Foundations of a Query and Simulation System for the Modeling of Biochemical and Biological Processes

M. Antoniotti, F. Park, A. Policiriti, N. Ugel, B. Mishra

Pacific Symposium on Biocomputing 8:116-127(2003)

FOUNDATIONS OF A QUERY AND SIMULATION SYSTEM FOR THE MODELING OF BIOCHEMICAL AND BIOLOGICAL PROCESSES

M. ANTONIOTTI† F. PARK∗ A. POLICRITI+ N. UGEL† B. MISHRA†‡

† *Courant Institute of Mathematical Sciences, NYU, New York, NY, U.S.A.*

∗*Seoul National University, Seoul, S. Korea*

+ *Università di Udine, Udine (UD),* Italy

‡ *Watson School of Biological Sciences, Cold Spring Harbor, NY, U.S.A.*

The analysis of large amounts of data, produced as (numerical) *traces* of *in vivo*, *in vitro* and *in silico* experiments, has become a central activity for many biologists and biochemists. Recent advances in the mathematical modeling and computation of biochemical systems have moreover increased the prominence of *in silico* experiments; such experiments typically involve the simulation of sets of Differential Algebraic Equations (DAE), *e.g.*, Generalized Mass Action systems (GMA) and S-systems [1,2,3]. In this paper we reason about the necessary theoretical and pragmatic foundations for a query and simulation system capable of analyzing large amounts of such trace data. To this end, we propose to combine in a novel way several well-known tools from numerical analysis (approximation theory), temporal logic and verification, and visualization. The result is a preliminary prototype system: simpathica/xssys. When dealing with simulation data simpathica/xssys exploits the special structure of the underlying DAE, and reduces the search space in an efficient way so as to facilitate any queries about the traces. The proposed system is designed to give the user possibility to systematically analyze and simultaneously query different possible timed evolutions of the modeled system.

1 Introduction

The emerging fields of system biology, and its sister field of bioinformatics, focuses on creating a finely detailed and "mechanistic" picture of biology at the cellular level by combining the part-lists (genes, regulatory sequences, other objects from an annotated genome, and known metabolic pathways), with observations of both transcriptional states of a cell (using micro-arrays) and translational states of the cell (using proteomics tools).

After more than a decade of progress, it has become evident that the mathematical foundation of these systems needs to be explored accurately, and that their computational models should be implemented in software packages

faithfully while exploiting the potential trade-offs among usability, accuracy, and scalability dealing with large amounts of data. The work described in this paper is part of a much larger project still in progress, and thus only provides a partial and evolving picture of a new paradigm for computational biology.

We assume the following scenario. Imagine a biologist seeking to test some hypotheses against a corpus of data produced by several *in vitro*, *in vivo*, and *in silico* experiments regarding the behavior of a given biological system, *e.g.*, a regulated metabolic pathway in a given organism. A (graphical) metabolic map of the biochemical system under study, together with a specific associated S-system, is assumed available, and that the number or quantities recorded is large. The biologist can access one or both of the following items:

- Raw data stored somewhere about the temporal evolution of the biological system. This data may have been previously collected by *observing* an in vivo or an in vitro system, or by *simulating* the system in silico.
- Some mathematical model of the biological system[a].

The biologist will want to formulate *queries* about the evolution encoded in the data sets. For example, the biologist may ask: *will the system reach a "steady state"?*, or *will a temporary increase in the level of a certain protein repress the transcription of another?* Clearly the set of numerical *traces* of very complex systems rapidly becomes unwieldy to wade through for increasingly larger numbers of variables.

We first discuss what are the building blocks needed to construct a comprehensive and well-founded system capable of answering such queries. We implemented a prototype system called simpathica/xssys that builds on this foundation. The proposed computational tool derives its expressiveness, flexibility and power by integrating in a novel manner many commonly available tools from numerical analysis, symbolic computation, temporal logic, model-checking, and visualization.

Finally we show how we applied our system to a sizable example: the purine metabolism pathway as described in [3,4,5].

2 Related Works

The modeling and qualitative simulation of complex genetic regulatory networks of large dimension has received considerable attention in the literature,

[a] We note that simulating a system *in silico* actually requires a mathematical model. However, we want to consider the case when such mathematical model is unavailable to both the biologist and the software system.

(see, *e.g.*, [6] and the references cited therein). Most of these works treat situations in which the lack of quantitative information forces simulation in a qualitative way. These qualitative methods differ markedly from the types of qualitative modeling we propose, which are formulated in a *functional approximation* setting; the survey by DeVore [7] in the context of numerical analysis is particularly illuminating. Our approach also differs from systems such as *Cellerator*, [8] a Mathematica package for biological modeling. While this package bears many similarities with our approach, our focus is on a single collection of biochemical reactions rather than a hierarchy. Moreover the temporal ingredient introduced by our integration with a temporal language enhances the reasoning abilities of our proposed tool. Our temporal logic query language also differs from that of [9] (which can be viewed as an extension of the *Qualitative Reasoning* approach of *cf.* [10]) by relying on a full numerical simulation trace or a (sampled) trace of a physical experiment.

The kind of formalization and tools we are proposing in this paper could, in general, be used to study in a more systematic way hypotheses on properties of complex systems of biochemical reactions. The research by Bhalla et al. in [11], for example, aims at proving that a sort of "learned behavior" of biological systems is in fact stored within the mechanisms regulating intracellular biochemical reactions constituting signaling pathways. For this kind of studies, following [12], both qualitative and quantitative features of the system under study should be taken into account.

Finally, the recent control theory literature contains several relevant works on, *e.g.*, the general problem of constructing an automaton from a differential equation models (Brockett [13]), hybrid systems models for biochemical reactions (Alur *et al.* [14]). With respect to the former, our approach takes advantage of the specific mathematical models (S-systems) under consideration, and relates the numerical integration of the S-system with the construction of the automaton. With respect to the latter, the idea of using the discrete part of the (hybrid) automaton to switch from one mode to another when, for example, the number of molecules grows over a certain threshold, can be replicated in our framework by "gluing" different simulations (corresponding to different sets of parameters) in a bottom-up fashion.

3 Mathematical and Computational Models

3.1 Canonical Forms of Biological Systems Models

In the following we build on ideas introduced in [1,2,3] (from which we will borrow also most of the notation) and, e.g., [15]. Central to our discussions

will be the notion of S-system (or of a GMA system). The basic ingredients of an S-system are n dependent variables to be denoted $X_1, \ldots, X_n$, m independent variables $X_{n+1}, \ldots, X_m$, each of which has domain $D_1, \ldots, D_{n+m}$, respectively. We augment the S-system form with a set of *algebraic constraints* that serve to characterize the conditions under which a given set of equations is derived from a set of maps. The justification for this construction is beyond the scope of this paper and it appears elsewhere. The basic differential equations constituting the system then take the following *power law* form:

$$\dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}} \tag{1}$$

$$C_j(X_1(t), \ldots, X_m(t)) = \sum \left(\gamma_j \prod_{k=1}^{n+m} X_k^{f_{jk}} \right) = 0 \tag{2}$$

where the α_i's and β_i's are called *rate constants* and govern the positive or negative contributions to a given substance (represented by X_i as a function of time) with other variables entering in the differential equation with exponents to be denoted as g_{ij}'s and h_{ij}'s. The γ_j are called *rate constraints* acting concurrently with the exponents f_{jk} to delimit the evolution of the system over a specified manifold embedded in the $n+m$-dimensional surface. Note that we have $\alpha_i \geq 0$ and $\beta_i \geq 0$ for all i's. S-systems can be integrated numerically, and symbolically in certain special cases (we address the symbolic case in a much more general way in a future work—see [16] for a preliminary treatment of the topic). The particular S-system "canonical" form allows for very efficient computations of both the function flows X_i and their derivative fields $\dot{X}_i$ [17].

3.2 XS-Systems: S-Systems Extended with Automata

The following properties of biochemical metabolic systems and corresponding S-systems serve as our starting point:

- The value of the dependent and independent variables uniquely characterize the state of the system when *normalized* with respect to time (and possibly other values);
- The transitions from one state to another are not necessarily encoded in the metabolic map of the system, and are instead *parametric* with respect to the value of constants in the S-system;

- There exists an *exogenous* set of "functions" that represent special perturbations of the system that the user may decide to include in the trace generation, *e.g.*, *ramps*, *impulses*, and *oscillations* (possibly with parametric amplitude and frequency).

We start with snapshots of the system variables' values constituting the possible states of the automaton. Transitions will be inferred from *traces* of the system variables' values evolution.

Definition 1 *Given an S-system $\mathcal{S}$, the* S-system automaton $\mathcal{A}_\mathcal{S}$ *associated to $\mathcal{S}$ is 4-tuple $\mathcal{A}_\mathcal{S} = (S, \Delta, S_0, F)$, where $S \subseteq D_1 \times \cdots \times D_{n+m}$ is a (finite or infinite) set of states, $\Delta \subseteq S \times S$ is the transition relation, and $S_0, F \subset S$ are the initial and final states, respectively.*

Definition 2 *A* trace *of an* S-system automaton $\mathcal{A}_\mathcal{S}$ *is a (finite or infinite) sequence $s_0, s_1, \ldots, s_n, \ldots$, such that $s_0 \in S_0$, $\Delta(s_i, s_{i+1})$ for $i \geq 0$. Equivalently, a trace can be defined as:*

$$\mathsf{trace}(\mathcal{A}_\mathcal{S}) = \langle \langle X_1(t) \ldots X_n(t) \rangle \mid t \in \{t_0 + k\,\mathbf{step} : \ \ k \in \mathbb{N}\} \rangle.$$

Notice that, for fixed values of the independent variables, a unique trace is obtained when a simulation is performed. Moreover, while studying a trace of a system, it can be useful to focus on one or more specific variables, which justifies the following definition:

Definition 3 *Given any set of variables $U \subseteq \{X_1, \ldots, X_{n+m}\}$, the sequence:*

$$\mathsf{trace}(\mathcal{A}_\mathcal{S}|_U) = \langle \langle X_i(t) \mid X_i \in U \rangle \ : \ t \in \{t_0 + k\,\mathbf{step} : \ \ k \in \mathbb{N}\} \rangle,$$

is called the trace of U. *If U consists of a single variable X_i the trace is called the* trace of X_i.

Multiple traces arise as we start varying the values in the primary parameter sets. Collection of such traces, in general, allows one to study the different instances of the simulated metabolic pathway evolution. Such a collection will give rise to an automaton with corresponding transitions once a suitable *equivalence relation* on states has been defined.

3.3 Construction of a Collapsed Automaton

Given a trace $\mathsf{trace}(\mathcal{A}_\mathcal{S})$ we want to construct an automata that can be used in conjunction with a TL based query system. The construction we present in the following corresponds to finding a partition of the real line (time) into a set I of non-overlapping intervals where the functions $X_i(t)$ are approximated in a piecewise-linear fashion. This is a very simple case of approximation as defined in much greater detail in [7]. Here we show a very simple automaton construction method that essentially is an *adaptive approximation* for the

given $\vec{X}(t)$; more sophisticated construction algorithms can be constructed based on, *e.g.*, adaptive step-size integration methods for ordinary differential equations, and other types of segmentation algorithms. We will describe an idea about how to improve the construction of such *collapsed automata* when there is more information available about the kind of *in silico* simulation experiment. Essentially we will use a "dictionary" of base functions to guide the collapsing method in a more adaptive way. We will present a fully developed treatment of this approach in a subsequent paper.

A *linear* automaton can be constructed simply by associating a different state to each tuple $\langle X_1(t) \dots X_n(t) \rangle$ as time is incremented according to a given fixed time step. To illustrate, consider the function $X_i(t)$ at times $t_i, t_{i+1}, \dots t_{i+5}$ as depicted in Figure 1. In this case we have $\mathbf{step} = t_{i+1} - t_i$ as a result of (fixed) sampling or numerical integration. We associate the automata $\mathcal{A}_{\mathcal{S}}$ to the trace of X_i simply by taking into account each time step.

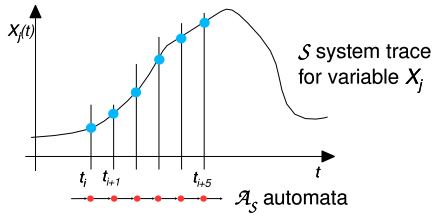

Figure 1. Simple one-to-one construction of the "trace" automata $\mathcal{A}_{\mathcal{S}}$ for a S-system $\mathcal{S}$.

We now suggest a simple method for collapsing the states of a linear automaton into piecewise-linear equivalence classes. Given a numerical solution for our S-system, two states $X_i(t + k\,\mathbf{step})$ and $X_i(t + (k+j)\,\mathbf{step})$, $j > 0$, are said to be equivalent under the relation R_{i,δ_i} if

$$\mid \dot{X}_i(t + k\,\mathbf{step}) - \dot{X}_i(t + (k-1)\,\mathbf{step}) \mid \leq \delta_i.$$

The above construction is easily extended to the full collection of variables (see the following definition) and allows us to consider equivalent two states if the difference between the rate of growth of the corresponding variables is sufficiently small (with respect to the parameter δ_i).

Definition 4 *The relation $R_{\vec{\delta}}$ holds between two states $s_k = \vec{X}(t + k\,\mathbf{step})$ and $s_{k+j} = \vec{X}(t + (k+j)\,\mathbf{step}))$ with $j > 0$, if and only if, for each $i \in$*

$\{1, \ldots, n+m\}$,

$$|\dot{X}_i(t + k\,\mathbf{step}) - \dot{X}_i(t + (k+j)\,\mathbf{step})| \leq \delta_i.$$

The collection $\{\delta_i \mid 1 \leq i \leq n+m\}$ *is denoted by* $\vec{\delta}$.

The above relation turns out to be an equivalence relation collapsing states whose variables' values are sufficiently similar, *if* computed *over* the numerical approximation of the solution. The (simple) idea is to choose as representative in each equivalence class, the element corresponding to the minimum time in the class. It is easy to build an iterative algorithm based of the above relation that determines the *knot* points in the state space that can be used as *state set representatives* in the collapsed automata. The algorithm is presented in [18] and has $O(hk)$ time complexity, where h is the size of the trace and k is the number of variables in the system.

In the following, given a collection of linear automata (traces), we will propose to "glue" them together in a unique automaton capable of modeling various possible behaviors of the system.

Example 5 *Consider again the function* $X_i(t)$ *depicted in Figure 1. In Figure 2 the effects of applying the collapsing algorithm are shown. With respect to* $X_i(t)$ *we obtain an automata* $\mathcal{A}_{\mathcal{S}i}$ *which has the reduced number of states*

$$\mathsf{states}(\mathcal{A}_{\mathcal{S}i}) = \langle \ldots (t_i, X_j(t_i), \dot{X}_j(t_i)),\ \ (t_{i+2}, X_j(t_{i+2}), \dot{X}_j(t_{i+2})),$$
$$(t_{i+5}, X_j(t_{i+5}), \dot{X}_j(t_{i+5})), \ldots \rangle$$

Now suppose we have a different function $X_k(t)$. *We associate to* $X_k(t)$ *the collapsed automata* $\mathcal{A}_{\mathcal{S}k}$, *such that*

$$\mathsf{states}(\mathcal{A}_{\mathcal{S}k}) = \langle \ldots (t_i, X_k(t_i), \dot{X}_k(t_i)),\ \ (t_{i+4}, X_k(t_{i+4}), \dot{X}_k(t_{i+4})), \ldots \rangle$$

i.e. the "landmark" times are t_i *and* t_{i+4} *in this case. In order to construct a useful automata for the analysis tool we construct the* merged *automata* $\mathcal{A}_{\mathcal{S}jk}$ *such that*

$$\begin{aligned} \mathsf{states}(\mathcal{A}_{\mathcal{S}k}) = \langle \ldots\ & (t_i, X_j(t_i), \dot{X}_j(t_i)), \\ & (t_{i+2}, X_j(t_{i+2}), \dot{X}_j(t_{i+2})), \\ & (t_{i+4}, X_k(t_{i+4}), \dot{X}_k(t_{i+4})), \\ & (t_{i+5}, X_j(t_{i+5}), \dot{X}_j(t_{i+5})), \ldots \rangle \end{aligned}$$

i.e. automata $\mathcal{A}_{\mathcal{S}jk}$ *is an* ordered merge *of the two automata* $\mathcal{A}_{\mathcal{S}j}$, $\mathcal{A}_{\mathcal{S}k}$.

3.3.1 Normalization and Projection

In order to capture the notion of state-equivalence *modulo* normalization we begin with the following definition:

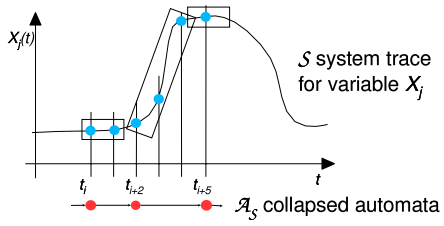

Figure 2. The effects of the *collapsing* construction of the "trace" automata $\mathcal{A}_\mathcal{S}$ for a S-system $\mathcal{S}$.

Definition 6 *Given a subset V of the set $\{X_1, \ldots, X_{n+m}\}$ of variables, we define the set of states* normalized *with respect to V as the following set of tuples:*

$$S_{\setminus V} = \left\{ \left\langle \frac{X_i(t_0 + k\,\mathbf{step})}{\nu_i(V, t_0 + k\,\mathbf{step})} \right\rangle \;\; : \;\; k \in \mathbb{N} \right\},$$

with the ν_i's are normalizing functions.

More complex forms of normalization can be obtained when other variable contribute into play. Moreover, notice that when we normalize with a collection of normalizing functions defined as follows:

$$\nu_i(U, t) = \begin{cases} X_i & \text{if } X_i \notin U, \\ 1 & \text{otherwise,} \end{cases}$$

then the normalization corresponds to projecting with respect to the set of variables U.

3.3.2 Adaptive Collapsing by Dictionary Functions

Suppose we know that the user is trying to see what is the response of a simulated system to an impulse (*cf.* the example in Section 4.1). This is a piece of information that we can use to build a collapsed automaton more effectively. We assume that our tool comes equipped with a dictionary **D** of predefined basic functions: *ramps*, *impulses*, *sigmoids*, etc. Suppose that the user specifies that the simulation will include an impulse $I(t; \tau)$ at time τ. The collapsing algorithm will then use this information to compute the partition from time τ to $\tau + \Delta\tau$ using the function $I(t; \tau)$ as a component of the approximation of the solution.

This is just a special form of an approximation scheme. The computational complexity will vary according to the set of functions in **D**, e.g. it is known that when using *wavelets* as components, finding an *optimal* approximation to a function using a dictionary of wave-forms is an NP-hard problem [19]. We conjecture that, since we are willing to loose information in the collapsing operation, we will be able to use a greedy adaptive approach as well that will keep the computational complexity low. A full treatment of this topic will appear in a forthcoming paper.

4 Pathways Simulation Query language

In this section we briefly outline a language that can be used to inspect and formulate queries on the simulation results of XS-systems.

Our aim is to provide the biologists with a tool to formulate various queries against a repository of *simulation traces* and a set of *known pathways specifications*. We propose a *Temporal Logic* language (*cf.* [20]) with a specialized set of predicate variables whose aim is to make it easy to formulate queries on numerical quantities. For instance, all our discussions can be carried out in the standard CTL, Computation Tree Logic: a branching-time propositional temporal logic.

We can also augment the standard CTL language with a set of *domain dependent* queries. Such queries may be implemented in a more efficient way and express typical questions of interest to biologists.

4.1 An Example: Purine Metabolism

We now revisit in detail the example of purine metabolism described in [3] Chapter 10 and fully analyzed in [4,5]. The pathway for purine metabolism is presented in Figure 3. A brief description of the key reactions follows, and the reader is invited to examine the more detailed summaries contained in the literature referenced in [3,4,5].

The main metabolite in purine biosynthesis is *5-phosphoribosyl-α-1-pyrophosphate* (PRPP). A linear cascade of reactions converts PRPP into *inosine monophosphate* (IMP). IMP is the central branch point of the purine metabolism pathway. IMP is transformed into AMP and GMP. Guanosine, adenosine and their derivatives are recycled (unless used elsewhere) into *hypoxanthine* (HX) and *xanthine* (XA). XA is finally oxidized into *uric acid* (UA). In addition to these processes, there appear to be two "salvage" pathways that serve to maintain IMP level and thus of *adenosine* and *guanosine* levels as well. In these pathways, *adenine phosphoribosyltransferase* (APRT)

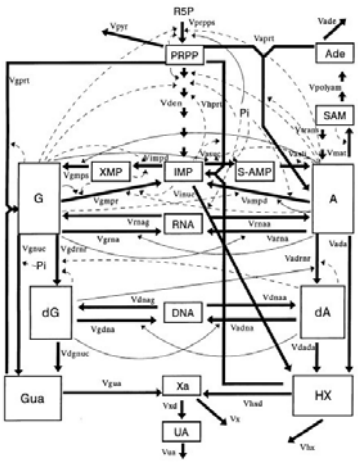

Figure 3. The metabolic scheme of purine metabolism in human. (Reprinted from [5], where a full description and further references may be found.)

and *hypoxanthine-guanine phosphoribosyltransferase* (HGPRT) combine with PRPP to form ribonucleotides.

The consequences of a malfunctioning purine metabolism pathway are severe and can lead to death. The entire pathway is quite complex and contains several feedback loops, cross-activations and reversible reactions, and thus an ideal candidate for reasoning with the computational tools we have developed.

In [3], a sequence of models for purine metabolism is presented alongside an analysis of how to identify discrepancies with physically observed data, and how to amend the current model in order to explain these discrepancies.

We show how to formulate queries over the simulation traces to express various desirable properties (or absence of undesirable ones) that the model should possess. Should any of these queries "fail," the model will be marked for further examination, experimentation and correction.

As an example consider the "Final" model for purine metabolism presented in [3]. The in silico experiment shows that when an initial level of PRPP is increased by 50-fold, the steady state concentration is quickly absorbed by the system. The level of PRPP returns rather quickly to the expected steady state values. IMP concentration level also rises and HX level falls before returning to predicted steady state values.

Suppose that we wanted to ask the system how it will respond to a temporary (instantaneous) increase in the level of PRPP. Such request can be formulated as follows:

```
always(PRPP > 50 * PRPP1
       implies
         (steady_state()
          and eventually(IMP > IMP1)
          and eventually(HX < HX1)
          and eventually(always(IMP == IMP1))
          and eventually(always(HX == HX1))
```

an (instantaneous) increase in the level of PRPP will not make the system stray from the predicted steady state, even if temporary variations of IMP and HX are allowed. A detailed discussion in the full paper shows how `xssys` responds to the query and helps the biologist to acquire the needed intuition.

5 Concluding Remarks

We have presented a novel framework under which we bring together well known tools from numerical analysis (approximation theory), temporal logic and verification, and visualization. This is a work in progress whose final aim is to construct an effective tool to aid biologists analyze experimental results and design new ones[b].

There are several open questions in our work that we need to address. We are now considering how to extend our automata construction by using notions from *approximation theory* that will allow us to take into consideration time and frequency domain aspects of a trace. Of course more theoretical treatments of the subject are possible as well (*cf.* [13]).

References

1. M. A. Savageau. *Biochemical System Analysis: A Study of Function and Design in Molecular Biology.* Addison-Wesley, 1976.
2. E. O. Voit. *Canonical Nonlinear Modeling, S-system Approach to Understanding Complexity.* Van Nostrand Reinhold, New York, 1991.
3. E. O. Voit. *Computational Analysis of Biochemical Systems A Practical Guide for Biochemists and Molecular Biologists.* Cambridge University Press, 2000.
4. R. Curto, E. O. Voit, A. Sorribas, and M. Cascante. Mathematical models of purine metabolism in man. *Mathematical Biosciences*, 151:1–49, 1998.

[b] A more complete version of this paper is available in our web site `bioinformatics.cims.nyu.edu`.

5. R. Curto, E. O. Voit, A. Sorribas, and M. Cascante. Analysis of abnormalities in purine metabolism leading to gout and to neurological dysfunctions in man. *Biochemical Journal*, 329:477–487, 1998.
6. H. de-Jong, M. Page, C. Hernandez, and J. Geiselmann. Qualitative simulation of genetic regulatory networks: methods and applications. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Art. Int.*, San Mateo, CA, 2001. Morgan Kaufmann.
7. R. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.
8. B. E. Shapiro and E. D. Mjolsness. Developmental simulation with cellerator. In *Proc. of the Second International Conference on Systems Biology (ICSB)*, Pasadena, CA, November 2001.
9. B. Shults and B. J. Kuipers. Proving properties of continuous systmes: qualitative simulation and temporal logic. *Artificial Intelligence Journal*, 92(1-2), 1997.
10. B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.
11. U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *SCIENCE*, 283:381–387, 15 January 1999.
12. D. Endy and R. Brent. Modeling cellular behavior. *Nature*, 409(18):391–395, January 2001.
13. R. W. Brockett. Dynamical systems and their associated automata. In U. Helmke, R. Mennicken, and J. Saurer, editors, *Systems and Networks: Mathematical Theory and Applications—Proceedings of the 1993 MTNS*, volume 77, pages 49–69, Berlin, 1994. Akademie-Verlag.
14. R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biological systems. In *Proc. of the Fourth International Workshop on Hybrid Systems: Computation and Control*, LNCS 2034, pages 19–32, Berlin, 2001. Springer-Verlag.
15. A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics*. Portland Press, London, second revised edition, 1999.
16. B. Mishra. A symbolic approach to modeling cellular behavior. In *Proceedings of HiPC 2002, Bangalore,* India, December 2002.
17. D. H. Irvine and M. A. Savageau. Efficient solution of nonlinear ordinary differential equations expressed in S-System canonical form. *SIAM Journal on Numerical Analysis*, 27(3):704–735, 1990.
18. M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. XS-systems: extended S-Systems and Algebraic Differential Automata for Modeling Cellular Behaviour. In *Proceedings of HiPC 2002, Bangalore,* India, December 2002.
19. G. Davis, S. Mallat, and M. Avellaneda. Adaptive Greedy Approximations. *Constructive Approximation*, 13:57–98, 1997.
20. E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. MIT Press, 1990.