



Visualization in Process Algebra Models of Biological Systems

LUCA CARDELLI
Microsoft Research

CORRADO PRIAMI
Microsoft Research -
University of Trento
Centre for Computational
and Systems Biology and
University of Trento

IN A RECENT PAPER, NOBEL LAUREATE PAUL NURSE calls for a better understanding of living organisms through “both the development of the appropriate languages to describe information processing in biological systems and the generation of more effective methods to translate biochemical descriptions into the functioning of the logic circuits that underpin biological phenomena.” [1]

The language that Nurse wishes to see is a formal language that can be automatically translated into machine executable code and that enables simulation and analysis techniques for proving properties of biological systems. Although there are many approaches to the formal modeling of living systems, only a few provide executable descriptions that highlight the mechanistic steps that make a system move from one state to another [2]. Almost all the techniques related to mathematical modeling abstract from these individual steps to produce global behavior, usually averaged over time.

Computer science provides the key elements to describe mechanistic steps: algorithms and programming languages [3]. Following the metaphor of molecules as processes introduced in [4], process calculi have been identified as a promising tool to model biological systems that are inherently complex, concurrent, and driven by the interactions of their subsystems.

Causality is a key difference between language-based modeling approaches and other techniques. In fact, causality in concurrent languages is strictly related to the notion of concurrency or independence of events, which makes causality substantially different from temporal ordering. An activity A causes an activity B if A is a necessary condition for B to happen and A influences the activity of B—i.e., there is a flow of information from A to B. The second part of the condition defining causality makes clear the distinction between precedence (related only to temporal ordering) and causality (a subset of the temporal ordering in which the flow of information is also considered) [5]. As a consequence, the list of the reactions performed by a system does not provide causal information but only temporal information. It is therefore mandatory to devise new modeling and analysis tools to address causality.

Causality is a key issue in the analysis of complex interacting systems because it helps in dissecting independent components and simplifying models while also allowing us to clearly identify cross-talks between different signaling cascades. Once the experimentalist observes an interesting event in a simulation, it is possible to compact the previous history of the system, exposing only the preceding events that caused the interesting one. This can give precise hints about the causes of a disease, the interaction of a drug with a living system (identifying its efficacy and its side effects), and the regulatory mechanisms of oscillating behaviors.

Causality is a relationship between events, and as such it is most naturally studied within discrete models, which are in turn described via algorithmic modeling languages. Although many modeling languages have been defined in computer science to model concurrent systems, many challenges remain to building algorithmic models for the system-level understanding of biological processes. These challenges include the relationship between low-level local interactions and emergent high-level global behavior; the incomplete knowledge of the systems under investigation; the multi-level and multi-scale representations in time, space, and size; and the causal relations between interactions and the context awareness of the inner components. Therefore, the modeling formalisms that are candidates to propel algorithmic systems biology should be complementary to and interoperable with mathematical modeling. They should address parallelism and complexity, be algorithmic and quantitative, express causality, and be interaction driven, composable, scalable, and modular.

LANGUAGE VISUALIZATION

A fundamental issue in the adoption of formal languages in biology is their

usability. A modeling language must be understandable by biologists so they can relate it to their own informal models and to experiments.

One attempt by biologists to connect formal languages and informal descriptions of systems involved the use of a constrained natural language organized in the form of tables that collect all the information related to the structure and dynamic of a system. This narrative representation is informative and structured enough to be compiled into formal description that is amenable to simulation and analysis [6, 7]. Although the narrative modeling style is not yet visual, it is certainly more readable and corresponds better to the intuition of biologists than a formal (programming) language.

The best way to make a language understandable to scientists while also helping to manage complexity is to visualize the language. This is harder than visualizing data or visualizing the results of simulations because a language implicitly describes the full kinetics of a system, including the dynamic relationships between events. Therefore, language visualization must be dynamic, and possibly reactive [8], which means that a scientist should be able to detect and insert events in a running simulation by direct intervention. This requires a one-to-one correspondence between the internal execution of a formal language and its visualization so that the kinetics of the language can be fully reflected in the kinetics of the visualization and vice versa.

This ability to fully match the kinetics of a general (Turing-complete) modeling language to visual representations has been demonstrated, for example, for pi-calculus [9], but many practical challenges remain to adapting such general methods to specific visualization requirements. (See Figure 1 on the next page.) One such requirement, for example, is the visualization and tracking of molecular complexes; to this end, the BlenX language [10] and its support tools permit explicit representation of complexes of biological elements and examination of their evolution in time [11]. (See Figure 2 on page 103.) The graphical representation of complexes is also useful in studying morphogenesis processes to unravel the mechanistic steps of pattern formation. (See Figure 3 on page 104.)

ANALYSIS

Model construction is one step in the scientific cycle, and appropriate modeling languages (along with their execution and visualization capabilities) are important, particularly for modeling complex systems. Ultimately, however, one will want to analyze the model using a large number of techniques. Some of these techniques may be centered on the underlying mathematical framework, such as the analysis of

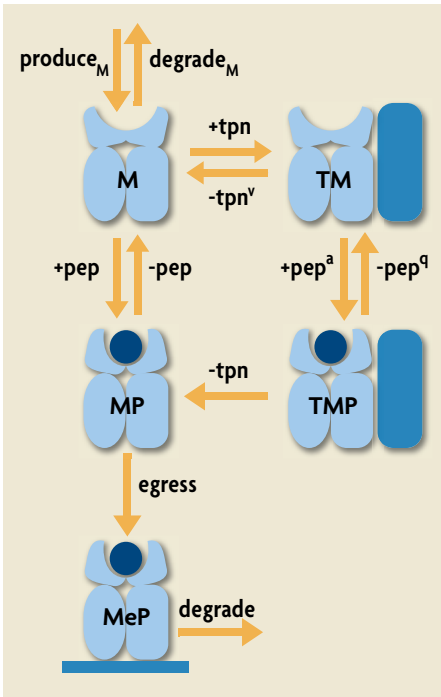


FIGURE 1. This diagram can be placed in 1:1 correspondence with formal stochastic pi-calculus models [9, 12, 13] so that one can edit either the diagrams or the models. The nodes represent molecular states (the node icons are just for illustration), and the labeled arcs represent interactions with other molecules in the environment. The models use a biochemical variant of pi-calculus with rate weight as superscripts and with +/- for binding and unbinding.

tactic relationships between genes, genomes, and proteins. An entirely new avenue of research is the investigation of the semantic equivalences of biological entities populating complex networks of interactions. This approach could lead to new visions of systems and reinforce the need for computer science to enhance systems biology.

Biology is a data-intensive science. Biological systems are huge collections of in-

differential equations, Markov chains, or Petri nets generated from the model. Other techniques may be centered on the model description (the language in which the model is written). For example, we may want to know whether two different model descriptions actually represent the same behavior, by some measure of behavior equivalence. This kind of model correspondence can arise, for example, from apparently different biological systems that work by the same fundamental principles. A similar question is whether we can simplify (abstract) a model description and still preserve its behavior, again by some measure of behavior equivalence that may mask some unimportant detail.

Behavioral equivalences are in fact a primary tool in computer science for verifying computing systems. For instance, we can use equivalences to ensure that an implementation is in agreement with a specification, abstracting as much as possible from syntactic descriptions and instead focusing on the semantics (dynamic) of specifications and implementations. So far, biology has focused on syntactic

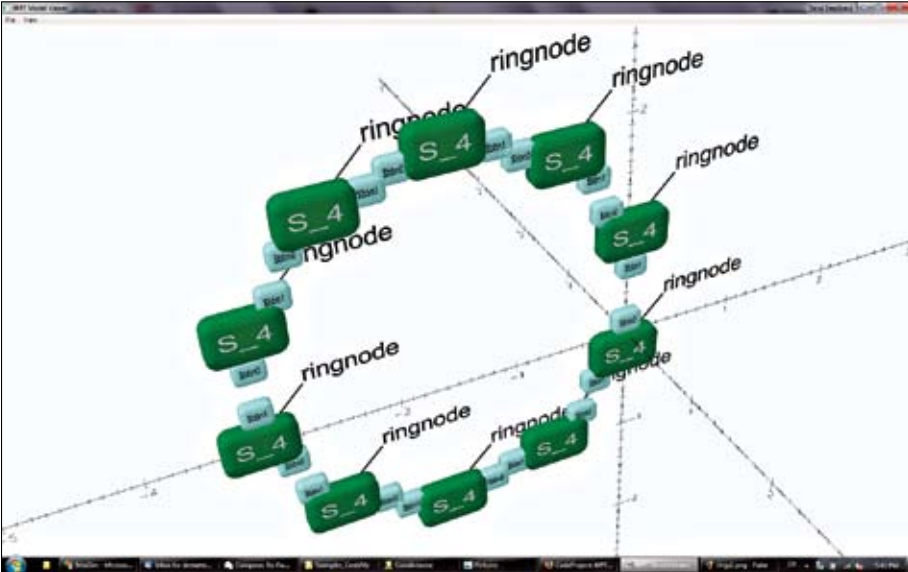


FIGURE 2.
The green S boxes in the diagram represent entities populating the biological system under consideration. The light blue rectangles attached to the green boxes represent the active interfaces/ domains available for complexation and decomplexation. The diagram shows how the simulation of the BlenX specification formed a ring complex and provides the position and the connections between boxes for inspection.

teracting components. The last decade of research has contributed to identifying and classifying those components, especially at the molecular level (gene, metabolites, proteins). To make sense of the large amount of data available, we need to implicitly represent them in compact and executable models so that executions can recover the available data as needed. This approach would merge syntax and semantics in unifying representations and would create the need for different ways of storing, retrieving, and comparing data. A model repository that represents the dynamics of biological processes in a compact and mechanistic manner would therefore be extremely valuable and could heighten the understanding of biological data and the basic biological principles governing life. This would facilitate predictions and the optimal design of further experiments to move from data collection to knowledge production.

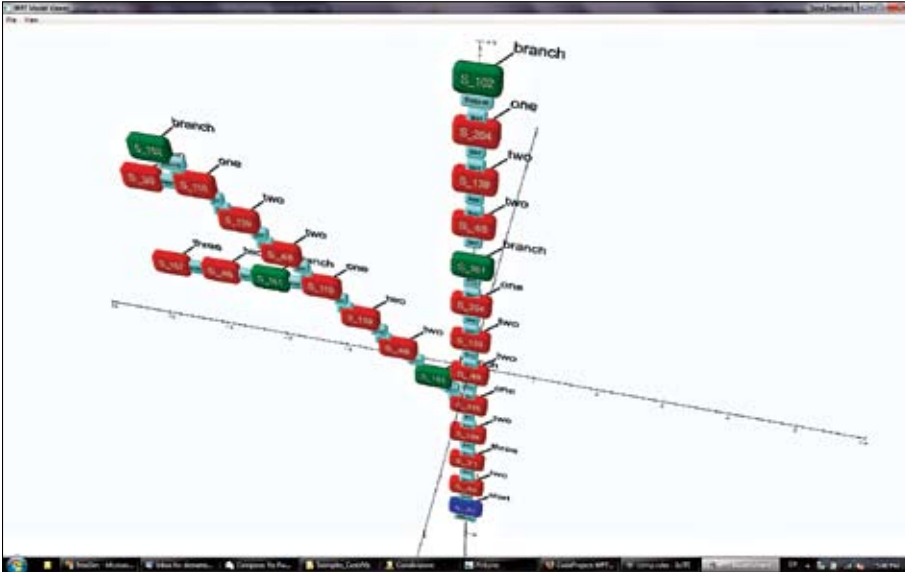


FIGURE 3.

The green, red, and blue *S* boxes in the diagram represent different species populating the biological system under consideration. The light blue rectangles attached to the boxes represent the active interfaces/domains available for complexation and decomplexation. The diagram elucidates how patterns are formed in morphogenesis processes simulated by BlenX specifications.

ANALYSIS VISUALIZATION

Executable models need visualization to make their execution interactive (to dynamically focus on specific features) and reactive (to influence their execution on the fly). Execution is one form of analysis; other analysis methods will need visualization as well. For complex systems, the normal method of “batch” analysis, consisting of running a complex analysis on the model and then mining the output for clues, needs to be replaced with a more interactive, explorative approach.

Model abstraction is an important tool for managing complexity, and we can envision performing this activity interactively—for example, by lumping components together or by hiding components. The notion of lumping will then need an appropriate visualization and an appropriate way of relating the behavior of the original components to the behavior of the lumped components. This doesn’t mean visualizing the modeling language, but rather visualizing an abstraction function between

models. We therefore suggest visualizing the execution of programs/models in such a way that the output is linked to the source code/model specification and the graphical abstraction performed by the end user is transformed into a formal program/model transformation. The supporting tool would then check which properties the transformation is preserving or not preserving and warn the user accordingly.

All the above reinforces the need for a formal and executable language to model biology as the core feature of an *in silico* laboratory for biologists that could be the next-generation high-throughput tool for biology.

ACKNOWLEDGMENTS

The authors thank Andrew Phillips and Lorenzo Dematté for preparing the figures.

REFERENCES

- [1] P. Nurse, “Life, Logic and Information,” *Nature*, vol. 454, pp. 424–426, 2008, doi: 10.1038/454424a.
- [2] J. Fisher and T. Henzinger, “Executable Cell Biology,” *Nature Biotechnology*, vol. 25, pp. 1239–1249, 2007, doi: 10.1038/nbt1356.
- [3] C. Priami, “Algorithmic Systems Biology: An opportunity for computer science,” *Commun. ACM*, June 2009, doi: 10.1145/1506409.1506427.
- [4] A. Regev and E. Shapiro, “Cells as computation,” *Nature*, vol. 419, p. 343, 2002, doi: 10.1038/419343a.
- [5] P. Degano and C. Priami, “Non-interleaving semantics of mobile processes,” *Theor. Comp. Sci.* vol. 216, no. 1–2, pp. 237–270, 1999.
- [6] M. L. Guerriero, J. Heath, and C. Priami, “An automated translation from a narrative language for biological modelling into process algebra,” *Proc. of CMSB 2007*, LNBI 4695, 2007, pp. 136–151, doi: 10.1007/978-3-540-75140-3_10.
- [7] M. L. Guerriero, A. Dudka, N. Underhill-Day, J. Heath, and C. Priami, “Narrative-based computational modelling of the Gp130/JAK/STAT signalling pathway,” *BMC Syst. Biol.*, vol. 3, no. 1, p. 40, 2009, doi: 10.1186/1752-0509-3-40.
- [8] S. Efroni, D. Harel, and I. R. Cohen, “Reactive Animation: Realistic Modeling of Complex Dynamic Systems,” *Computer*, vol. 38, no. 1, pp. 38–47, Jan. 2005, doi: 10.1109/MC.2005.31.
- [9] A. Phillips, L. Cardelli, and G. Castagna, “A Graphical Representation for Biological Processes in the Stochastic Pi-calculus,” *Trans. Comput. Syst. Biol.*, VII - LNCS 4230, 2006, pp. 123–152, doi: 10.1007/11905455_7.
- [10] L. Dematté, C. Priami, and A. Romanel, “The BlenX Language: a tutorial,” *Formal Meth. Comput. Syst. Biol.*, LNCS 5016, 2008, pp. 313–365, doi: 10.1145/1506409.1506427.
- [11] L. Dematté, C. Priami, and A. Romanel, “The Beta Workbench: a computational tool to study the dynamics of biological systems,” *Brief Bioinform.*, vol. 9, no. 5, pp. 437–449, 2008, doi: 10.1093/bib/bbn023.
- [12] C. Priami, “Stochastic pi-calculus,” *Comp. J.*, vol. 38, no. 6, pp. 578–589, 1995, doi: 10.1093/comjnl/38.7.578.
- [13] A. Phillips and L. Cardelli, “Efficient, Correct Simulation of Biological Processes in Stochastic Pi-calculus,” *Proc. Comput. Meth. Syst. Biol.*, Edinburgh, 2007, pp. 184–199, doi: 10.1007/978-3-540-75140-3_13.