XS-systems: eXtended S-Systems and Algebraic Differential Automata for Modeling Cellular Behavior *

Marco Antoniotti[1], Alberto Policriti[2], Nadia Ugel[1], and Bud Mishra[1,3]

[1] Courant Institute of Mathematical Sciences, NYU, New York, NY, U.S.A.
[2] Università di Udine, Udine, (UD) Italy
[3] Watson School of Biological Sciences, Cold Spring Harbor, NY, U.S.A.

Abstract. Several biological and biochemical mechanisms can be modeled with relatively simple sets of differential algebraic equations (DAE). The numerical solution to these differential equations provide the main investigative tool for biologists and biochemists. However, the set of numerical *traces* of very complex systems become unwieldy to wade through when several variables are involved. Therefore, we propose a novel way to query large sets of numerical traces by combining in a new way well known tools from numerical analysis, temporal logic and verification, and visualization.

In this paper we describe *XS-systems*: computational models whose aim is to provide the users of S-systems with the extra tool of an *automaton* modeling the *temporal evolution* of complex biochemical reactions. The automaton construction is described starting from both numerical and analytic solutions of the differential equations involved, and parameter determination and tuning are also considered. A temporal logic language for expressing and verifying properties of XS-systems is introduced and a prototype implementation is presented.

1 Introduction

In this paper we reason about issues related to the construction of tools aiming at helping biologists and biochemists who perform simulation of complex biochemical pathways in conjunction with their experimental activities. The content of this paper is a work in progress, whose aim is to create a framework where to bring together several disciplines in a focused way.

Several biological and biochemical mechanisms can be modeled with sets of relatively simple differential algebraic equations (DAE). The numerical solution to these differential equations provide the main investigative tool for biologists and biochemists. The simple (canonical) forms of the differential equations may

* 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.

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. To cope with this problem, we propose a novel way to query large sets of numerical traces by combining in a new way well known tools from numerical analysis, temporal logic and verification, and visualization.

Our starting reference points are the S-systems, described in [15, 16], and the idea that a natural completion for that approach would be 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.

The automaton construction is 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* and from the corresponding approximate numerical solution for the DAEs, 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 here some correspondences with some of the work done in the analysis of Hybrid Systems (*cf.* [2]), which we will explore in a future work.

The automaton obtained after the second of the above two steps is "asynchronous" (untimed, *cf.* [2]), as the values of the temporal intervals connecting collapsed states are ignored, but suitable for a verification analysis of temporal properties. 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.

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 has many similarities 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. E.g., 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 DAE 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.

One of the main features of our approach is the intermixing of both quantitative and qualitative modeling, In particular, 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 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. 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.

Preliminaries

In the following we will build on ideas introduced in [15, 16] (from which we will borrow also most of the notation) and, e.g., [5].

S-systems. 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 form described in [15, 16] 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 [16] 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 $\dot{X}_i$ [11]. 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. When a numerical approximation is involved, the notion of *time-step* "**step**" becomes central to our considerations.

Example 1. Consider the following example consisting of the pathway of a so-called *repressilator system* [7]. The metabolic map corresponding to the above system is shown in Figure 1 (a). 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.

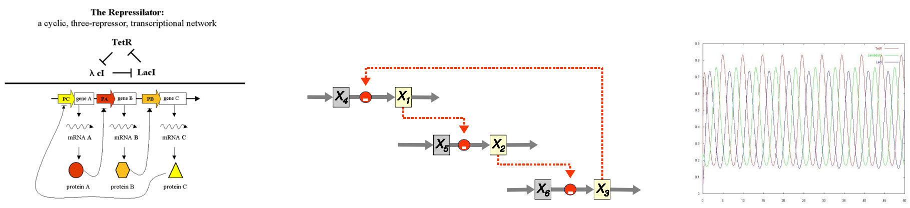
The following is the S-system corresponding to the above metabolic map:

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

with X_4, X_5, and X_6 as controls (independent variables).
Figure 1(b) shows the oscillatory trace of the system (*cf.* [7] for a discussion of the numerical and analytical analysis of the system).

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

For example, in [6] 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



(a) Pathway (b) Rendition (c) Simulation

Fig. 1. The repressilator system metabolic map (a) (reprinted from [7]), its rendition in "cascade" form (b) and its oscillatory trace (c).

the lack of quantitative information forces simulation in 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 [4]. 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.

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 [3], 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 [9], 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 [13] Cellerator is presented, a Mathematica package for biological modeling that bears many similarities with our project here.

Using a Temporal Logic query language to analyze continuous systems is investigated also in [14], as an extension of the *Qualitative Reasoning* approach (*cf.* [12]).

Finally, in [1] 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.

2 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.

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 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.*

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

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

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

Construction of a *Collapsed* automaton. We can easily construct an (in general unreduced) automaton $\mathcal{A}_{\mathcal{S}}$ by simply associating a different state to each tuple $\langle X_1(t) \dots X_n(t) \rangle$ 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.

Consider the function $X_i(t)$ at times $t_i, t_{i+1}, \dots t_{i+5}$ as depicted in Figure 2 (a). 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. Note that this is not much different than what it is done by "untiming" a Timed Automata. These automata 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).

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

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

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. This construction is extended to the full collection of variables as follows:

Definition 4. *The relation $R_{\boldsymbol{\delta}}$ holds between two states $s_k = \boldsymbol{X}(t{+}k\,\mathbf{step}\,)$ and $s_{k+j} = \boldsymbol{X}(t{+}(k{+}j)\,\mathbf{step}\,))$ with $j > 0$, if and only if, for each $i \in \{1, \dots, n{+}m\}$,*

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

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

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($\boldsymbol{\delta}$, **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 ∈ δ (|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.

Consider again the function $X_i(t)$ described in Figure 2 (a). In Figure 2 (b) 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

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

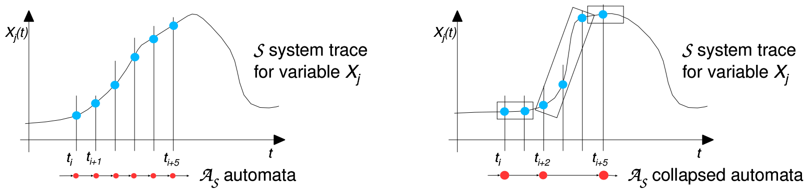
Now suppose to have a different function $X_k(t)$. We associate to $X_k(t)$ the collapsed automata $\mathcal{A}_{\mathcal{S}\,k}$, such that

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

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 $\mathcal{A}_{\mathcal{S}\,j}$, $\mathcal{A}_{\mathcal{S}\,k}$.



(a) Simple (b) Collapsed

Fig. 2. (a) Simple one-to-one construction of the "trace" automata $\mathcal{A}_{\mathcal{S}}$ for a S-system $\mathcal{S}$, and (b) the effects of the *collapsing* construction of the "trace" automata $\mathcal{A}_{\mathcal{S}}$ for a S-system $\mathcal{S}$.

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

Definition 5. *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:*

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

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 $\{\nu_i\}$ defined as $\nu_i(U,t) = X_i$ if $X_i \notin U$, or $\nu_i(U,t) = 1$ otherwise. Then the normalization corresponds to projecting with respect to the set of variables U.

3 Pathways Simulation Query System

In this section we briefly outline a language that can be used to inspect and formulate queries on the simulation results of XS-systems. The language is called ASySA (*Automata S-systems Simulation Analysis* language) and is used for expressing and verifying temporal properties of XS-systems and ADA systems. ASySA is essentially a *Temporal Logic* language (*cf.* [8]) 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 ASySA is beyond the scope of this paper. Suffice to say that the standard CTL operators are available in

English-ized form[4]. The main operators in ASySA (and 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*>$_2$.

A Simple Example. Suppose we have a system like the well known *repressilator* system [7], coded as an *S*-system, as displayed in Figure 1. One of the proteins in the system is *LacI*. We can easily formulate and compute the following query oscillates(lacI_low, lacI_high) which can be translated into a regular Temporal Logic formula stating

```
eventually (not (always(lacI_low)) or (always(lacI_high))))
```

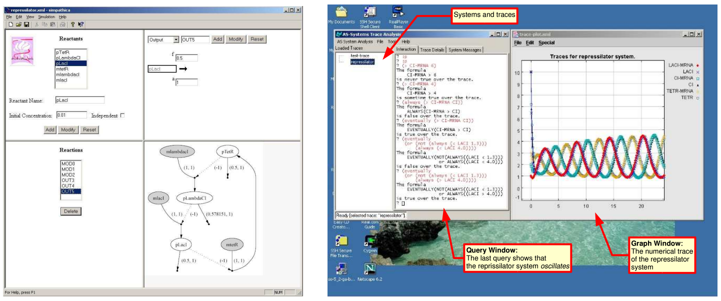
where lacI_low $\equiv$ (lacI_low $\leq$ *low*) and lacI_high $\equiv$ (lacI_high $\geq$ *high*), for appropriate values of $low < high$. The query asks the system whether the main property of the repressilator system holds over the length of the trace.

Implementation. We have implemented a prototype system embodying the concepts we have described in the previous sections. The system we describe here is the analysis component of the larger Simpathica (*Sim*ulation of *Path*ways and *I*ntegrated *C*oncurrent *A*nalysis). Figure 3 shows the main windows of the Simpathica pathway simulation tool and of the XSSYS analysis tool with the repressilator trace loaded and several queries performed.

4 Conclusions and Future Work

We have presented a "work in progress" that aims to construct a useful analysis tool by combining in a novel way several tools and techniques from Computer Science, Engineering and Biology/Biochemistry. By combining different

[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.



(a) Simpathica Main Window.

(b) XSSYS Interaction.

Fig. 3. (a) The Simpathica main window. Reactants are entered on the upper left side and single reactions are entered in the top right side. Their list and graphical rendition appears in the bottom quadrants. (b) A view of the main window of the XSSYS XS-system analysis tool.

and relatively simple components we obtain a synergistic effect that allows us to construct an efficient and effective analysis tool. We motivated our approach by analyzing a simple, yet interesting synthetic biological system: the "repressilator" [7].

Future Work. There are several topics we will investigate in our future work. First of all, since our collapsed automata relies on a "linear" approximation of a system trace, we will investigate the connections with the body of work on the analysis of Hybrid Systems (*cf.* [2]).

We are aware that our approach is be generalizable in several ways. We will explore ideas from *Signal Processing* in our future work. The "model checking" algorithm we have implemented so far is extremely simple minded (yet extremely efficient), and is exploits the *linear* and *finite* nature of the XS-systems traces. This may turn out to be insufficient when comparing different traces produced under different conditions. Nevertheless, we conjecture that because of the special format of the XS-system traces we will be able to construct simple and efficient algorithms even in that case, without resorting to the full complexity of general Model Checkers and Theorem Provers.

Finally, the study of the XS-systems' canonical form poses two sets of problems. First, the use of XS-systems automata as *semantics* of S-systems in connection with the modular design of large maps simulating biochemical systems is still not completely resolved. Secondly, the *symbolic* manipulation of the DAE, algebraic constraints and collapsed automata, along with novel property checking algorithms is still open. We conjecture that the specialized and constrained form of XS-systems will allow us to successfully walk the fine line between efficiency, efficacy and expressiveness.

References

1. 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.
2. 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.
3. U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *SCIENCE*, 283:381–387, 15 January 1999.
4. 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.
5. A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics*. Portland Press, London, second revised edition, 1999.
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. M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.
8. 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.
9. D. Endy and R. Brent. Modeling cellular behavior. *Nature*, 409(18):391–395, January 2001.
10. R. Hofestädt and U. Scholz. Information processing for the analysis of metabolic pathways and inborn errors. *BioSystems*, 47:91–102, 1998.
11. 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.
12. B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.
13. 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.
14. B. Shults and B. J. Kuipers. Proving properties of continuous systmes: qualitative simulation and temporal logic. *Artificial Intelligence Journal*, 92(1-2), 1997.
15. E. O. Voit. *Canonical Nonlinear Modeling, S-system Approach to Understanding Complexity*. Van Nostrand Reinhold, New York, 1991.
16. E. O. Voit. *Computational Analysis of Biochemical Systems A Practical Guide for Biochemists and Molecular Biologists.* Cambridge University Press, 2000.