Computational Thinking in Biology

Corrado Priami

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

Abstract. The paper presents a new approach based on process calculi to systems modeling suitable for biological systems. The main characteristic of process calculi is a linguistic description level to define incrementally and compositionally executable models. The formalism is suitable to be exploited on the same system at different levels of abstractions connected through well defined formal rules. The abstraction principle that represents biological entities as interacting computational units is the basis of the computational thinking that can help biology to unravel the functions of the cell machinery. We discuss then the perspectives that process calculi can open to life sciences and the impact that this can in turn produce on computer science.

1 Introduction

Systems level understanding of phenomena has recently become an issue in biology. The complexity of molecular interactions (gene regulatory networks, signaling pathways, metabolic networks, etc.) makes impossible to handle the emergent behavior of systems simply by putting together the behavior of their components. Interaction is a key point in the study of emergence and complexity in any field and hence in biology as well where the molecular machinery inside a cell determines the behavior of complex organisms.

Besides interaction, the other key issue to develop computer-based tools for systems biology is incremental construction of models. We need to add something to a model once new knowledge is available without altering what we already did. This is an essential feature for modeling formalisms being applicable to real size problems (not only in the biological applicative domain). Many approaches have been investigated in the literature to model and simulate biological systems (e.g., ODE or stochastic differential equations, Petri nets, boolean networks, agent-based systems), but most of them suffer limitations with respect to the above issues.

In recent times, programming languages based approaches have been proposed to generate executable models at a linguistic level. We think that they are a suitable tool to address interaction, incremental building of models and complexity of emergent behavior. As usual in computer science, the definition of a high level linguistic formalism that then must be mapped onto lower level representations to be executed may loose efficiency but gain a lot in expressive

C. Priami (Ed.): Trans. on Comput. Syst. Biol. VIII, LNBI 4780, pp. 63–76, 2007.

power and usability. Being the systems in hand huge, we need such formalisms to minimize the error prone activity of specifying behavior.

The main idea is that computer networks, and Internet in particular, are the artificial systems most similar to biological systems. Languages developed in the last twenty years to study and predict quantitatively the dynamic evolution of these networks could be of help in modeling and analysing biological systems. Recent results show that process calculi (very simple modeling languages including the basic feature to model interaction of components) have been successfully adopted to develop simulators [22,24,19] that can faithfully represent biological behavior.

The correspondence between the way in which computer scientists attacked the complexity of artificial systems and the way in which such complexity is emerging in biology when interpreting living systems as information processing unit [13] is very strict. Therefore computational thinking [26] is a tool that can extremely help enhance our understanding of living systems dynamics. Computational thinking expresses the attitude of looking at the same problem at different levels of abstraction and to model it through executable formalisms that can provide insights on temporal evolution of the problem in hand. Therefore the basic feature of computational thinking is abstraction of reality in such a way that the neglected details in the model make it executable by a machine. Of course, different executable abstractions of the same problem exist and the choice is driven by the properties to be investigated. Indeed, science history shows us that a single model for the whole reality does not exist: our modeling activity must be driven by the properties of the phenomenon under investigation that we want to look at.

Process calculi have been originally introduced [15,12] as specification languages for distributed software systems. The specification can be refined towards an actual implementation within the same formalism. Any refinement step is validated by formal proofs of correctness. This approach is a good example of a framework that imposes the application of the computational thinking and therefore we work on it to obtain a similar framework for biological systems.

We here briefly and intuitively introduce process calculi (in particular we concentrate on the β-language) to show on an example how they can be used to model biological systems. We then investigate the potential of the approach in a perspective vision. We first discuss how life scientists can improve their performance by relying on software and conceptual tools that allow them to mimick the standard activities they perform in wet labs. There are however two main advantages to work in silico: speed and cost. Actually experiments last few minutes instead of hours or days and the cost is extremely reduced. Once the scientist think of having something concrete in silico can move towards the wet lab and test *in practice* the hypotheses. Essentially there is an iterative loop between in silico production of hypothesis and wet testing of them.

The longer term perspective of the approach is related to enhancement of computer science. The knowledge we gain from developing linguistic mechanisms to describe and execute the dynamics of complex biological systems could lead to

the definition of a new generation of programming languages and new programming paradigms that can enhance the software production tools now available.

2 Abstracting Biological Systems

A biological system can be studied at the molecular, cellular, tissue, organ and population levels. The dynamic steps that drive the state change of the above systems can be always reduced to interactions of some kind (e.g., protein-protein, cell-cell, member-member, etc.). Actually, one of the most used concepts in systems level understanding of biology is the one of network (see Fig. 1): a structure made up of nodes (the components of the considered system) linked by arcs (the interactions between components). The interaction can enhance or inhibit some activities, thus we usually observe activation arcs (ending with an arrow) and repressor arcs (ending with a line orthogonal to the arc).

A reasonable formalisms able to model the dynamics of biological systems must then be able to represent components and their interactions. Since networks are essentially graphs whose arcs represent some kind of chemical/physical reaction between components, a mathematical tool to represent the system could be

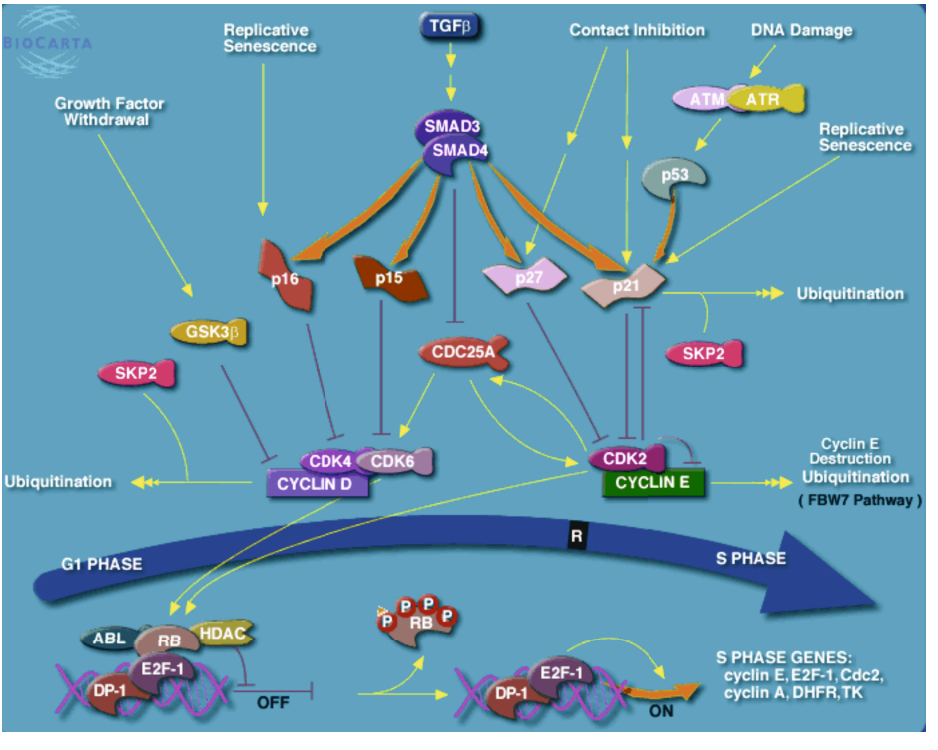

Fig. 1. A cell cycle pathway from the Biocarta database

a stoichiometric matrix [17]. Rows and columns are components, the entries store some quantitative measure characterizing the interaction of the corresponding components. Although the matrix contains all the relevant information, its size is extremely large for practical useful systems.

Computer science, executable, formalisms that resemble the stoichiometric matrix are the ones based on multi-set rewriting (e.g., P systems [18,5] or variant of Petri Nets, e.g., [10]). The main limitation of the stoichiometric matrix as well as of these computer science formalisms is that they need to represent in the description of the model all the possible configuration in which the network can pass and all the possible interactions explicitly. No emergent behavior can arise from the execution or the analysis of the model if it has not been explicitly modeled.

Since biological networks exhibit combinatorial features, i.e., the number of possible configurations of the network grows exponentially with respect to the number of components, the explicit representation of all the configurations is a difficult, time-consuming and error-prone task. Summing up the approaches mentioned above are mainly used and suitable to systematize the knowledge on dynamics of systems and to make it unambiguous. The main applications are then storing and comparison of models.

Another important feature of models is however the predicting power that can help designing new experiments to discover new knowledge. To stress the heuristic value of a model, we think that it is better to have an intentional description of the system whose dynamics as well as the set on intermediate step and configuration is determined by the execution of the model. In other words we are looking for a formalism that allows us to represent the components of the network and a set of general rules that provide information on how components may interact. Then, "in silico" experiments (i.e., execution of the model) provide us with possible scenarios of interaction (i.e., with possible network configurations that the system can pass through). As an example, we list the proteins in a system and their slectivity/affinity or binding/unbinding parameters and we let the execution of the model to predict which are the actual interactions and/or complexation/decomplexation of the proteins.

Since the most similar artificial systems to biological networks are networks of interacting computer or of interacting software programs, we re-use in the life science domain description languages like CCS [15], CSP [12] and then π-calculus [16] that have been invented to model and study properties of distributed and mobile software systems. Actually, the first process calculus applied to biological problem has been the stochastic π-calculus [20] (also supported by automatic available tools for simulation [22,19]) that opened a field of research that is more and more populated of calculi adopted for biological modeling. To mention a few of them we recall BioAmbients [23], CCS-R [6], PEPA [1].

The abstraction principle underlying this approach [24] is that any biological component is represented by a program. The interaction between biological entities is then represented as an exchange of information between programs. Since in a distributed software system in which many programs run simultaneously the

information exchanged between them makes the future behavior of the system change, the interaction becomes the basic step of state change exactly as it happens in biological networks. We slightly refine this principle by equipping communication links with types that define the sensitivity of the biological component they belong to (see Fig. 2).

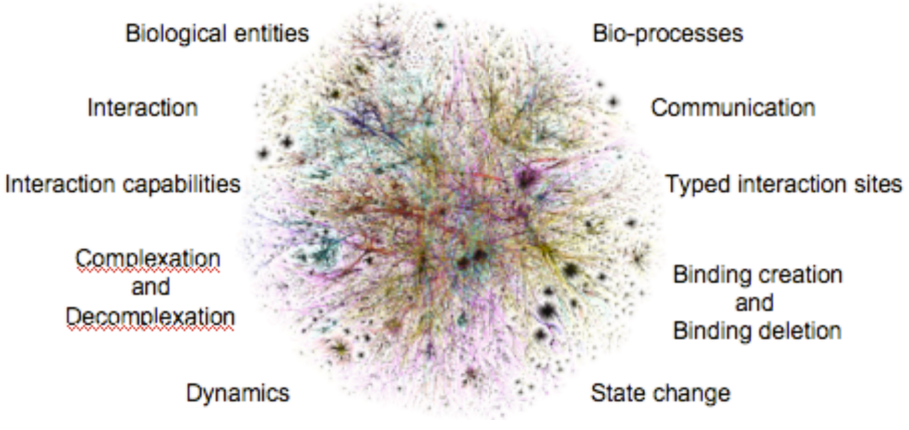

Fig. 2. Abstraction principles to model biological systems - figure prepared by A. Romanel and L. Dematté

The linguistic level of the calculi allows us to have representations that grow linearly with the number of components of the net, while only the execution of the model faces the exponential number of configurations that the network can reach. The selection of the next state from the current configuration is driven by the quantities that express the affinity of interaction through the implementation of a stochastic run-time engine based on the Gillespie's algorithm [9]. Therefore, the execution of a model provide us with the variation over time of the concentration of the components that form the system, i.e., the execution of a model corresponds to the stochastic simulation of the system.

The other property that makes process calculi good candidates to overcome the actual limitations of modeling approaches is the so-called *compositionality* or capability to build models incrementally. Most of the current approaches allow to increase the size of a model by adding new knowledge to it and performing a partial rewriting of the current available model (think of ODE when adding new variables, or rewriting systems when adding new interacting elements – one needs to take care of the new comers variables or components in the existing equations or rewriting rules). This is clearly an obstacle to the scalability of the approach towards genome-size and organism applications. Since the combinatoric features of networks are handled in process calculi at execution time, the description of new elements can be implemented just by adding to the pool of already existing programs the new one describing the new biological entity and

by adding to the affinity rules the values describing the interacting capabilities of the new entity. Then the run-time engine will take care of the new program according to its interaction capabilities. This is common practice in software development where *interfaces* of programs are defined to let them interact without the need of re-coding their internal structure. As an example see Fig. 3 where the interface of the β-workbench introduced in the next section is reported. The rectangles represent the bio-processes, i.e. the biological elements, and the arcs their interaction capabilities. Adding new elements to the system is just a matter of introducing a new box with the corresponding interaction arcs. The code generator will then take care of the new possible behavior of the overall system.

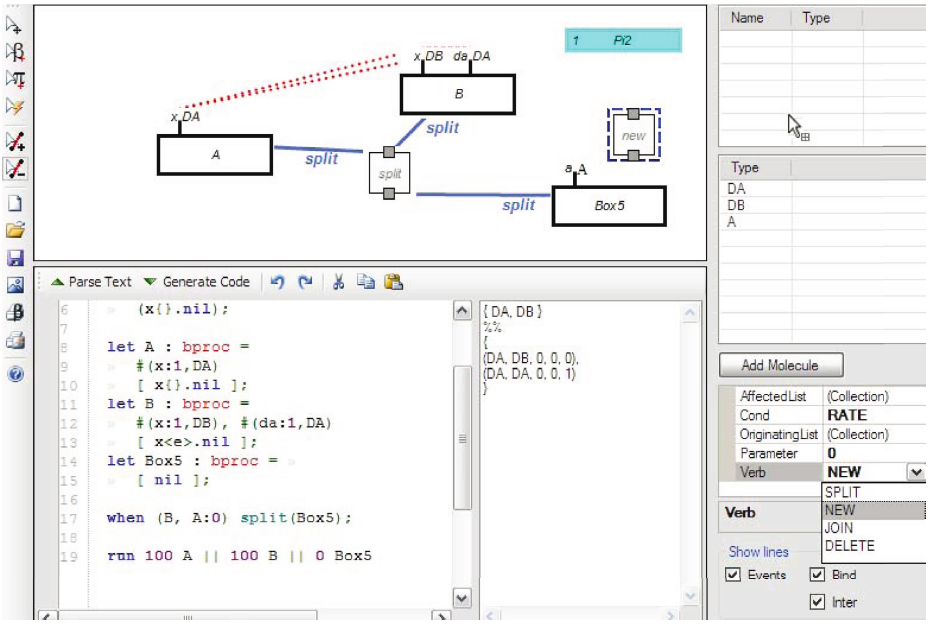

Fig. 3. The modeling interface (β-designer) of the β-workbench

Most of the process calculi mentioned above have been applied to biological problems although they have been defined for computer science modeling. The result was a feasibility study of the potentials of these calculi to model and simulate biological systems. At the same time some limitations emerged at the modeling level. Hence, many researchers defined new variants and extensions of the existing calculi to directly address biological features. Among them we mention the Brane calculi [2] designed to model membrane interactions, the κ-calculus [7] designed to model complexation and decomplexation, SPICO [14] designed to add object-oriented features to stochastic π-calculus, β-binders [21] designed to exploit the notion of interface of biological entities and introduce

a sensitivity/affinity based interaction. In fact all the calculi defined before β-binders have been implicitly assuming that two structures can interact only if they are exactly complementary (a perfect key-lock mechanism of interaction). Besides stochastic π-calculus, β binders is the only process calculus equipped with a stochastic simulation engine [25]. Since β binders is the last process calculus defined and enjoys properties that the previous proposals do not have, in the next section we introduce the basics of process calculi relying on β binders.

3 The β Workbench

We present in this section a frame to model and simulate biological systems relying on process calculi. The β workbench (hereafter βWB) is based on *beta binders* and it is made up of three components: the β language with its stochastic abstract machine, the β designer (see Fig. 3) to help modeling activities and the β plotter (see Fig. 4) to inspect the results of the simulations. Since the purpose of the paper is to illustrate the perspectives of applying process calculi in modeling biological systems, we concentrate here only on the β language and we refer the reader to [25] for more details on the βWB.

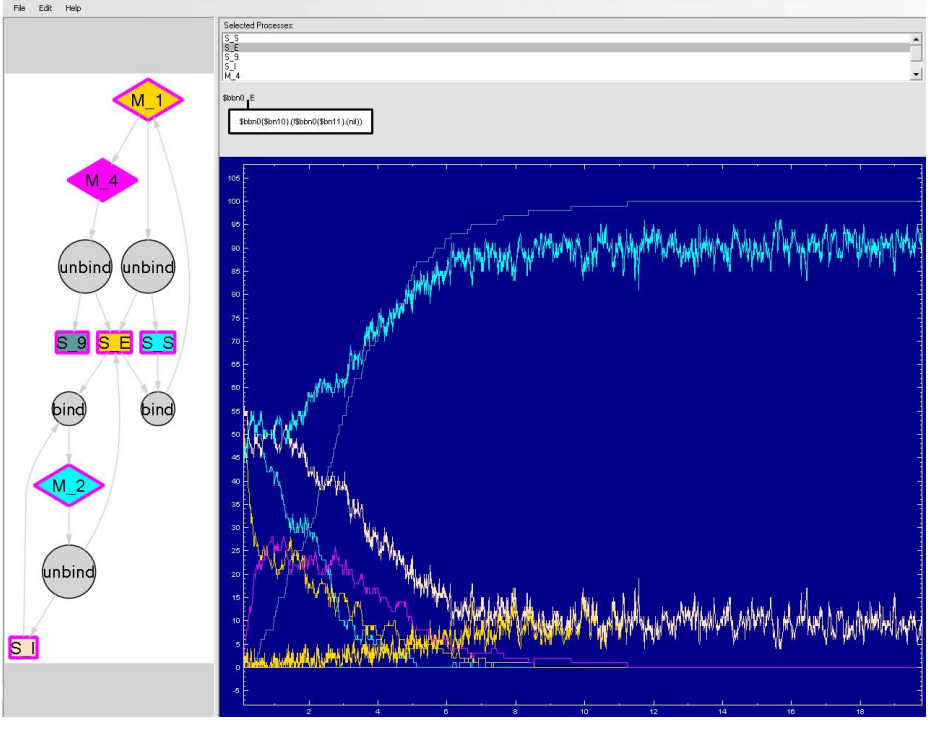

Fig. 4. The output interface (β-plotter) of the β-workbench

Every biological entity is represented as a box (e.g. a protein, a cell, a member of a population) with a set of interfaces through which the entity interacts with other entities (e.g. a set of receptors). Any box contains a small program describing the activities that the box can perform in response to stimuli on its interfaces (e.g., conformational changes, activation or deactivation of other interfaces). A whole biological system is composed of a set of boxes (proteins, cells, populations) each of them equipped with a unique name (species) and an arity determining the available numbers or concentration of biological entities abstracted by the box in the system. Note that a box can pass through different states according to the configurations of its interfaces and its internal program without the need of changing its unique name (species) in the model. For instance, Fig. 6(a) represents graphically a box where b_1 is the unique name of the box and 17 is the number of the copies of the box available in the system (the concentration of the corresponding component); P inside the box is its internal program (how the box works in response to stimuli on interfaces); and rectangles, triangles and circles on the border of the box are the interfaces. The color of an interface denotes its type, while x_i denotes its name and s_i is the stochastic parameter describing a continuous time exponential distributions. The type of an interface is used to determine the affinity of interaction between boxes. In fact the run time engine is based on a function α between types that provides a quantity expressing the propensity of interaction. Finally the parameter s_i is a stochastic information needed to drive the simulation of the system. Interfaces can be active (rectangles), or hidden (circles) or complexed (triangles). An interface can become hidden for instance when forming a complex that following a conformational change makes a binding side hidden by its three dimensional structure. An interface can become complexed when two boxes glue together by that interface.

The actions that a box can perform are either internal to a box (monomolecular operations), or they affect two boxes (bimolecular operations) or they are driven by global conditions on the system (events). Monomolecular operations manipulates interfaces through hiding and unhiding, creating new interfaces or changing the types (i.e., interaction capabilities) of existing interfaces. Furthermore they may allow interaction between different part of the same box or may decide to kill the box. These operations are graphically represented in Fig. 6(b). Bimolecular operations involve two boxes and allow them to interact. Interaction is implemented via exchange of information between the two boxes or via complexation or decomplexation of interfaces. The speed and probability of interaction is driven by the affinity function over the types of the interfaces selected for the current interaction. These operations are graphically depicted in Fig. 6(c).

Events are global rules of the execution environment triggered by conditions such as comparison with threshold on concentrations of boxes or existence of a given species in the current state. We can also check whether a given step or simulation time is reached. The actions associated with conditions can be deletion or creation of boxes, joining of two boxes into one or the splitting of

a a box into two. The notion of event inherited from event-based programming allow us to control the context in which the phenomenon under investigation is happening. Furthermore, events easily allows us to perturbate the system modeled and to analyse the new behavior. Perturbation of models is an essential feature if we want to develop an in-silico lab. In fact most of the experiments are the observation of the reactions of a system to some pre-defined perturbations.

The selection of the actions to be performed is driven by the Gillespie's algorithm in connection with the flow of control of boxes coded in their internal programs.

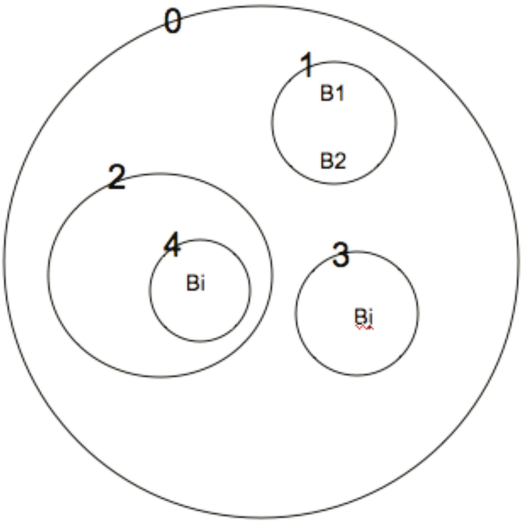

Fig. 5. Hierarchical structures. Compartments are uniquely identified by sequences of natural numbers. The largest one is identified by 0 and it contains three comparments identified by 02, 01 and 03. Furthermore, the compartment 02 contains the compartment 024.

Due to the large number of membranes existing in biological systems, an important feature to model real case studies is the ability of expressing compartments. The need of adding spatial information to models is also in computer science when modeling distributed software systems. For instance, it is important to partition a system into administrative domains to associate privileges with them and to check security polices. Two approaches have been adopted in process calculi: explicit representation of domains into hierarchical structures as in the ambient-family calculi [3] or implicit representation of domain through the notion of location of a program [4]. We focus here on the implicit approach because it allows us to maintain a flat structure of boxes still describing compartments [11]. Flat structures are more efficient to implement simulators. We only need to associate every box with a location. Then all the boxes in the same

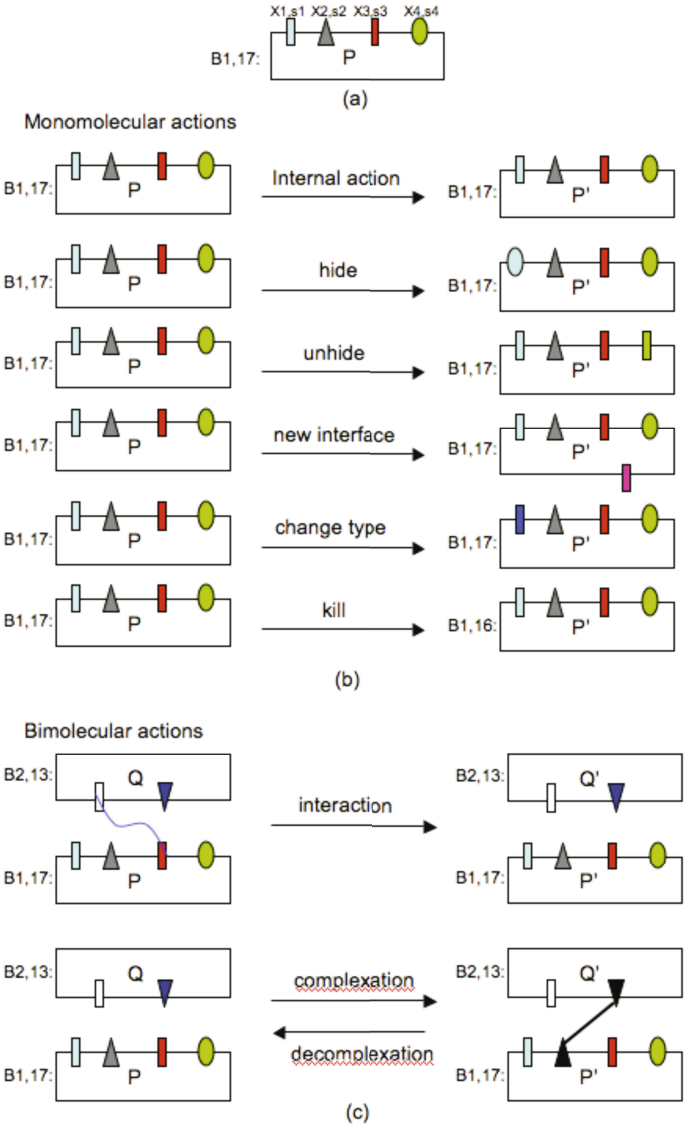

Fig. 6. Boxes, monomolecular and bimolecular operations. (a) Representation of a biological entity named B1 of 17 copies are available. The internal program P drives the behavior of the entity and the interfaces specify the interaction capabilities. Rectangle interfaces are active sites, triangle interfaces are complexed interfaces and oval represents hidden sites. The color of the sites denotes their type or interaction affinity. (b) The set of available monomolecular actions. The first one is just an internal updating that affect the P program. The other actions are needed to manipulate interfaces. Note that the kill action affects the number of available copies of the entity. (c) Bimolecular actions model exchange of information (the first action) or creation and destruction of complexes (the last two actions).

location are interpreted as being in the same compartment. Hierarchy of compartments is then obtained by imposing a containment relation over locations. A practical solutions is to use a tree-structure imposed by indexes similar to the ones of Dewey. For instance, consider the system represented in Fig. 5. The box b_i is represented by $024, b_i$ or the box b_j by $03, b_j$, where locations are paths in the hierarchy. Furthermore, the the fact that b_1 and b_2 are in the same compartment is rendered by using the same location 01 for both b_1 and b_2. Summing up, process calculi are suitable to represent compartments as well.

4 Potentials

The perspectives that emerge from the biological modeling based on process calculi are both in the life science domain and in the computer science domain. As far as life science is concerned, the predictive power of the approach could be exploited to inform wet biologists and help them planning focused experiments. An application is the description of a systems and then the study of its behavior under predefined perturbations. A perturbation in our approach is simply the adding of a new program to the set of the ones defining the system and inspecting the results of the new execution of the model. Clearly this capability is of interest both in better understanding the aetiology of diseases and in the definition of new drugs that exactly act on the causes of diseases. Note that the perturbation of a system can be caused by environmental factors in the case of diseases or by drugs when we want to inspect whether a molecule can stop or even prevent a disease. Furthermore, we can also think of perturbation caused by molecules contained in foods to test their toxicity or their actions on some diseases (for instance nutrigenomics could benefit from this approach).

Biology is mainly driven by quantities that emerge from real experiments. Hence we need to connect wet experiments and models in such a way that the available knowledge is used to inform simulators and hence constrain their levels of freedom. The iteration of these action should lead to an exact model of the phenomenon considered. To address this issue the bayesian inference of rate parameters from measures of concentrations at different times seems to be a promising approach.

Since process calculi are linguistic constructs to describe the dynamics of systems, they could heavily influence the definition of exchange model formalisms like SBML [8]. In fact, one of the main limitation of xml-based approaches is the inherent ambiguity of the descriptions that makes it hard to develop automatic translators into formal tools for analysing and simulating systems. Process calculi based notation could integrate SBML-like description languages to limit ambiguity and hence improve the already valuable usefulness of SBML. Of course, adding a degree of dynamics to static xml-based descriptions could be of interest also for computer science applications.

The real challenge we face in modeling biological systems is the definition of artificial systems that resemble the real biological behavior at a level of details that allows us to use them in place of animal models of diseases. This goal is still

far from being available, but we need to work in that direction by addressing real biology and by letting us driven by biology if we want our community to grow and to be beneficial for life scientists.

The impact of the proposed approach on computer science can be further explained in terms of medium and long term goal. The medium term goal is the development of a set of quantitative tools for the modeling and analysis of complex artificial systems like sensor networks or hybrid large networks. Performance prediction, load balancing and pricing are all issues that deserve quantitative frameworks that share the incremental properties we showed for the βWB.

Another huge applicative domain that could benefit of the outcome of biological modeling is the one of web-services. In fact, orchestration and contract negotiation is an interaction which is inherently not key-lock. Hence understanding how biological interaction is driven and modeled could provide breakthrough insights on the definition and implementation of new and better web-services.

Long term goals concern the definition of new computational models and new programming languages that allow us to build software systems that are more robust, fault tolerant, secure than the current ones. All the mentioned properties are typical of biological systems, but are lacking in the actual artifacts. If we can enhance our understanding of biological functioning, we could get inspiration for a new generation of software developing environments.

5 Conclusions

The new field of computational and systems biology can have a large impact on the future of science and society. The engine driving the new coming discipline is its inherent interdisciplinarity at the convergence of computer science and life sciences. To continue fueling the progress of the field we must ensure a peer-to-peer collaboration between the scientists of the two disciplines. In fact if one discipline is considered a service for the other the cross-fertilization will stop soon. We must create common expectations and really joint projects in which both computer science and biology can enhance their state-of-the-art.

We must ensure a critical mass of people working in the field and a common language to exchange ideas. This is a major problem in current collaborations due to the lack of curricula that form people to work in this intersection area. We must invest time and resources in creating interdisciplinary curricula (together with new ways of recruiting people considering interdisciplinarity an added value) to form the new researchers of tomorrow.

Summing up, although a lot has still to be done, we started a new way of making science that can lead in the next years to unravel the machinery of cell behavior that in turn can lead to the creation of artificial systems enjoying the properties of living systems. Computational thinking is different way of approaching a problem by producing descriptions that are inherently executable (differently, e.g., from a set of equation). Furthermore the same specification can be examined at different level of abstractions simply by building a virtual hierarchy of interpretations. This a common practice in computer science where

artificial systems are usually defined and described in layers depending on the growing abstraction from the physical architecture.

Acknowledgements. I would like to thank the whole research and admin team of CoSBi for the fruitful discussions and the beautiful environment in which it is my pleasure to work. I would like to thank Stephen Emmott and Luca Cardelli as well.

References

1. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. In: Dumke, R.R., Abran, A. (eds.) IWSM 2000. LNCS, vol. 4230, Springer, Heidelberg (2006)
2. Danos, V., Schachter, V. (eds.): CMSB 2004. LNCS (LNBI), vol. 3082. Springer, Heidelberg (2005)
3. Cardelli, L., Gordon, A.D.: Mobile ambients. In: Nivat, M. (ed.) ETAPS 1998 and FOSSACS 1998. LNCS, vol. 1378, Springer, Heidelberg (1998)
4. Castellani, I.: Process algebras with localities. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 945–1046 (2001)
5. Ciobanu, G., Rozenberg, G. (eds.): Modelling in Molecular Biology. Springer, Heidelberg (2004)
6. Danos, V., Krivine, J.: 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)
7. Danos, V., Laneve, C.: Formal molecular biology. TCS 325(1) (2004)
8. Finney, A., Sauro, H., Hucka, M., Bolouri, H.: An xml-based model description language for systems biology simulations. Technical report, California Institute of Technology, Technical report (September 2000)
9. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81(25), 2340–2361 (1977)
10. Goss, P.J.E., Peccoud, J.: Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. In: Proceedings of the National Academy of Sciences, USA, vol. 12, pp. 6750–6754 (1998)
11. Guerriero, M.L., Priami, C., Romanel, A.: Beta-binders with static compartments. In: Algebraic Biology, 2007. to appear. Also TR-09-2006. The Microsoft Research - University of Trento Centre for Computational and Systems Biology (2007)
12. Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM 21(8), 666–677 (1978)
13. Hood, L., Galas, D.: The digital code of DNA. Nature 421, 444–448 (2003)
14. Kuttler, C., Niehren, J.: Gene regulation in the pi-calculus: simulating cooperativity at the lambda switch. In: Priami, C., Ingólfsdóttir, A., Mishra, B., Nielson, H.R. (eds.) Transactions on Computational Systems Biology VII. LNCS (LNBI), vol. 4230, Springer, Heidelberg (2006)
15. Milner, R.: Communication and Concurrency. International Series in Computer Science. Prentice-Hall, Englewood Cliffs (1989)
16. Milner, R.: Communicating and mobile systems: the π-calculus. Cambridge Universtity Press, Cambridge (1999)
17. Palsson, B.O.: Systems Biology. Properties of reconstructed networks. Cambridge Universtity Press, Cambridge (2006)

18. Păun, G. (ed.): Membrane Computing. An Introduction. Springer, Heidelberg (2002)
19. Phillips, A., Cardelli, L.: A Correct Abstract Machine for the Stochastic Pi-calculus. In: Priami, C., Ingólfsdóttir, A., Mishra, B., Nielson, H.R. (eds.) Transactions on Computational Systems Biology VII. LNCS (LNBI), vol. 4230, Springer, Heidelberg (2006)
20. Priami, C.: Stochastic π-calculus. The Computer Journal 38(6), 578–589 (1995)
21. Priami, C., Quaglia, P.: Beta Binders for Biological Interactions. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 20–33. Springer, Heidelberg (2005)
22. Priami, C., Regev, A., Silverman, W., Shapiro, E.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters 80(1), 25–31 (2001)
23. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: BioAmbients: An Abstraction for Biological Compartments. TCS 325(1) (2004)
24. Regev, A., Shapiro, E.: Cells as computation. Nature 419(6905), 343 (2002)
25. Romanel, A., Dematté, L., Priami, C.: The Beta Workbench. Technical Report TR-3-2007, The Microsoft Research - University of Trento Centre for Computational and Systems Biology (February 2007)
26. Wing, J.: Computational thinking. Communications of the ACM 49(3), 33–35 (2006)