^x),V(in^x,a1^y),T1(a1^y)

$\mathbf{r}_3$: M(ph^x),V(in^x,a2),T2(a2) $\rightarrow$ M(ph^x),V(in^x,a2^y),T2(a2^y)

System evolution

(x)(M(ph^x),V(in^x,a1),T1(a1)) $\rightarrow \mathbf{r}_2$
(xy)(M(ph^x),V(in^x,a1^y),T1(a1^y))

(b) Language representation

Figure 20: TCELL activation in κ-calculus

The graphical notation does not clearly represent the selection of the right lymphocyte. This gap is filled by the formal model via the definition of the basic reactions. In particular, the system described in Figure 19(b) may be extended with the rules defined in Figure 20(b). By these rules, it is possible to infer the reduction shown in Figure 20(b).

6.7.3 Comments

The κ-calculus was designed to represent complexation and decomplexation of proteins, and therefore it does not allow to represent compartment primitives. We want to point out that the main goal of the authors of κ-calculus, i.e. to "provide a formalism that could be a suitable modeling language allowing direct descriptions of molecular events" [16], has been achieved in an effective way: the visual language is intuitive, and the formal one rather simple to use. Moreover, despite its simplicity, in [13], κ-calculus was shown to be expressive to translate Kohn Interaction Map [41], a diagrammatic formalism to represent networks containing multi-protein complexes, protein modifications, and enzymes.

7 Concluding remarks

The languages mentioned in this survey are quite different, and have been conceived for specifying entities at different levels of abstractions. As expected, none of them is 'the perfect language', which allows to model in an easy and correct way all kinds of biological operations. Each language, however, has some distinguishing features that make it particularly suitable for modeling certain kinds of systems or operations.

We can classify the various calculi depending on whether they are adaptations or extensions of calculi introduced to specify distributed systems, or rather they have been directly defined to model biological systems. For convenience, we refer to the languages of the first family as to bottom-up calculi, and to the others as top-down languages.

Biochemical stochastic π-calculus, BioAmbients, PEPA and CCS-R are bottom-up calculi. They are based on languages used to describe distributed systems. The main advantages of bottom-up languages are that they can rely on well-assessed mathematical basis and they are well-known in the community of distributed systems. The main drawback is that, since they were not meant to describe biological systems, they are often too abstract and not much intuitive.

Brane calculus, Beta-binders, and κ-calculus are top-down languages. Their authors made the opposite effort: they tried to identify the fundamental biological primitives and to represent them by the techniques and tools of concurrency theory. The advantages and drawbacks of top-down languages are opposite to those of bottom-up ones: these languages are usually more intuitive and more biologically correct but, since they are very recent, they lack theoretical works and few tools exist to allow them to be of practical use for validation/simulation purposes.

Some of the languages we have described permit an explicit representation of biological compartments: Brane calculus, BioAmbients and the Beta-binders extension with compartments. Therefore they can be more suitable to model phenomena at compartment level. Brane calculus is very interesting when the focus

is on membrane interactions and their evolution; BioAmbients also provides an intuitive representation of compartments, and the main difference between the two languages is the place where the computation occurs: on membranes in the former, and inside membranes in the latter. Therefore, BioAmbients seems to be more appropriate when the internal structure of compartments is relevant. The Beta-binders extension with compartments is somehow more similar to BioAmbients because the focus is primarily on interactions between internal objects: Beta-binders overcomes some of the known problems of BioAmbients, but it is not meant to model operations involving fusion of membranes, therefore it is not applicable in modeling systems in which such kind of operations is important.

References

[1] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for Identifying Boolean Networks and Related Biological Networks Based on Matrix Multiplication and Fingerprint Function. *Journal of Computational Biology*, 7(3):331–343, 2000.

[2] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for inferring qualitative models of biological networks. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 5, pages 293–304, Singapore, 2000. World Scientific Press.

[3] R. Alves and M. A. Savageau. Extending the method of mathematically controlled comparison to include numerical comparisons. *Bioinformatics*, 16(9):786–798, 2000.

[4] G. D. Bader, I. Donaldson, C. Wolting, B. F. Ouellette, T. Pawson, and C. W. Hogue. BIND-The Biomolecular Interaction Network Database. *Nucleic Acids Research*, 29(1):242–245, 2001.

[5] M. Calder, A. Duguid, S. Gilmore, and J. Hillston. Stronger computational modelling of signalling pathways using both continuous and discrete-state methods. In Corrado Priami, editor, *Proceedings of the Fourth International Conference on Computational Methods in Systems Biology (CMSB 2006)*, volume 4210 of *Lecture Notes in Computer Science*, pages 63–77, Trento, Italy, 2006. Springer.

[6] M. Calder, S. Gilmore, and J. 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, 2005.

[7] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. In *Transactions on Computational Systems Biology VII*, number 4230 in LNCS. Springer, 1–23 2006.

[8] M. Calder, S. Gilmore, J. Hillston, and V. Vyshemirsky. Formal methods for biochemical signalling pathways. *BCS*, 2006. To appear.

[9] L. Cardelli. Membrane interactions. In *BioConcur '03, Workshop on Concurrent Models in Molecular Biology*, 2003.

[10] L. Cardelli. Brane Calculi - Interactions of Biological Membranes. In *Proceedings of Workshop on Computational Methods in Systems Biology (CMSB'04)*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278, 2005.

[11] L. Cardelli and A. D. Gordon. Mobile Ambients. In *Proceedings of Conference on Foundations of Software Science and Computation Structures (FoSSaCS'98)*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer-Verlag, Berlin Germany, 1998.

[12] R. Chang. *Physical Chemistry for the Biosciences*. University Science, 2005.

[13] M. Chiaverini and V. Danos. A core modeling language for the working molecular biologist. In C. Priami, editor, *Proc. 1st Int. Workshop on Computational Methods in Systems Biology (CMSB 2003)*, volume 2602 of *Lecture Notes in Computer Science*. Springer, 2003.

[14] M. Curti, D. Chiarugi, P. Degano, and R. Marangoni. Vice: a virtual cell. In *Proceedings of Workshop on Computational Methods in Systems Biology (CMSB'04)*, volume 3082 of *Lecture Notes in Computer Science*, pages 207–220, 2004.

[15] V. Danos and J. Krivine. Formal Molecular Biology done in CCS-R. In *Proceedings of Workshop on Concurrent Models in Molecular Biology (Bio-CONCUR'03)*, Electronic Notes in Theoretical Computer Science, 2003.

[16] V. Danos and C. Laneve. Graphs for Core Molecular Biology. In *Proceedings of Workshop on Computational Methods in Systems Biology (CSMB'03)*, volume 2602 of *Lecture Notes in Computer Sciences*, pages 34–46. Springer, 2003.

[17] V. Danos and C. Laneve. Formal molecular biology. *TCS*, 325(1), 2004.

[18] V. Danos and S. Pradalier. Projective Brane Calculus. In *Proceedings of Workshop on Computational Methods in Systems Biology (CMSB'04)*, volume 3082 of *LNCS*, 2005.

[19] P. Degano, D. Prandi, C. Priami, and P. Quaglia. Beta-binders for biological quantitative experiments. In *Proceedings of the 4th International Workshop on Quantitative Aspects of Programming Languages (QAPL 2006)*, volume 164 of *Electronic Notes in Theoretical Computer Science*, pages 101–117. Elsevier, 2006.

[20] P. Dhaeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[21] D. Duchier and C. Kuttler. Biomolecular agents as multi-behavioural concurrent objects. In *Proceedings of the First International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005)*, volume 150 of *Electronic Notes in Theoretical Computer Science*, pages 31–49. Elsevier, 2005.

[22] S. Efroni, D. Harel, and I. R. Cohen. Towards rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation. *Genome Research*, 13(11):2485–97, 2003.

[23] A. Finney, H. Sauro, M. Hucka, and H. Bolouri. An XML-Based Model Description Language for Systems Biology Simulations. Technical report, California Institute of Technology, September 2000. Technical report.

[24] F. Fontana, L. Bianco, and L. Manca. A symbolic approach to the simulation of biochemical models: application to circadian rhythms. In *CSB Workshops*, pages 168–169, 2005.

[25] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[26] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in Lecture Notes in Computer Science, pages 353–368, Vienna, 1994. Springer-Verlag.

[27] P. J. E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. In *Proceedings of the National Academy of Sciences USA*, volume 95, pages 6750–6755, 1998.

[28] M.L. Guerriero, C. Priami, and A. Romanel. Beta-binders with Static Compartments. Technical Report TR-09-2006, The Microsoft Research - University of Trento Centre for Computational and Systems Biology, 2006.

[29] D. Harel and R. Marelly. Specifying and Executing Behavioral Requirements: The Play In/Play-Out Approach. *Software and System Modeling (SoSyM)*, 2:82–107, 2003.

[30] M. Heiner, I. Koch, and K. Voss. Analysis and Simulation of Steady States in Metabolic Pathways with Petri Nets. In K. Jensen, editor, *CPN'01-Third Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 15–34, 2001.

[31] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[32] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

[33] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

[34] R. Hofestädt and S. Thelen. Quantitative Modeling of Biochemical Networks. *In Silico Biology*, 1(1):39–53, 1998.

[35] PEPA home page. `http://www.dcs.ed.ac.uk/pepa/`.

[36] N. Kam, D. Harel, and I. R. Cohen. Modeling Biological Reactivity: Statecharts vs. Boolean Logic. In *Proc. of the Second International Conference on Systems Biology*, pages 301–310, Pasadena, CA, 2001.

[37] N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, E. J. A. Hubbard, and M. J. Stern. Formal Modeling of C. elegans Development: A Scenario-Based Approach. In C. Priami, editor, *Proc. 1st Int. Workshop on Computational Methods in Systems Biology (CMSB 2003)*, volume 2602 of *Lecture Notes in Computer Science*, pages 4–20. Springer, 2003.

[38] P. D. Karp, M. Riley, M. Saier, I. T. Paulsen, J. Collado-Vides, S.M. Paley, A. Pellegrini-Toole, C. Bonavides, and S. Gama-Castro. The EcoCyc Database. *Nucleic Acids Research*, 30(1):56–58, 2002.

[39] T. Kazic. Semiotes: a Semantics for Sharing. *Bioinformatics*, 16(12):1129–1144, 2000.

[40] H. Kitano, editor. *Foundations of Systems Biology*. The MIT Press, 2001.

[41] K.W. Kohn. Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems. *Molecular Biology of the Cell*, 10(8):2703–2734, 1999.

[42] M. C. Kohn and D. R. Lemieux. Identification of Regulatory Properties of Metabolic Networks by Graph Theoretical Modeling. *Journal of Theoretical Biology*, 150:3–25, 1991.

[43] D.E. Koshland. Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proceedings of the National Academy of Science of the United States of America*, 44(2):98–104, 1958.

[44] C. Kuttler and J. Niehren. Gene regulation in the pi calculus:simulating cooperativity at the lambda switch. In *Transactions on Computational Systems Biology VII*, number 4230 in LNCS. Springer, 2006.

[45] P. Lecca, C. Priami, P. Quaglia, B. Rossi, C. Laudanna, and G. Costantin. A Stochastic Process Algebra Approach to Simulation of Autoreactive Lymphocyte Recruitment. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 80(4), 2004.

[46] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri Net Representation of Gene Regulatory Network. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 5, pages 338–349, Singapore, 2000. World Scientific Press.

[47] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice hall, 1989.

[48] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[49] W. M. Mounts and M. N. Liebman. Qualitative Modeling of Normal Blood Coagulation and its Pathological States Using Stochastic Activity Networks. *International Journal of Biological Macromolecules*, 20:265–281, 1997.

[50] Hanne Riis Nielson, Flemming Nielson, and Henrik Pilegaard. Spatial Analysis of BioAmbients. In Roberto Giacobazzi, editor, *Static Analysis, 11th International Symposium, SAS 2004*, volume 3148 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.

[51] The P Systems We Page. Bibliography: `http://psystems.disco.unimib.it/biblio.html`.

[52] A. Păun and G. Păun. The power of communication: P systems with symport/antiport. *New Generation Comput.*, 20(3):295–306, 2002.

[53] G. Păun. P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.

[54] G. Păun. *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.

[55] M. Peleg, I. Yeh, and R. Altman. Modeling biological processes using Workflow and Petri Net models. *Bioinformatics*, 18:825–837, 2002.

[56] M.J. Perez-Jimenez and F.J. Romero-Campero. A study of the robustness of the egfr signalling cascade using continuos membrane systems. In *First Intern. Work-Conference on the Interplay between Natural and Artificial Computation, IWINAC 2005*, 2005.

[57] A. Phillips. SPiM home page: `http://www.doc.ic.ac.uk/~anp/spim`.

[58] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI-FN-19, Computer Science Department, Aarhus University, 1981.

[59] D. Prandi, C. Priami, and P. Quaglia. Shape spaces in formal interactions. *ComPlexUS*, 2(3-4):128–139, 2006.

[60] C. Priami. Stochastic pi-calculus. *The Computer Journal*, 38(7):578–589, 1995.

[61] C. Priami and P. Quaglia. Beta binders for biological interactions. In *Proc. of Computational Methods in Systems Biology*, volume 3082 of *LNCS*, pages 20–33, 2005.

[62] C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25–31, 2001.

[63] V. N. Reddy, M. L. Mavrouvouniotis, and M. N. Liebman. Qualitative analysis of Biochemical Reduction Systems. *Computers in Biology and Medicine*, 26(1):9–24, 1996.

[64] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Y. Shapiro. BioAmbients: an Abstraction for Biological Compartments. *Theoretical Computer Science*, 325(1):141–167, 2004.

[65] A. Regev and E. Shapiro. Cells as Computation. *Nature*, 419:343, 2002.

[66] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Proceedings of Pacific Symposium on Biocomputing (PSB'01)*, volume 6, pages 459–470, 2001.

[67] W. Reisig. *Petri Nets : an Introduction*. Springer-Verlag, 1985.

[68] A. Romanel, L. Dematté, and C. Priami. The betasim system. Technical Report TR-3-2007, The Microsoft Research - University of Trento Centre for Computational and Systems Biology, February 2007.

[69] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge Universtity Press, 2001.

[70] H. M. Sauro, M. Hucka, A. Finney, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano. Next Generation Simulation Tools: The Systems Biology Workbench and BioSPICE Integration. *OMICS: A Journal of Integrative Biology*, 7(4):355–372, 2003.

[71] S. Schuster, D. A. Fell, and T. Dandekar. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nature Biotechnology*, 18(3):326–332, 2000.

[72] W. Silverman. BioSpi home page: `http://www.wisdom.weizmann.ac.il/~biopsi/`.

[73] J. Stenesh. *Biochemistry*. Springer, 1998.

[74] Z. Szallasi. Genetic network analysis in light of massively parallel biological data. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 4, pages 5–16, Singapore, 1999. World Scientific Press.

[75] J. van Helden, A. Naim, R. Mancuso, M. Eldridge, L. Wernisch, D. Gilbert D, and S. J. Wodak. Representing and analysing molecular and cellular function using the computer. *Biological Chemistry*, 381(9–10):921–935, 2000.

[76] E. Wingender, X. Chen, E. Fricke, R. Geffers, R. Hehl, I. Liebich, M. Krull M, V. Matys, H. Michael, R. Ohnhauser, M. Pruss, F. Schacherer, S. Thiele, and S. Urbach. The TRANSFAC system on gene expression regulation. *Nucleic Acids Research*, 29(1):281–283, 2001.

[77] T-M. Yi, Y. Huang, M. I. Simon, and J. Doyle. Robust perfect adaptation in bacterial chemotaxis through integral feedback control. *Proceedings of the National Academy of Sciences USA*, 97(9):4649–4653, 2000.