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.

Abstract

The analysis of large amounts of data has become the main activity of many biologists and biochemists.

Large data sets are being produced as (numerical) *traces* of *in vivo*, *in vitro* and *in silico* experiments.

In the last case such traces are the result of *simulating* mathematical models of biochemical systems. Such models usually consist of relatively simple sets of differential algebraic equations (DAE) alongside with their numerical solutions. E.g. the well known *Generalized Mass Action* systems (GMA) and S-systems [1, 2, 3].

In this paper we reason about what are the necessary theoretical and pragmatic foundations for a query and simulation system capable of analyzing large amounts of 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 prototypical combined system: simpathica/xssys. When dealing with simulation data simpathica/xssys takes advantage of the particular form of the DAE used for modeling, and is capable of reducing the search space where the queries are to be evaluated.

1 Introduction

Recent advances in genomics have made it possible for the first time for a biologist to access enormous amounts of information for a number of organisms, including human, mouse, arabidopsis, fruit fly, yeast and E. coli. These developments are at the heart of the many renewed ambitious attempts by biologists to understand the functional roles of a group of genes using powerful computational models and high throughput microbiological protocols. 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 transcriptional states of a cell (using micro-arrays) and translational states of the cell (using proteomics tools).

In the process, it has become evident that the mathematical foundation of these systems needs to be explored accurately and that their computational models 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 looking 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). Let's also make the assumption that the number or quantities recorded is large. The biologist can access one or both 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[1].

The biologist will want to formulate *queries* about the evolution encoded in the data sets. E.g the biologist may ask: *will the system reach a "steady state"?* Or: *will a temporary increase in the level of a certain protein represses the transcription of another?*

We will discuss what are the building blocks needed to construct a comprehensive and well-founded system capable to answer such queries.

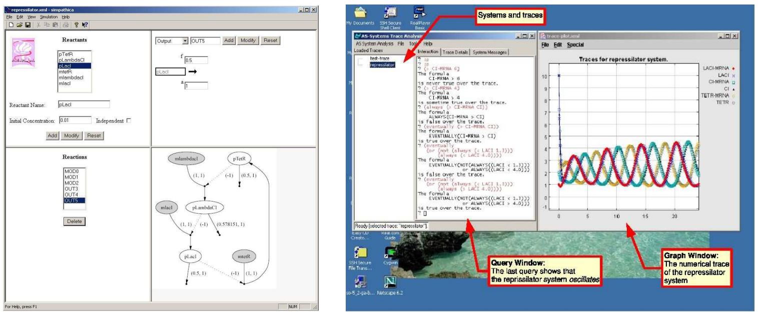
1.1 Methods and Tools

Several biological and biochemical mechanisms can be modeled with relatively simple sets of differential algebraic equations (DAE). The numerical solution to these differential equations provides a potentially powerful and effective investigative tool for biologists and biochemists. In the following, we demonstrate the power of a novel computational tool with the ability to query massive sets of numerical data obtained from in silico experiments on complex biological systems; the 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.

We start by considering the the *S-systems* models of biochemical processes developed mostly by Savageau and Voit [1, 2, 3], and "extend" them in order to facilitate their analysis. We name such extension *XS-system*. The main innovation provided by an XS-system consists in an automaton-based semantics of the temporal evolution of complex biochemical reactions presented as a set of DAE. Such automaton-based semantics has the advantage to reduce the search space over which the truth or falsity of a given query must be established.

The automaton underlying an XS-system can be constructed. in several ways. One of the aims of this paper is to summarize the mathematical tools and frameworks we think most promising for such construction, always keeping in sight the use scenario we described. Essentially we will propose a construction based on an *approximation* of a numerical trace (*cf.* [4] and the references contained therein).

[1] 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.



(a) (b)

Figure 1: (a) The simpathica Main Window. The system being analyzed is the "repressilator" system [5]. The upper left frame contains a list of the reactants. The upper right frame is used to insert different kinds of reactions. The lower left frame contains a list of known reactions. Finally the lower left frame contains a depiction of the reactions' network. (b) The main xssys view. The xssys interface proper is on the left. The left frame contains a list of the "loaded" traces (obtained from simulators like the simpathica/Octave engine or PLAS [3]). The right frame is used to type in various temporal logic queries and to check the results. The right window is simply a plotting application (PtPlot from UC Berkeley) which we use for visualization purposes.

In the following, we will briefly describe the prototype implementation underlying the xssys system and we will discuss the biological applications illustrating how we envision simpathica and xssys to be used in practice (*cf.* Figure 1 to see screen-shots of the two systems).

[**Missing:** The rest of the paper is organized as follows....]

2 Mathematical and Computational Models

We stated that several biological systems may be modeled with systems of DAE. The simple (canonical) forms of the differential equations may contrast with the actual "complexity" of the system being modeled, as measured by the number of *variables* involved. The set of numerical *traces* of very complex systems rapidly becomes unwieldy to wade through when several variables are involved.

Starting from an S-system formulation (as described in [1, 2, 3]), we want to construct an automaton *summarizing* the phases along which the simulated biochemical system passes during its evolution in time. The automata we are proposing will allow the user to view and manipulate a, detectably specified, representation of the set of states through which the system evolves. The approach will mainly serve the following two purposes:

- Explicitly render the significant changes in the values of substances involved in the biochemical reactions during the *in silico* experiment, thereby providing a better control on the physical plausibility of the latter.

- Provide a precise language for controlling sets of possible experiments, based on different values of the parameters involved.

Our basic assumption is that a (graphical) metabolic map of the biochemical system under study, together with a specific associated S-system, is available. The automaton construction is then based on the computation of the approximate numerical solution of the S-system and is performed in two steps: first, starting from any given *time step* (which will be one of the crucial parameters in our construction) and from the corresponding approximate numerical solution for the power-law differential equations, a synchronous automaton whose states are collection of values for the dependent values is determined. Then, a qualitative analysis (based on the first derivatives $\dot{X}_i$'s of the functions expressing concentrations) is carried out to the effect of collapsing states modeling non-significant variations in the evolution of the system.

The second operation simply corresponds to finding a set of "linear approximations" of the function flow as represented in the computed or sampled trace. We note that this fact may find a correspondence in some of the work done in the analysis of Hybrid Systems (*cf.* [6]), which we will explore in a future work.

The automaton obtained after the second of the above two steps is "asynchronous" (untimed, *cf.* [6]), as the values of the temporal intervals connecting collapsed states are ignored, but suitable for a verification analysis of temporal properties [7]. To this end a (temporal) language suited for such a kind of analysis is proposed and studied. Moreover, the equivalence relation collapsing states at different stages of the temporal evolution can be either global (unique) or local, that is itself a function of time, providing the ability of refining the qualitative analysis in sensitive regions of the metabolic pathway[2].

From a general point of view, the main features of the approach we described for designing tools to model biochemical systems are the following:

1. The possibility of both quantitative and qualitative modeling, within one integrated environment;

2. A central rôle of time;

3. The possibility of interacting with the tool for designing new experiments.

In particular, with respect to the first of the above points, the qualitative modeling of the system is supposed to be specified *after* a quantitative description has been determined and (numerically) solved. The automaton is in fact obtained by "gluing" together different representations of possible evolution of the system. This sort of *bottom-up* automata construction is one of the characterizing

[2]Tools based on temporal specification and verification are widely recognized as crucial in the realm of embedded reactive systems, and the approach we present here is easily seen to bear some similarity with ideas very much exploited in that area. Modeling the evolution of a biochemical system as we are proposing consists, in fact, in seeing the system's state-sequence as the analogue of a computation, and different computations as (simulations of) experiments differing for some given parameters' values. However, a special feature of the biochemical-systems' modeling arena is its stronger focus on one (or a restricted family of) experiment(s). This must be contrasted with the case of formal verification of temporal properties of embedded systems, in which the focus must stand much more on the careful consideration of *all* possible computations (in order to be able exclude the bad one!).

In fact, our automata construction can produce different values as the parameters involved (i.e. time-step, rate constants, exponents, etc.) are varied at the start. For example, as a limit to the choice of a smaller time-step in the construction, we could consider the analytic description of the solution to the differential equation governing the system. The, more abstract, study of the issues related with families of possible numerical solutions (and hence automaton constructions), is going to be our next goal.

aspects of our proposal, the other being the idea that the notion of state of the system should be less restrictive than the one usually employed in formal verification and strictly based on variables' values (see Section 3.2).

Notice, finally, that the proposal we are putting forward could be discussed in the context of regulatory pathways modeling as well. We will discuss the details of this application in a future work.

2.1 Canonical Forms of Biological Systems Models

In the following we will build on ideas introduced in [1, 2, 3] (from which we will borrow also most of the notation) and, e.g., [8]. Central to our discussions will be the notion of S-system (or of a GMA system) that we briefly introduce below together with some definition that will be used throughout the paper.

The basic ingredients of an S-system are n dependent variables to be denoted $X_1, \ldots, X_n$, m independent variables $X_{n+1}, \ldots, X_m$ and let $D_1, \ldots, D_{n+m}$ be the domains where the $n+m$ variables take value. We augment the S-system form with a set of *algebraic constraints* which 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 take the general form:

$$\dot{X}_i(t) = V_i^+(X_1(t), \ldots, X_m(t)) - V_i^-(X_1(t), \ldots, X_m(t)), \tag{1}$$

for each dependent variable X_i (see [3] for a complete discussion and justification of the assumptions underlying the format of the above equation). The set of algebraic constraints take the form

$$\{C_j(X_1(t), \ldots, X_m(t)) = 0\} \tag{2}$$

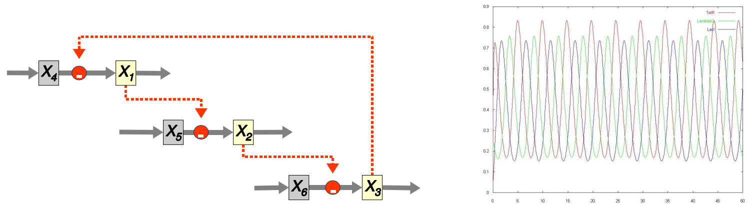
The above equations take, in general, the following *power law* form:

$$\dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}} \tag{3}$$

$$C_j(X_1(t), \ldots, X_m(t)) = \sum \left(\gamma_j \prod_{k=1}^{n+m} X_k^{f_{jk}} \right) = 0 \tag{4}$$

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.

A system of differential equations (power-laws) such as the one above can be integrated by either symbolically – in particularly favorable cases – or by numerical approximation. The particular S-system "canonical" form allows for very efficient computations of both the function flows X_i and the derivative field $\dot{X}_i$ [9].



(a) (b)

Figure 2: The repressilator system metabolic map in "cascade" form and its oscillatory trace [5].

In the following we will concentrate on the "numerical" case and we will also exploit the special nature of the S-system traces for our Temporal Logic"query language". We will address the "symbolic" case in a much more general way in a future work (see [10] for a preliminary treatment of the topic). When a numerical approximation is involved, the notion of *time-step* "**step**" becomes central to our considerations[3].

From now on we will denote by $\mathcal{S}$ (to be used also as index for parameters) a specific S-system given as both a set of differential equations of the above kind as well as a *metabolic map*, which is a pictorial counterpart of the system especially useful as a compact representation for qualitative study.

Example 1 *Consider the following example consisting of the pathway of a so-called* repressilator system *[5]. The repressilator system metabolic map involves six variables $X_1, \ldots, X_6$, the first three of which (namely X_1 ,X_2 and ,X_3) are independent while the remaining are dependent.*

The following is the S-system corresponding to the above metabolic map:

$$\begin{cases} \dot{X}_1 &= \alpha_1 X_3^{g_{13}} X_4^{g_{14}} - \beta_1 X_1^{h_{11}}; \\ \dot{X}_2 &= \alpha_2 X_1^{g_{21}} X_5^{g_{25}} - \beta_2 X_2^{h_{22}}; \\ \dot{X}_3 &= \alpha_3 X_2^{g_{32}} X_6^{g_{36}} - \beta_3 X_3^{h_{33}}, \end{cases}$$

with X_4, X_5, and X_6 as controls (independent variables).
*Figure 2(b) shows the oscillatory trace of the system (*cf. *[5] for a discussion of the numerical and analytical analysis of the system).*

2.2 Related works

Many interesting researches are dealing with more or less the general themes treated in this work.

For example, in [11] the problem of modeling and simulating qualitatively complex genetic regulatory networks of large dimension is studied. That work, as well as other along the same line, aims at dealing with situations in which the lack of quantitative information forces simulation in

[3]We are particularly interested in the *fixed* time step case since it will allow us to treat data produced by sampling of in-vitro and in-vivo experiments. Data produced numerically by *variable step* integrators obviously simplify the automata construction since, they already factor in their computation the *smoothness* of the curve.

a qualitative way, an assumption that marks the difference with the situation we study here and with kind of qualitative modeling we propose.

The problem of constructing an automaton from a given mathematical model of a complex system has also been previously considered in the literature. In particular, in the control literature, it has been deeply studied by Brockett in [13]. Our approach here is certainly at a lower level of generality as it deals with specific mathematical models (S-systems) and, moreover, tries to integrate the numerical determination of a solution for the system of differential equations involved with the automata construction.

In the Numerical Analysis literature the problem can be recast as a *function approximation* one. The survey by DeVore [4] contains a wealth of information about Approximation Theory that we intend to reuse for our tool.

The kind of formalization and tools we are proposing in this paper could, in general, be used to study on a more systematic way hypothesis on properties of complex systems of biochemical reactions. The research by Bhalla et al. in [14], for example, aims at proving that a sort of "learned behaviour" of biological systems is in fact stored within the mechanisms regulating intracellular biochemical reactions constituting signaling pathways. For this kind of studies, following [15], both qualitative and quantitative features of the system under study should be taken into account and we hope the system we propose can turn out useful on the ground of its ability to capture and compare temporal evolution of the system under study.

In [16] Cellerator is presented, a Mathematica package for biological modeling that bears many similarities with our project here.

While our aim is more focused, as we study a single collection (not an hierarchy) of biochemical reactions, the temporal ingredient introduced by our integration with a temporal language should enhance the reasoning abilities of our proposed tool.

Using a Temporal Logic query language to analyze continuous systems is investigated also in [17], as an extension of the *Qualitative Reasoning* approach (*cf.* [18]). We differ from this interesting approach inasmuch as we rely on a full numerical simulation trace or a (sampled) trace of a physical experiment.

Finally, in [19] is reported a use of Hybrid Systems in modeling properties of systems of biochemical reactions. It is very interesting the idea of using the discrete component of the (hybrid) automaton to switch from one mode to another when (for example) the number of molecules grows over a certain threshold. In our framework the same effect should be captured in a sort of bottom-up fashion, by "gluing" different simulations determined with different sets of parameters.

3 XS-systems: S-systems extended with Automata

In this section we describe the general idea underlying the automata construction. Our starting point are the following property of biochemical metabolic systems and corresponding S-systems:

- 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 the other are not necessarily encoded in the metabolic map of the system and are *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).

The idea behind the automata definition and construction we are going to define is to start with snapshots of the system variables' values that will constitute the possible states of the automaton. Transitions will be inferred from *traces* of the system variables' values evolution.

On the ground of the above observations we define:

Definition 2 *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.*

Final states will be those states in which the simulation reaches a recognizable end and are supposed to represent "equilibrium" points for the pathway being modeled.

Definition 3 *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 $i \geq 0$. A trace can also be defined as:*

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

is called the trace of $\mathcal{A}_{\mathcal{S}}$

Clearly, from now on, if the analysis of the system is to be carried out in a finite interval of time $[0, t]$, a k such as the one in the above definition of trace will vary in $\{1, \ldots \lceil \frac{t}{\mathbf{step}} \rceil\}$.

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 concentrate (i.e. *project*) on one or more variables, which justifies the following definition:

Definition 4 *Given any set of variables $U \subseteq \{X_1, \ldots, X_{n+m}\}$, the sequence:*

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

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 the automaton with corresponding transitions when a suitable *equivalence relation* on states is defined.

3.1 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 by a straight line. This is a very simple case of approximation as defined in much greater detail in [4]. In particular, we show a very simple automata construction algorithm that essentially is an *adaptive approximation* one for the given $\vec{X}(t)$.

In a future work we will explore the use of a more sophisticated basis for our automata construction. In particular we will explore *wavelet* basis and *Volterra* series expansions to recover better information about time-localized impulses and frequencies of periodical phenomena.

Automata collapsing by piecewise linear segments. We can easily construct an (in general unreduced) automaton $\mathcal{A}_{\mathcal{S}}$ by simply associating a different state to each tuple $\langle X_1(t) \ldots X_n(t) \rangle$ as time grows according to a given time step. In this case there is a unique trace for the obtained automaton that is therefore called *linear* automaton.

Example 5 *Consider the function $X_i(t)$ at times $t_i, t_{i+1}, \ldots t_{i+5}$ as depicted in Figure 3. In this case we have* **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. Note that this is not much different than what it is done by "untiming" a Timed Automata.*

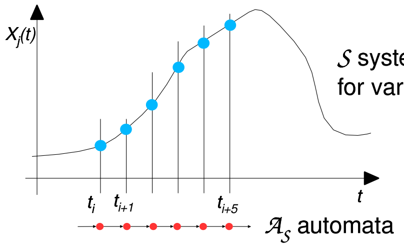


Figure 3: Simple one-to-one construction of the "trace" automata $\mathcal{A}_{\mathcal{S}}$ for a S-system $\mathcal{S}$.

The automata constructed as above are in some way "synchronous", in the sense that the time elapsed while passing from one state to the other is known and fixed (with respect to the numerical trace obtained from a source sampled at "fixed" intervals or from an integration algorithm employing a "fixed" step size).

If the trace is the result of a variable step size integrator, then the automata constructed in this way will already summarize the system behavior in a set of "smooth" regions.

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. However, before that, we propose a method to collapse states of a linear automaton into equivalence classes capturing *qualitatively* the behavior of the system (along a single trace).

A solution to a given S-system is, in general, determined by a numerical approximation once the values for the independent (constant) variables are given. Given a numerical approximation to a solution of our S-system, for any given time step and any given (dependent) variable X_i, a linear automaton corresponding to the trace $\mathsf{trace}(X_i)$ could be reduced by using the following equivalence relation R_{i,δ_i}, which depends on the parameter δ_i: R_{δ_i} holds between two states $X_i(t + k\,\mathbf{step})$ and $X_i(t + (k+j)\,\mathbf{step})$ for $j > 0$, if and only if

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

Notice that the first derivatives of functions expressing the variations of dependent variables, involved in establishing the validity of the above condition, are available during the numerical computation carried out to solve the underlying S-system. Moreover, notice that if the above collapsing is performed on an independent variable (assumed to be constant), the construction trivializes and a unique state is obtained.

The above construction is extended to the full collection of variables as follows:

Definition 6 *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 $i \in \{1, \ldots, n + m\}$,*

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

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 if computed *during* the determination of the numerical approximation to the solution. The (simple) idea is to choose as representative in each equivalence class, the element corresponding to the minimum time in the class. The following pseudo-algorithm explains how we compute the set of states of the collapsed automata:

Algorithm 1 COLLAPSE_STATES_INCREMENTALLY($\vec{\delta}$, **step** , t)

```
let s_0 = ⟨X_1(t_0) ... X_n(t_0)⟩ ∈ S_0;
STATE := 0;                          –initialize the current state
REPR := 0;                           –initialize the representative of the equivalence class
while STATE < ⌈t / step⌉ do
  STATE := STATE +1;
  ...                                –simulation proceeds using e.g. Euler's method
  if ∃δ_i ∈ δ⃗ (|Ẋ_i(t_0 + REPR step) − Ẋ_i(t_0 + STATE step)| ≥ δ_i) then
    REPR := STATE;
  end if
end while
```

If the guard of the **if**-statement in the above algorithm is weakened (e.g.) restricting the set of variables for which the condition is checked, the effect will be to "concentrate" on the restricted set of variables and, in general, to producing *less* states.

Example 7 *Consider again the function $X_i(t)$ of* cf. *Example 5. In Figure 4 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 fewer 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 to 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 $\mathcal{A}_{\mathcal{S}\,j}$, $\mathcal{A}_{\mathcal{S}\,k}$.*

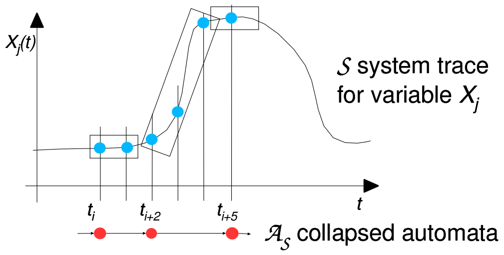


Figure 4: The effects of the *collapsing* construction of the "trace" automata $\mathcal{A}_{\mathcal{S}}$ for a S-system $\mathcal{S}$.

3.2 Normalizing and projecting

According to the previous remark, in order to capture state-equivalence *modulo* normalization we begin with the following definition:

Definition 8 *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:*

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

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:

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

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

Gluing linear automata Given different simulation runs coming from either a unique or many S-system, we can group the corresponding linear automata into a single automaton. The formal definition (not given here) is rather straightforward and would collapse states (belonging to different automata) using parameters in order to ignore small differences in values. Maintaining a coarser grain for some variable in the collapsing process forces a change of state only when "significant" dependent variables satisfy the condition. This corresponds to producing automata concentrating on specific behaviors.

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.* [7]) with a specialized set of predicate variables whose aim is to make it easy to formulate queries on numerical quantities.

The full rendition and semantics of our query language is beyond the scope of this paper. Suffice to say that the standard CTL operators are available in English-ized form[4].

Example 9 *The main operators of our language (*cf. *CTL) are used to denote* possibility *and* necessity *over time. E.g. to express the query asking whether a certain protein p level will eventually grow above a certain value K we write* `eventually(p > K)`.

Extensions: Domain Dependent Queries We 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 asked by biologists in their daily data analysis tasks.

- `growing(`<*variable*>$_1$`, ..., `<*variable*>$_k$`)` and `shrinking(`<*variable*>$_1$`, ..., `<*variable*>$_k$`)` the `growing` special operator is a *state formula* saying that all the variables mentioned are *growing* (*shrinking*) in a given state.
- `represses(`<*variable*>$_1$`, `<*variable*>$_2$`)` and `activates(`<*variable*>$_1$`, `<*variable*>$_2$`)` this special predicate is to be interpreted as a *path formula* stating that <*variable*>$_1$ (informally interpreted as a "gene product") *represses* (*activates*) the production of <*variable*>$_2$.
- `steady_state()` this special predicate is true is the system eventually reaches a steady state; note that its implementation could be provided either as an analytical solution of the XS-system or as a combination (macro) of TL predicates.

4.1 An Example: *Purine Metabolism*

Let us now revisit in detail the example of purine metabolism described in [3] Chapter 10 and fully analyzed in [20, 21]. The pathway for purine metabolism is presented in Figure 5. 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, 20, 21].

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

[4] We are providing an English form to the standard operators, in order to make the content of resulting language easier to manipulate for the intended audience, who has not been exposed to the notation used in Temporal Logic. We also note that, technically, we are missing EG, since we are only providing `always` as a rendition of AG.

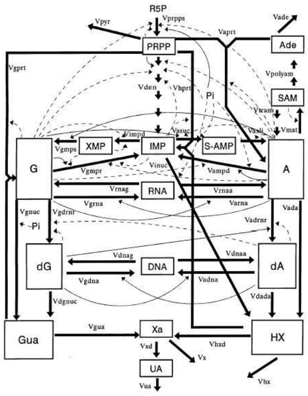


Figure 5: The metabolic scheme of purine metabolism in human. (Reprinted from [21], where a full description and further references may be found.)

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. Figure 6 shows how xssys

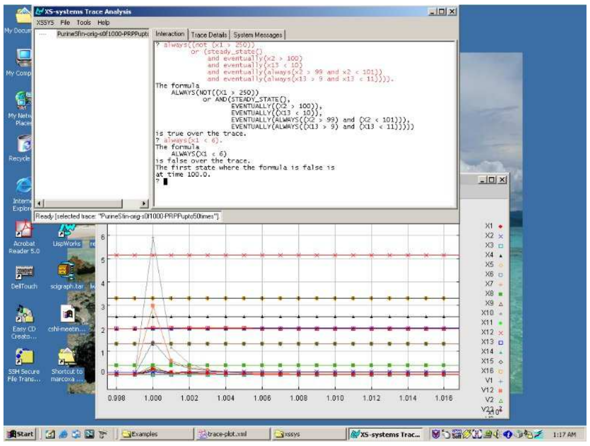


Figure 6: The in silico trace of the "final model" from [3]. We arbitrarily increased the level of PRPP (variable X1) to more than 250 at time step 100. The XSSYS system correctly answers both queries. Because of numerical effects in the floating point equality tests we had to ask a less stringent question about the steady state value of IMP (variable X2) and HX (variable X13).

responds to the query.

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.

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 cosideration time and frequency domain aspects of a trace. Of course more theoretical treatments of the subject are possible as well (*cf.* [13]). Finally, we list some of the challenges that Computational Biology needs to address in order to provide researchers with effective tools.

Reactions Models: We have primarily focused on a simple ODE model (Differential Algebraic Equations, DAE) and narrowed this even further to a model based on S- and XS-systems. Does this imply that we are accommodating a significant deviation from reality? How can a stochastic model representing small number of molecules interacting pair-wise and randomly be incorporated?

Hybrid Systems: Certain interactions are purely discrete and after each such interaction, the system dynamics may change. Such a hybrid model implies that the underlying automaton must be modified for each such mode. How do these enhancements modify the basic symbolic model?

Spatial Models: The cellular interactions are highly specific to their spatial locations within the cell. How can these be modeled with richer abstractions of automata, e.g., cellular-automata? How can we account for dynamics due to changes to the cell volume? The time constants associated with the diffusion may vary from location to location; how can that be modeled?

State Space: A number of interacting cells can be modeled by product automata. In addition to the classical "state-explosion problem" we also need to pay attention to the variable structure due to i) Cell division, ii) Apoptosis and iii) Differentiation.

Communication: How do we model the communication among the cells mediated by the interactions between the extra-cellular factor and external receptor pairs?

Hierarchical Models: Finally, as we delve into more and more complex cellular processes, a clear understanding can only be obtained through modularized hierarchical models. What are the ideal hierarchical models? How do we model a population of cells with related statistics?

Acknowledgements. We thank Mike Wigler, Misha Gromov, Ale Carbone, Thomas Anantharaman, Vivek Mittal, Rob Lucito, Jack Schwartz, Roger Brockett, Sanjoy Mitter, Shankar Sastry, Sri Kumar, Mita Desai and Harel Weinstein for offering many exciting ideas, useful contributions and ways to think about genomes, proteomes, pathways and cellular processes. Many of our close colleagues from NYU Bioinformatics group, Cold Spring Harbor Laboratory and Mt. Sinai School of Medicine have directly and indirectly contributed to this effort: Toto Paxia, Raoul Daruwalla, Joey Zhou, Archi Rudra, Naomi Silver, Chris Wiggins, Violet Chang, Elizabeth Thomas, Ken Chang and Joe West. To all of them, we are grateful. The work reported in this paper was supported by grants from NSF's Qubic program, DARPA, HHMI biomedical support research grant, the US department of Energy, the US air force, National Institutes of Health and New York State Office of Science, Technology & Academic Research.

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. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

[5] M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. -H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Theoretical Computer Science*, 138:3–34, 1995.

[7] 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.

[8] A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics.* Portland Press, London, second revised edition, 1999.

[9] 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.

[10] B. Mishra. A symbolic approach to modeling cellular behavior. Submitted to *HiPC 2002*, 2002.

[11] 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.

[12] R. Hofestädt and U. Scholz. Information processing for the analysis of metabolic pathways and inborn errors. *BioSystems*, 47:91–102, 1998.

[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] U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *SCIENCE*, 283:381–387, 15 January 1999.

[15] D. Endy and R. Brent. Modeling cellular behavior. *Nature*, 409(18):391–395, January 2001.

[16] 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.

[17] B. Shults and B. J. Kuipers. Proving properties of continuous systmes: qualitative simulation and temporal logic. *Artificial Intelligence Journal*, 92(1-2), 1997.

[18] B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.

[19] 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.

[20] R. Curto, E. O. Voit, A. Sorribas, and M. Cascante. Mathematical models of purine metabolism in man. *Mathematical Biosciences*, 151:1–49, 1998.

[21] R. Curto, E. O. Voit, and M. Cascante. Analysis of abnormalities in purine metabolism leading to gout and to neurological dysfunctions in man. *Biochemical Journal*, 329:477–487, 1998.