Biomodel Engineering – From Structure to Behavior

Rainer Breitling[1], Robin A. Donaldson[2], David R. Gilbert[2,3], and Monika Heiner[4]

[1] Groningen Bioinformatics Centre, University of Groningen, Kerklaan 30, 9751 NN Haren, The Netherlands
r.breitling@rug.nl
[2] Bioinformatics Research Centre, University of Glasgow, Glasgow G12 8QQ, UK
[3] School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge, UB8 3PH, UK
[4] Brandenburg University of Technology at Cottbus, Dept. of Computer Science, Postbox 10 13 44, D 03013 Cottbus, Germany

Abstract. Biomodel engineering is the science of designing, constructing and analyzing computational models of biological systems. It forms a systematic and powerful extension of earlier mathematical modeling approaches and has recently gained popularity in systems biology and synthetic biology. In this brief review for systems biologists and computational modelers, we introduce some of the basic concepts of successful biomodel engineering, illustrating them with examples from a variety of application domains, ranging from metabolic networks to cellular signaling cascades. We also present a more detailed outline of one of the major techniques of biomodel engineering – Petri net models – which provides a flexible and powerful tool for building, validating and exploring computational descriptions of biological systems.

Keywords: Systems biology, Petri nets, computational modeling, differential equations.

1 What Is Biomodel Engineering?

Biomodel engineering takes place at the interface of computing science, mathematics, engineering and biology. It provides a systematic approach for designing, constructing and analyzing computational models of biological systems. Some of its central concepts are inspired by efficient software engineering strategies. Biomodel engineering does not aim at engineering biological systems *per se,* but rather aims at describing their structure and behavior, in particular at the level of intracellular molecular processes, using computational tools and techniques in a principled way. The two major application areas of biomodel engineering are systems biology and synthetic biology. In the former, the aim is the design and construction of models of existing biological systems, which explain observed properties and predict the response to experimental interventions; in the latter, biomodel engineering is used as part of a general strategy for designing and constructing synthetic biological systems with novel functionalities.

C. Priami et al. (Eds.): Trans. on Comput. Syst. Biol. XII, LNBI 5945, pp. 1–12, 2010.

2 Building a Computational Model

In this paper we focus on computational models in the narrow sense, i.e. models that use formal computational descriptions or algorithms to describe and understand natural phenomena (these are also sometimes called "executable models" [1]). There are many strategies for building a predictive computational model of a biological system. All of them involve a high level of abstraction and need to start with a careful analysis of what the model is supposed to achieve (requirements capture analysis). In that context, it is most important to determine what kind of properties the biologists can observe (How do they measure it? What are the accuracy, scale and temporal resolution of the measurements?), and which components of the system are known to influence system behavior and how they can be manipulated (By knocking down genes? By inhibiting of enzymes? By changing environmental conditions?). These factors determine which type of model should be built and which variables have to be included in it. In general, models (of intracellular pathways) can be *qualitative*, describing the topology or wiring diagram of the biochemical entities, or *quantitative*, with the addition of kinetic information, e.g. equations and rates. The latter can be in general *continuous* or *stochastic* [2, 3].

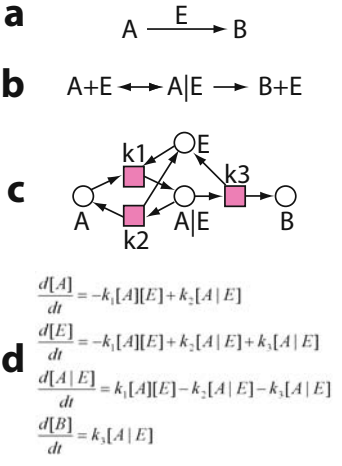

Fig. 1. A simple enzyme-catalyzed reaction in four different representations. (**a**) As a biological cartoon, (**b**) as a set of chemical equations, (**c**) as a computational Petri net model, and (**d**) in the form of the corresponding system of differential equations. The equivalence of the last two representations can be clearly seen. The Petri net can be uniquely translated into differential equations (although the reverse is not generally the case).

If a model of a cellular system has to be built from scratch, the next step is to describe all relevant causal connections between the components in the system (chemical reactions, regulatory influences, physical interactions). This is most conveniently done in a graphical fashion, which comes as close as possible to the cartoon models of biological systems that are common in the biological literature (**Figure 1a**). The main challenge is a proper disambiguation of the informal descriptions provided in these

cartoons. A popular software program for this purpose is, for example, Cell Illustrator [4-6], which uses a graphical user interface that employs an intuitive graphical notation to describe interactions between molecules. This is then translated into a Petri net, one of the most powerful and general types of computational models. More extensive reviews of this and alternative computational model types are provided in [1, 7, 8].

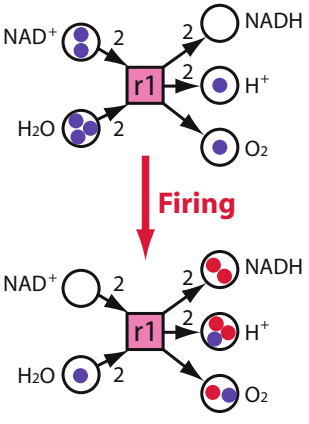

Fig. 2. Illustration of the token game of a simple chemical reaction represented as a Petri net

3 Playing the Token Game – An Example of Computational Modeling

To illustrate the properties of computational models, we will briefly describe one of them, Petri nets, in more detail. Petri nets are just one of the many formalisms that can be used in biomodel engineering. They are a graphical tool for the description of complex system behavior that was originally inspired by the description of chemical processes [7, 9, 10]. A Petri net is a graph containing two types of nodes, *places* and *transitions*. In a (bio-)chemical system, places correspond to chemical entities, and transitions to the reactions that convert them. The state of a system (e.g., the current concentration of molecules) is indicated by the amount of *tokens* that are present at each place. Simulating the behavior of a biological system using Petri nets is done by "playing the token game" (**Figure 2**): Whenever there are sufficient tokens at the places upstream of a particular transition, the transition can *fire*, i.e it consumes tokens at upstream places and creates new tokens downstream. Weights on the transitions indicate the number of tokens that are "used" when a transition fires, i.e. they correspond to the stoichiometry of the chemical reaction.

The token game can be played computationally, or using a visualization tool like Snoopy (**Table 1**), which can show how tokens move through the system over time. Such a direct exploration of molecular and signaling fluxes in a system can provide useful insights into the validity of the model and the underlying biology.

Places, tokens and transitions in a Petri net can be interpreted in multiple ways, and different interpretations can be combined in a single model, which explains the

flexibility of Petri net models in biology. For example, places can not only encode chemical compounds, but also different states of compounds, including the status of protein complexes. Tokens can represent individual molecules, but also discrete levels of concentrations (e.g., high–medium–low). Transitions can correspond to mass action chemical reactions, but also to complex enzymatic reactions, complex formation or the transmission of a cellular signal.

The Petri nets described above are classical, discrete nets, where the amount of tokens is always an integer and there is not explicit notion of time (it is only implicitly given in the causality). Continuous Petri nets are not limited in this way and can describe continuous reactions in the same way as differential equations. In fact, they can be directly translated into the corresponding differential equation system using simple rules. Additional extensions of the Petri net formalism include *stochastic modeling*, which makes use of the fact that the firing of transitions is non-deterministically delayed, so that multiple token games (simulations) of the system can result in quite different outcomes each time. All these Petri nets can employ *hierarchical modeling*, where certain subgraphs in the model can be defined independently and composed into larger models.

4 Automated Model Construction

However, as systems biology is advancing, it is becoming more likely that some form of quantitative system model already exists and initially just needs to be translated into a computational model.

The translation of quantitative systems models into computational models is particularly straightforward for systems of ordinary differential equations, which can be shown to be equivalent to the continuous Petri net model of the system (although it is important to note that only the translation from Petri nets to differential equations is unique, while the reverse is not guaranteed to be the case). For small systems, the translation algorithm is easy to implement manually. For larger systems, it is more convenient to use automated translation tools, for example [11], which can directly convert Systems Biology Markup Language (SBML) descriptions of the differential equations into a Petri Net Markup Language (PNML) description of the corresponding Petri net. Automated translation has also been used on a larger scale to translate entire databases of biological knowledge into the corresponding Petri net description: Nagasaki et al. [12] demonstrated the feasibility of this approach by translating the TRANSPATH database of signaling pathways [13] into a Petri net model, stored in Cell System Markup Language (CSML), yet another systems biology file format.

Another obvious target for translation into a computational model are the stoichiometric matrices used for Flux Balance Analysis [14] and related popular approaches [15-17]. A stoichiometric matrix corresponds to the adjacency matrix of a Petri net graph. It has been shown that many of the key ideas of metabolite network analysis, such as elementary modes/extreme pathways [18] and metabolic pools [19] are equivalent to traditional concepts of Petri net theory, such a T(ransition)- and P(lace)-invariants [18, 20, 21]. Even extended models, which include Boolean descriptions of protein complexes, alternative splicing and isozymes [22], and transcriptional regulatory interactions [23], can be unified by translation into, for instance, a Petri net model to facilitate model management and exploration.

5 Managing Models in Biomodel Engineering

Biomodel engineering goes well beyond building a computational model. In particular it provides powerful ways for managing collections of models, including different versions of the same model. In biomodel engineering, computational models are considered in the same way as large computer programs in software engineering. For example, it is important to implement rules for version control, which document sources of information and each step in the model construction process. This makes it possible to explore alternative hypotheses about the system structure. Another important aspect is the systematic identification of building blocks and sub-models (modules), which can be described independently. It will be possible to reuse model modules across the model, in a similar fashion to objects in computer programming. They can be modified in a hierarchical manner, so that there may be one general module for enzymatic reactions, and specialized instances for reactions with different kinetics, which inherit the main structure and can in turn be modified to allow different mechanisms of enzyme inhibition.

Within a model management system, version control and modularity permit an efficient distributed multi-modeler approach to model building, which is essential for genome-scale modeling by large teams of curators. All of the modelers have access to the same database of building blocks, allowing for rapid composition of larger models.

6 Model Behavior Checking

Another large and very promising area of research in biomodel engineering is based on the concept of model behavior checking. Once a computational (or mathematical) model of a biological system has been created, it needs to be validated in a principled way. Does it produce reasonable predictions of system behavior? Sometimes this can be done ‘by eye’, but for large models this is not an option. Most importantly, it is often necessary to explore model behavior in a wide range of conditions. For this purpose, the technique of formal model behavior checking can be very useful. It is based on the principles of model checking in traditional computer engineering, an approach that uses formal logics to express certain properties of a computer system, for example whether it is safe against deadlock and other forms of systems failure [21, 24-28], which are then proved in a rigorous way. The approach we present here differs from this classical model checking in that it can also be used to check the behavior of real biological systems, e.g. to check whether the observed time series behavior in a wet laboratory experiment conforms to some formally defined properties.

Simulative model behavior checking comprises four main components:

Model. A model of the system of interest. This can be implemented in a wide range of formalisms.

Simulator. A program that produces simulated time courses from the system model. These can be stochastic or continuous simulations of a single model, or variant models that cover a range of possible parameters.

Property. A temporal logic formula describing a (desired or expected) biochemical behavior. Such a logic is essentially a specialized "mini-language" to make statements about model behavior, such as "the concentration of [P] is oscillating" or "the peak of [X] occurs at least 5 minutes before the peak of [Y]" or "an increase of activator [A] is followed by a transient repression of its target [B]".

Model Checker. A program which tests whether the model exhibits the specified temporal property, using a collection of simulated runs as input.

There are five major areas of application for model behavior checking in biology:

1. Model validation: Does the model behave in the way we expect?

2. Model comparison: How similar are two models, independent of their underlying formalisms?

3. Model searching: In a database of alternative models, which of them show a particular property? This can be used to select among competing descriptions of a system produced by various research groups. Given a large database of models (either a general collection or variants of the same model), one can use model behavior checking to perform systematic database queries, such as "find all models that show oscillatory behavior under some conditions" or "find all descriptions of this signaling pathway that are transiently active after growth factor stimulation"

4. Model analysis: In a collection of variants of a model (e.g., *in silico* gene knock-outs), which models show a certain behavior? E.g., how many knock-outs lead to a loss of oscillating behavior?

5. Model construction: Which modifications of a model lead to a desired property? Modifications can involve changes in kinetic parameters or initial concentrations, but they can also be more complex, for example changing the topology of the model by removing or even adding new components. How to do this efficiently is still an active area of research.

The model behavior checking approach can be combined with automated model modification (adding/removing/modulating edges in the network; or adding/removing molecular species in the network). Given a particular collection of behaviors that a correct model should exhibit, one can for example find the minimum number of modifications that are necessary for a given model that does not yet fulfill the conditions or detect the critical components that are most often added/removed to achieve a particular behavior [25].

Ideally, such analysis would be performed in a comprehensive biomodel engineering environment, which supports all of the above. First steps in this direction are currently being made, but no fully featured system is available yet. Eventually, biomodel engineering would be used as part of an integrated biosystem design and construction process, i.e. it will be performed in close interaction with synthetic biologists.

7 Computational versus Mathematical Models

The traditional approach of describing biological system dynamics is mathematical and is based on differential equations. They are familiar to many biologists, for

example in the form of mass action equations, Lotka-Volterra equations or Michaelis-Menten enzyme kinetics. Mathematical and computational models are closely related. It can be shown that the graphical model description shown in **Figure 1c** can be directly translated into the differential equation system in **Figure 1d**, and vice versa. Both of them describe the same biological behavior. So, how do computational models differ from differential equation models?

The main difference is the systematic structure enforced by the graphical form of the computational model. Differential equations lack this rigid structure; they can be re-arranged and transformed in infinite ways. In our view, this leads to three major advantages of a computational model description:

Firstly, the major conceptual advantage is the gain in explanatory power, due to the formal structure of the computational models. In a computational model, each component corresponds to a biological entity that can be the target of an experiment (or at least a thought experiment). The model thus implies a well-defined set of *modified* systems, and predicts the response to a range of experimental interventions. Differential equation systems don't allow this straightforward mapping onto biological entities that could be manipulated experimentally. To be interpreted and to be useful as a predictive biological tool, mathematical modeling critically depends on such a mapping, but this is provided only implicitly in the analysis process, rather than as an integral part of the model engineering process as for computational models.

Secondly, their formal structure makes computational models eminently suitable for the storage of a diverse biological knowledge base: detailed quantitative information (e.g., kinetic rate laws), linear constraints (e.g., stoichiometric ratios), Boolean relationships (e.g., gene regulatory patterns), and qualitative data (e.g., protein-protein interactions) can all be stored and analyzed in one unified framework. For example, as described above**,** Petri nets can be both continuous and qualitative without a change in the overall formalism, and they can even incorporate stochastic elements, which are important for an accurate simulation of many biology processes [29-31]. This multiple nature also makes it possible to first validate that the structure of a model is reasonable (using topology-based analysis methods), before going on to quantitatively parameterize the description.

Finally, the graphical structure of computational models allows straightforward exploitation of the modularity of biological systems. The structure of many biological processes, ranging from enzymatic mechanisms to regulatory motifs, occurs repeatedly in the system. Once its basic structure has been described, it can be re-used as a building block across the system, with only some re-parameterization necessary. Similar plug-and-play approaches, which are essential for structured biomodel engineering, are not possible for pure mathematical models. This ability to define structural modules is a major advantage for the use of computational modeling in synthetic biology: for instance, one can envisage that each standard biological part (such as the "BioBricks" [32]) comes along with a computational model of its function, so that predictive models of new systems can rapidly be assembled. Even automatic model composition and exploration of model behavior would be possible, making biomodel engineering a key tool for the new generation of large-scale synthetic bioengineering.

8 Case Studies of Biomodel Engineering

Many groups have recently shown the power of computational models for understanding biological systems. In this brief review we can only present a few selected examples that highlight the range of systems biology applications.

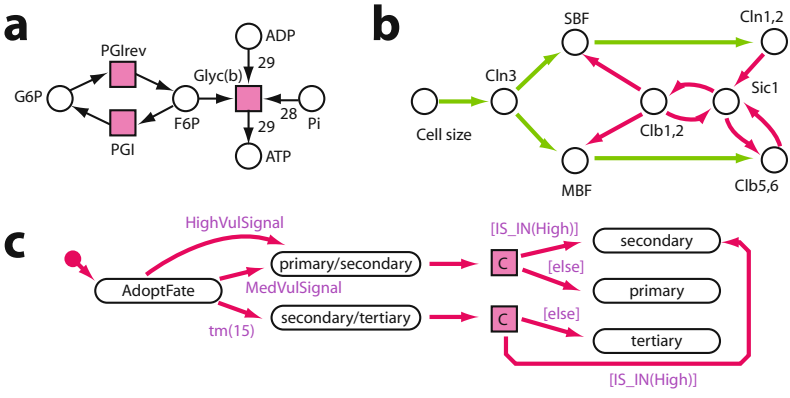

Fig. 3. Examples of computational models for different types of biological networks. (**a**) Part of the Petri net model of sucrose breakdown in the potato tuber discussed by Koch et al. [33]. (**b**) A subgraph of the simplified Boolean description of the yeast cell-cycle network presented by Li et al. [34]. (**c**) The main statechart of the vulval precursor cells in the model of *C. elegans* vulva development by Fisher et al. [35].

Metabolic networks were the first molecular biological systems to be described and analyzed by differential equations and related mathematical models, and they were also the first to be modeled by computational models such as Petri nets [36, 37]. Computational models have been shown to allow all the analytical methods that are used in the successful Flux Balance Analysis and related techniques [15]. For example, using a Petri net model of sucrose breakdown in the potato tuber, Koch et al. [33] use invariant analysis to identify sets of conserved co-factors (P-invariants) and independent sub-pathways (T-invariants) (**Figure 3a**).

Gene regulatory networks also were an early application area of biomodel engineering. Their topology can be determined experimentally on a large scale. The directionality of regulatory connections is also accessible, but the strength of the interaction is very hard to measure. Therefore, Boolean descriptions are a natural way of computationally modeling gene regulation. Li et al. [34] described the cell-cycle regulatory network of yeast as a dynamic Boolean network and identified its global dynamic properties, including a surprising level of robustness against network perturbations (**Figure 3b**).

Cellular signaling pathways are another popular target for computational modeling. For example, Ruths et al. [38] exploit the intermediate position of Petri nets between purely topological descriptions and fully kinetic descriptions to come up with a non-parametric strategy for characterizing the dynamics of cellular signaling. Their

method is fast and does not require detailed kinetic information, while still providing testable predictions about the temporal change of signaling intermediates, based on repeatedly playing the token game.

Developmental differentiation processes are one of the main challenges for large-scale system modeling, because they combine gene regulation and cellular signaling across multiple temporal and spatial scales. So far, computational models have been applied mainly to small, well-characterized model systems. For example, vulval development in the worm *C. elegans* involves only three cell types and was successfully modeled using interacting state machines [35] (**Figure 3c**). The model included different modes of crosstalk between signaling pathways, and the analysis was carried out using model checking to predict novel feedback loops that are required to explain the system behavior and robust patterning observed in biological experiments.

9 Conclusions

In this paper we have shown how using concepts from traditional computing science can be used to create more powerful models of biological systems. We have demonstrated that computational models are more than just quantitative descriptions of the system. Their structure enables systematic predictions of model behavior, and model behavior checking can be used to explore options for updating a model after new experimental evidence becomes available. The modularity of computational models makes composition of genome-scale models based on standardized building blocks much more efficient. The intuitive graphical nature of computational models and their ability to integrate qualitative/structural and quantitative/kinetic information contributes to their raising popularity in many areas of biology. We predict that we will see a continued surge of biomodel engineering throughout systems biology and synthetic biology in the coming years.

References

1. Fisher, J., Henzinger, T.A.: Executable cell biology. Nat. Biotechnol. 25, 1239–1249 (2007)
2. Gilbert, D., Fuss, H., Gu, X., Orton, R., Robinson, S., Vyshemirsky, V., Kurth, M.J., Downes, C.S., Dubitzky, W.: Computational methodologies for modelling, analysis and simulation of signalling networks. Brief Bioinform. 7, 339–353 (2006)
3. Gilbert, D., Heiner, M., Lehrack, S.: A Unifying Framework for Modelling and Analysing Biochemical Pathways Using Petri Nets. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 200–216. Springer, Heidelberg (2007)
4. Nagasaki, M., Doi, A., Matsuno, H., Miyano, S.: Computational modeling of biological processes with Petri Net-based architecture. In: Chen, Y. (ed.) Bioinformatics Technologies, pp. 179–242. Springer, Heidelberg (2005)
5. Matsuno, H., Li, C., Miyano, S.: Petri net based descriptions for systematic understanding of biological pathways. IEICE Trans Fundam Electron Commun. Comput Sci. E89-A, 3166–3174 (2006)
6. Nagasaki, M., Doi, A., Matsuno, H., Miyano, S.: Genomic Object Net: a platform for modeling and simulating biopathways. Applied Bioinformatics 2, 181–184 (2003)

7. Chaouiya, C.: Petri net modelling of biological networks. Brief Bioinform. 8, 210–219 (2007)
8. Heiner, M., Gilbert, D., Donaldson, R.: Petri nets for Systems and Synthetic Biology. In: Bernardo, M., Degano, P., Zavattaro, G. (eds.) SFM 2008. LNCS, vol. 5016, pp. 215–264. Springer, Heidelberg (2008)
9. Petri, C.A., Reisig, W.: Petri nets. Scholarpedia 3, 6477 (2008)
10. Breitling, R., Gilbert, D., Heiner, M., Orton, R.: A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. Brief Bioinform. 9, 404–421 (2008)
11. Shaw, O., Koelmans, A., Steggles, J., Wipat, A.: Applying Petri Nets to Systems Biology using XML Technologies. Technical Report Series. University of Newcastle upon Tyne (2004); CS-TR-827
12. Nagasaki, M., Saito, A., Li, C., Jeong, E., Miyano, S.: Systematic reconstruction of TRANSPATH data into cell system markup language. BMC Syst. Biol. 2, 53 (2008)
13. Krull, M., Pistor, S., Voss, N., Kel, A., Reuter, I., Kronenberg, D., Michael, H., Schwarzer, K., Potapov, A., Choi, C., et al.: TRANSPATH: an information resource for storing and visualizing signaling pathways and their pathological aberrations. Nucleic Acids Res. 34, D546–D551 (2006)
14. Edwards, J.S., Palsson, B.O.: Metabolic flux balance analysis and the in silico analysis of Escherichia coli K-12 gene deletions. BMC Bioinformatics 1, 1 (2000)
15. Reed, J.L., Famili, I., Thiele, I., Palsson, B.O.: Towards multidimensional genome annotation. Nat. Rev. Genet. 7, 130–141 (2006)
16. Becker, S.A., Feist, A.M., Mo, M.L., Hannum, G., Palsson, B.O., Herrgard, M.J.: Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. Nat. Protoc. 2, 727–738 (2007)
17. Palsson, B.O.: Systems Biology: Properties of Reconstructed Networks. Cambridge University Press, Cambridge (2006)
18. Papin, J.A., Stelling, J., Price, N.D., Klamt, S., Schuster, S., Palsson, B.O.: Comparison of network-based pathway analysis methods. Trends Biotechnol. 22, 400–405 (2004)
19. Famili, I., Palsson, B.O.: The convex basis of the left null space of the stoichiometric matrix leads to the definition of metabolically meaningful pools. Biophys. J. 85, 16–26 (2003)
20. Heiner, M., Koch, I., Will, J.: Model validation of biological pathways using Petri nets–demonstrated for apoptosis. Biosystems 75, 15–28 (2004)
21. Heiner, M., Koch, I.: Petri net based model validation in systems biology. In: Cortadella, J., Reisig, W. (eds.) ICATPN 2004. LNCS, vol. 3099, pp. 216–237. Springer, Heidelberg (2004)
22. Duarte, N.C., Becker, S.A., Jamshidi, N., Thiele, I., Mo, M.L., Vo, T.D., Srivas, R., Palsson, B.O.: Global reconstruction of the human metabolic network based on genomic and bibliomic data. Proc. Natl. Acad. Sci. U S A 104, 1777–1782 (2007)
23. Herrgard, M.J., Lee, B.S., Portnoy, V., Palsson, B.O.: Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in Saccharomyces cerevisiae. Genome Res. 16, 627–635 (2006)
24. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theoretical Computer Science 391, 239–257 (2008)
25. Kwiatkowska, M., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. ACM SIGMETRICS Performance Evaluation Review 35, 14–21 (2008)

26. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of Signalling Pathways using Continuous Time Markov Chains. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 44–67. Springer, Heidelberg (2006)
27. Chabrier, N., Fages, F.: Symbolic model checking of biochemical networks. In: Priami, C. (ed.) CMSB 2003. LNCS, vol. 2602, pp. 149–162. Springer, Heidelberg (2003)
28. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge (1999)
29. Losick, R., Desplan, C.: Stochasticity and cell fate. Science 320, 65–68 (2008)
30. Gillespie, D.T.: Stochastic simulation of chemical kinetics. Annu. Rev. Phys. Chem. 58, 35–55 (2007)
31. Kaern, M., Elston, T.C., Blake, W.J., Collins, J.J.: Stochasticity in gene expression: from theories to phenotypes. Nat. Rev. Genet. 6, 451–464 (2005)
32. Shetty, R.P., Endy, D., Knight Jr., T.F.: Engineering BioBrick vectors from BioBrick parts. J. Biol. Eng. 2, 5 (2008)
33. Koch, I., Junker, B.H., Heiner, M.: Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. Bioinformatics 21, 1219–1226 (2005)
34. Li, F., Long, T., Lu, Y., Quyang, Q., Tang, C.: The yeast cell-cycle network is robustly designed. Proc. Natl. Acad. Sci. U S A 101, 4781–4786 (2004)
35. Fisher, J., Piterman, N., Hubbard, E.J., Stern, M.J., Harel, D.: Computational insights into Caenorhabditis elegans vulval development. Proc. Natl. Acad. Sci. U S A 102, 1951–1956 (2005)
36. Hofestädt, R.: A Petri Net Application of Metabolic Processes. Journal of System Analysis, Modelling and Simulation 16, 113–122 (1994)
37. Reddy, V.N., Mavrovouniotis, M.L., Liebman, M.N.: Petri Net Representation in Metabolic Pathways. In: First International Conference on Intelligent Systems for Molecular Biology 1993, pp. 328–336. AAAI Press, Menlo Park (1993)
38. Ruths, D., Muller, M., Tseng, J.T., Nakhleh, L., Ram, P.T.: The signaling Petri net-based simulator: a non-parametric strategy for characterizing the dynamics of cell-specific signaling networks. PLoS Comput. Biol. 4, e1000005 (2008)

Table 1. A small selection of software and websites for biomodel engineering

	URL
BioNessie. ODE-based pathway construction, simulation and analysis	http://www.bionessie.org
CellDesigner. Graphical editing and simulation of biochemical models	http://www.systems-biology.org/cd/
Cell Illustrator. Draw, model, elucidate and simulate biochemical models; Petri net	http://www.cellillustrator.com/
Charlie. Tool for analyzing Petri net models	http://www.informatik.tu-cottbus.de/~wwwdssz/software/charlie.html
CPN Tools. Editing, simulating and analyzing Coloured Petri Nets	http://wiki.daimi.au.dk/cpntools/cpntools.wiki
MC2. Probabilistic linear-time temporal logic checker	http://www.brc.dcs.gla.ac.uk/software/mc2/
Pathway Logic Assistant. Create, simulate and analyze based on rewriting logic	http://pl.csl.sri.com/software.html
Prism. Probabilistic branching-time temporal logic model checker	http://www.prismmodelchecker.org
Snoopy. Petri net editor, simulator, and analysis interface	http://www.informatik.tu-cottbus.de/~wwwdssz/software/snoopy.html
SPiM. Stochastic pi-calculus simulator	http://research.microsoft.com/~aphillip/spim/
Cell System Markup Language	http://www.csml.org/
Petri Net Markup Language	http://www2.informatik.hu-berlin.de/top/pnml/about.html
Systems Biology Markup Language	http://sbml.org/Main_Page