BMA: Visual Tool for Modeling and Analyzing Biological Networks

David Benque[1], Sam Bourton[1], Caitlin Cockerton[1], Byron Cook[1], Jasmin Fisher[1], Samin Ishtiaq[1], Nir Piterman[2], Alex Taylor[1], and Moshe Y. Vardi[3]

[1] Microsoft Research Cambridge, [2] University of Leicester, [3] Rice University

Abstract. BioModel Analyzer (BMA) is a tool for modeling and analyzing biological networks. Designed with a lightweight graphical user interface, the tool facilitates usage for biologists with no previous knowledge in programming or formal methods. The current implementation analyzes systems to establish stabilization. The results of the analysis—whether they be proofs or counterexamples—are represented visually. This paper describes the approach to modeling used in BMA and also notes soon-to-be-released extensions to the tool.

Tool location: http://biomodelanalyzer.research.microsoft.com/ [1]

1 Introduction

In recent years, the verification community has seen a huge increase in the use of its tools for modeling and analyzing biological systems (*e.g.* [7–9, 13], etc). It is notable, however, that in almost all cases the biologists who want to access these tools need a proficiency in computer programming or have to rely on help from computer scientists. The difficulty here is that current tools require sophisticated knowledge of both modelling and analysis techniques, as well as experience with how to combine them. This presents a major barrier to the adoption of formal methods in biology and limits the extent to which its tools can be exploited.

This paper describes BioModel Analyzer (BMA – read "bee-ma"), a graphical tool for the construction and analysis of biological models. The goal of this tool is to support the construction of models using visual notations familiar to specialists in biology, not computer science. More generally, it is intended to illustrate how those with an expertise in biology and an interest in biological modeling, namely biologists, can be given direct access to formal modeling and analysis techniques.

The challenge in this domain is the mismatch between the demands imposed by formal verification tools and the ways biologists think about and compose models. Formal verification tools require models to be specified using logical formalisms whereas biologists tend to see their models in spatial and temporal terms, making distinctions between cells, proteins, genes, etc. and tracing the pathways between them. For example, in order to perform formal verification/analysis graphical models can require precise formal semantics, and yet biologists are, for the most part, unfamiliar with these logical forms. Furthermore,

[1] Usage requires Microsoft Silverlight to be installed on your system

the results of formal analysis (*e.g.* abstract counterexamples, invariants, proofs, etc) are often presented in ways that are unintuitive to biologists. With these issues in mind, BMA has been purposefully designed to fit into biologists' existing ways of understanding and working with biological models, whilst exploiting the benefits of formal verification techniques. Furthermore, the tool aims to present the results of analysis so that it is intelligible to biologists and facilitates their further scientific investigations.

Related work. There is a large body of work on usage of formal methods in biological modeling. For example, BioSpice [12] is a repository for many such tools and projects. Here, our focus is to support a very specific level of abstraction and, currently, very limited analysis. This allows us to focus on the user interface and connection between formal verification analysis tools and non-CS experts.

Several attempts have been made to make analysis and modeling tools easier to use for experimental biologists. For example, an English to formal model translator has been proposed [6], as well as a notation describing cells through tables of possible transitions [1]. Graphical notations have also been used for such purposes (*e.g.* [5], [4]). Our approach is similar to that in [5], where a graphical interface is used to create models. However, while their approach requires the biologist to supply state machines that produce the required behavior and programming in either Java or C++ to, e.g., coordinate execution and start it, our aim is to protect the non-CS expert from the need in such knowledge. In our approach the user supplies the rules that govern behavior and induce the state machines. The distinguishing characteristics of BMA are a) its high-performance analysis engine [3] and b) its focus on improving the interface between the biologist and the formal verification/analysis tool.

Motivated by SBML [10], BMA supports output to a custom XML format. Thus, it is possible to interface other analysis tools with BMA models.

2 Designing biological networks

BMA focuses on a very limited domain of modeling. Our models are composed of one or more cells, cell elements (*i.e.*, proteins), and connections that specify the relations between these elements. These elements represent the biological components very abstractly and in high level. As mentioned, BMA is intended to support the design of models using graphical notations. This notation has been intentionally designed to be familiar to biologists and match the representations they commonly use in modeling (*e.g.*, [11, 16, 18]). These graphical models produce an underlying semantic layer based on Qualitative Networks (QN) [15]. Thus, a drawing gives rise to a model which can be automatically analyzed. All advanced features of QNs are available to advanced users.

Fig. 1 shows an example of a simple model. On the canvas there is an isolated protein and two cell membranes, containing proteins. Each cell has two receptors (in green) and the arrows show a cascade that flows from the cell on the left to the cell on the right. The arrows and bar-arrows are the usual notation to indicate activation (positive influence) and inhibition (negative influence) between proteins. In QNs, these relationships are translated into rules that govern the behavior of proteins based on the values of proteins affecting them.

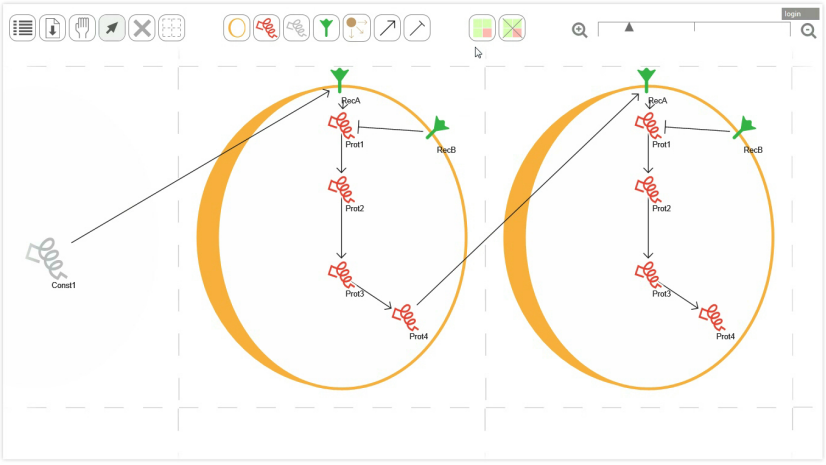

Fig. 1. A small model including two cells, receptors (green shape on the membranes), cell signalling (red coils), and a protein in the outer cell surrounding (grey coil).

The user constructs the models by dragging and dropping elements onto a gridded canvas. Cell membranes that are added fill an entire square in the grid. These have no functional use in the analysis and simply allow the user to pictorially represent a cell. Proteins can be placed either in or outside of these membranes. They can be represented as either receptors, that lie on the cell membrane, or as stand-alone proteins in or outside cells. Connections in the form of arrows or bar-arrows can be drawn between the proteins.

Each protein can be named and given a finite value range to represent the protein's concentration level. For example, a range of [0..3] for a protein might signify concentration levels "off", "low", "medium", or "high". In Fig. 2 we see the additional window in which the name and value ranges are set.

Semantically, a graphical BMA model takes the form of a transition system, which can be analyzed using existing formal analysis tools. BMA's underlying analysis is designed to prove stabilization [19]. A model is stabilizing if in every execution all variables (*i.e.*, protein value ranges) eventually reach a single fixed value and there is no possibility of further change. In addition, all executions result in the same fixed value for each protein. The user can execute this analysis and check whether the resulting model is stabilizing by pressing the "proof" button in the tool. Support of more temporal properties is currently developed.

The output of the test for stabilization is displayed graphically. Proteins that result in a fixed value are colored green and annotated by their value. Proteins that do not reach a fixed value are colored red and annotated by their range of possible values. With models that fail to stabilize, the user can access details of the analysis and the individual steps executed by right-clicking on the proof button and selecting the appropriate contextual menu option. Example results from stability analysis are given in Fig. 3. The left image shows a cell from a non-stabilizing model and the right a cell from a stabilizing model.

As sometimes happens with model checking, when properties are proven for a system, the reasons are not always clear. In biology and with BMA, especially,

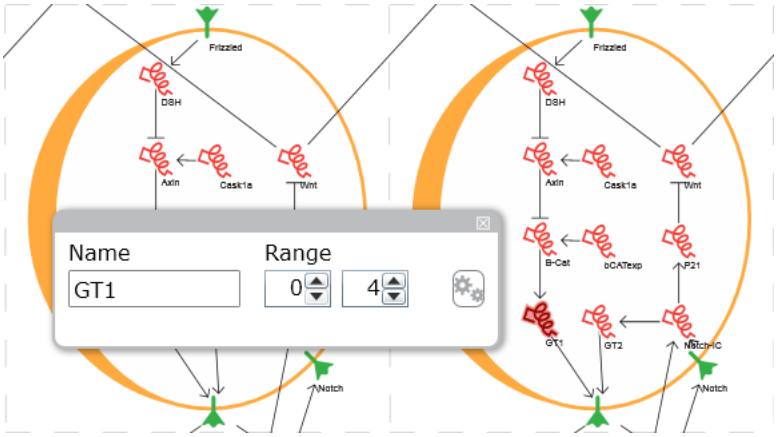

Fig. 2. Partial view of more complex model and window in which protein name and value ranges are entered.

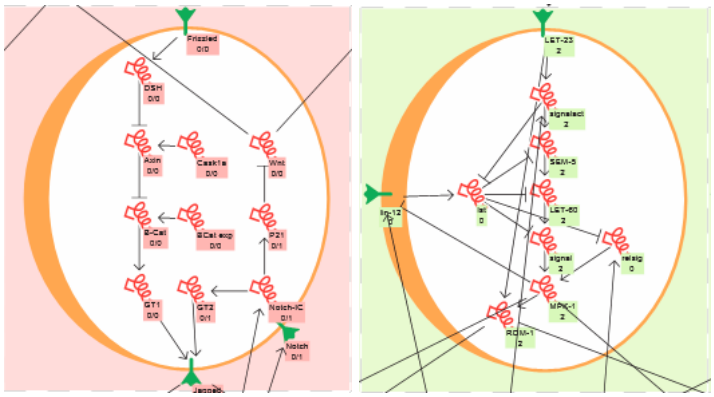

Fig. 3. Cell from non-stabilizing model (left) and cell from stabilizing model (right).

this is problematic because why and how stabilization occurs has important implications for what a biologist might do next in their research. To help users understand why stabilization holds, we are currently working on an animated visual representation of the proof's execution.

3 Analyzing biological networks

As we have noted, the graphical models users produce are formally represented using Qualitative Networks (QN) [15]. The tool automatically translates the graphical models to a QNs. The QNs include variables representing the concentration of proteins as a discrete value in a fixed range. Values of variables change gradually according to interactions between the proteins. QNs and similar formalisms (*e.g.*, genetic regulatory networks [17]) are simple enough to be represented graphically and expressive enough to capture interesting biological phenomena (*e.g.*, [2, 14, 15, 17]).

Qualitative networks. A *qualitative network* (QN) is $Q = (V, T, N)$, where $V = (v_1, v_2, \ldots, v_n)$ is a set of variables ranging over $\{0, 1, \ldots N\}$ and $T =$

$(T_1, \ldots, T_n)$ are their respective target functions. A state of the system is an assignment $s : V \to \{0, 1, \ldots N\}$. Let Σ denote the set of all possible states. A *target function* $T_i \in T$ is $T_i : \Sigma \to \{0, 1, \ldots N\}$. Intuitively, in a given state s, variable v_i "would like" to get the value $T_i(s)$. However, values of variables change by at most 1. The *successor* of state s is s', where for every $v_i \in V$ we have:

$$s'(v_i) = \begin{cases} s(v_i) + 1 & s(v_i) < T_i(s), \\ s(v_i) & s(v_i) = T_i(s), \\ s(v_i) - 1 & s(v_i) > T_i(s). \end{cases} \quad (1)$$

Thus, a QN defines a transition system over Σ. All variables change their value synchronously by following their target functions. We abuse notation and write also $T : \Sigma \to \Sigma$ as the function that associates with a state s its successor s'.

Each protein in the design corresponds to a variable in the underlying Qualitative Network. Perhaps the most complex feature of QNs is the target functions. In order to enable novice users to bypass the need to define target functions, we set a default target function induced by the activations and inhibitions applied to a given protein. Denoted in short as $ave(pos) - ave(neg)$. That is, the weighted average of the proteins that activate the protein minus the weighted average of the proteins that inhibit the protein.

Custom target functions. In the graphical representation, users can customize the target function by clicking on the cog-wheel in a protein's property window (see Fig. 2). Target functions are defined using a small language of possible mathematical operations (*e.g.* $+$, $-$, $/$ max, etc). For example we might use $T_i = N - \frac{v_1+v_2}{2}$, which models a situation where v_1 and v_2 affect v_i negatively. With no negative influence (i.e, when the values of v_1 and v_2 are 0), v_i aims to settle in its maximal value. When v_1 and v_2 are high, v_i aims to decrease. This models a situation in which a protein is constitutively produced unless proteins v_1 and v_2 inhibit its production. A more complex target function is $T_i = max(0, min(2 - v_1, 1)) \times v_2$. In this case if v_1 is 0 or 1 then v_i aims to follow v_2. However, if v_1 is more than 1, then v_i aims to decrease to 0. This models a situation when above a certain threshold protein v_1 strongly inhibits v_i but otherwise protein v_2 positively influences v_i.

Stabilization. We say that a state s is *recurring* if it is possible to reach s from itself after a finite number of applications of T. That is, for some $i \geq 1$ we have $s = T^i(s)$. We note that the number of states of a QN is finite, hence, the set of recurring states cannot be empty. A QN is *stabilizing* if for some state s we have $s = T(s)$ and no other state is recurring.

BMA attempts to prove stabilization using an approach from [3] which combines a search for thread-modular proofs of liveness together with techniques from abstract interpretation and the intervals domain.

4 Conclusion

In this paper we introduce a new graphical tool for modeling and analyzing biological networks. The tool is intended for use by biologists. The current implementation analyzes systems to establish stabilization, with support for additional temporal properties in development. The results of BMA's analysis are

represented visually: counterexamples are displayed by showing regions of the network that could take on additional values. In the future proofs will be demonstrated by visually displaying the lemmas found during the proof search. We are currently working on introducing additional types of analysis. On the one hand, introducing analysis that does not require the user to specify logical formulas. On the other hand, finding intuitive (and graphical) ways to allow users to specify logical formulas and feeding back the output of analysis.

References

1. Amir-Kroll, H., Sadot, A., Cohen, I., and Harel, D. GemCell: A generic platform for modeling multi-cellular biological systems. *TCS 391*, 3 (2008).
2. Beyer, A., Thomason, P., Li, X., Scott, J., and Fisher, J. Mechanistic insights into metabolic disturbance during type-2 diabetes and obesity using qualitative networks. *TCSB 12* (2010).
3. Cook, B., Fisher, J., Krepska, E., and Piterman, N. Proving stabilization of biological systems. In *VMCAI* (2011).
4. Danos, D., Feret, J., Fontana, W., Harmer, R., and Krivine, J. Rule-based modelling and model perturbation. *TCSB 11* (2009).
5. Efroni, S., Harel, D., and Cohen, I. R. Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation. *Genome Res 13*, 11 (2003).
6. Errampalli, D. D., Priami, C., and Quaglia, P. A formal language for computational systems biology. *Omics 8*, 4 (2004).
7. Fisher, J., Piterman, N., Hajnal, A., and Henzinger, T. Predictive modeling of signaling crosstalk during *c. elegans* vulval development. *PLoS Comp Bio 3*, 5 (2007).
8. Ghosh, R., Tiwari, A., and Tomlin, C. Automated symbolic reachability analysis; with application to delta-notch signaling automata. In *HSCC* (Prague, Czech Republic, 2003), vol. 2623.
9. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., and Tymchyshyn, O. Probabilistic model checking of complex biological pathways. In *CMSB* (2006).
10. Hucka, M., A., F., Sauro, H., Bolouri, H., Doyle, J., and Kitano, H. The systems biology markup language (SBML). *Bioinfo 19*, 4 (2003), 524–31.
11. Klerkx, E., Alarcón, P., Waters, K., Reinke, V., Sternberg, P., and Askjaer, P. Protein kinase vrk-1 regulates cell invasion and egl-17/fgf signaling in C. elegans. *Dev Bio 335*, 1 (2009).
12. Kumar, S., and Feidler, J. BioSPICE: A computational infrastructure for integrative biology. *OMICS 7*, 3 (2003), 225.
13. Li, C., Nagasaki, M., Ueno, K., and Miyano, S. Simulation-based model checking approach to cell fate specification during *C. elegans* vulval development by hybrid functional petri net with extension. *BMC Sys Bio 3*, 42 (2009).
14. Sanchez, L., and Thieffry, D. Segmenting the fly embryo: a logical analysis fo the *pair-rule* cross-regulatory module. *J of Theo Bio 244* (2003).
15. Schaub, M., Henzinger, T., and Fisher, J. Qualitative networks: A symbolic approach to analyze biological signaling networks. *BMC Sys Bio 1*, 4 (2007).
16. Sundaram, M. V. The love-hate relationship between Ras and Notch. *Genes Dev 19*, 16 (2005).
17. Thomas, R., Thieffry, D., and Kaufman, M. Dynamical behaviour of biological regulatory networks—i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bull of Math Bio 55*, 2 (1995).
18. Yoo, A. S., Bais, C., and Greenwald, I. Crosstalk between the EGFR and LIN-12/Notch pathways in C. elegans vulval development. *Science 303*, 5658 (2004).
19. Zotin, A. *The Stable state of organisms in thermodynamic bases of biological processes: Physiological Reactions and Adaptations.* 1990.