Multiple Biological Model Classification: From System Biology to Synthetic Biology

M. Antoniotti, P.E. Barbano, W. Casey, J.-W. Feng, N. Ugel, B. Mishra [1,2]

Bioinformatics Group, Courant Institute of Mathematical Sciences
New York University
New York, U.S.A.

Abstract

This paper describes how to classify a family of biochemical pathways and circuits in terms of their temporal behavior by systematic application of time-frequency analysis and temporal logic based model checking. There are two immediate pay-offs for this approach. First, a family of models obtained by systematic perturbations of an archetypal (e.g., wild-type or ancestral) model, when classified in this manner, can help in identifying various incorrect or implausible features of the model. Second, specific hypotheses about various features of a given model can be automatically generated from such analysis and then subjected to experimental verification. We illustrate, by two examples, how our approach can be used (1) to understand the behavior of any individual topologically distinct circuit among the set of 125 synthetic biological circuits *created out of similar elements, or (2) to ascertain the correctness of a well known Yeast Cell Cycle model by checking a family of perturbed models created through single and double mutations. We also show how this tool interacts with a propositional temporal logic model-checking system to present qualitative distinctions among the groups within the family of biological circuits or among the different multi-modal behaviors of a single pathway. The paper also poses several challenging open problems in computer science, related to our basic* generate and test *framework (with heuristics) used for systematically producing a set of temporal logic sentences that can be tested against different datasets.*

Key words: Computational Systems Biology, Temporal Logic, Time-Frequency Analysis

1 Introduction

Understanding biology by modeling cellular processes and genome evolution has emerged as a challenging new area: "systems biology." Sitting at the interface of mathematics and biology, this subject aims to address many questions requiring consilience of elegant ideas and concepts from applied mathematics, theoretical computer science, logic and physical modeling. The impulse has come from better understanding of processes involved at molecular level, technology at meso- and nano-scale, ability to perform high through-put experiments and amount of genomic and proteomic data that can now be generated and made publicly available for processing.

Answering the questions posed by this challenge requires a comprehensive approach leveraging several technologies and mathematical tools. As an example of this class of challenges we tackle an interesting problem posed by the analysis of the series of experiments of Leibler and Elowitz [6] and Guet et al. [7], in which the authors design and

[1] This research has been supported by DARPA under the BioCOMP Program.
[2] Email: {marcoxa,barbano,wcasey,jiawu,nadia,mishra}@bioinformatics.nyu.edu

implement *in vivo* a family of combinatorial circuits. Our long term goal is to solve the problem of mapping and reconstructing a mathematically sound and complete model to a set of wet-lab observations. In the specific case of Guet's 125 combinatorial *in vivo* circuits, we want to be able to map the behavior of each of them to a model representing one of the standard Boolean gates.

The original motivation for designing such a family of synthetic networks by combinatorial variations of the network topology were given as follows [7]: "A central problem in biology is determining how genes interact as parts of functional networks. Creation and analysis of synthetic networks, composed of well-characterized genetic elements, provide a framework for theoretical modeling. ... Combinatorial synthesis provides an alternative approach for studying biological networks, as well as an efficient method for producing diverse phenotypes *in vivo*." Nonetheless, lack of efficient tools for modeling and analysis of such synthetic networks has hindered many possible applications of these networks. Clearly, with appropriate tools, one could foresee applications where millions of randomly generated networks could be screened for selection of primitive circuits with specific properties (robustness, immunity to noise, etc.), or as building blocks of larger circuits with specific temporal properties, or even as scaffold structures for measuring kinetic parameters of a component as it operates *in vivo*. Here, we suggest that Simpathica/XSSYS and the ancillary modules NYUSIM and NYU BioWave along with their planned software progenies respond to these demands quite well. I.e., we show how the combination of Simpathica modules can be employed synergistically to analyze the set of Guet's biological combinatorial circuits, by providing a semi-automated way to classify them based on the profiles of their behaviors. The classification of behaviors is accomplished by a careful application of time-frequency clustering techniques and by a modified model-checking approach that directly tests for the truth value of Boolean expressions over traces of the system. Following such demonstration, we introduce an extension of this approach based on a *generate-and-test* procedure for temporal logic formulæ in order to automate a *discovery system* that can further aid a working biologist. We also examine how well our procedures work on a model of the Yeast Cell Cycle [13,14], for which a preliminary group of three datasets had been produced.

1.1 Guet's Biological Circuits

As an example of how the Simpathica set of tools (NYU BioWave and NYUSIM) may be used in analyzing biological systems, we will focus on a "bio-circuit" originally designed by Guet and others [7].

In the scheme created by Guet and colleagues, the authors used a combinatorial method to generate a library of networks with varying connectivity and implemented them as plasmids capable of transfecting *Escherichia coli*. These networks were composed of genes encoding the transcriptional regulators LacI, TetR, and λ CI, as well as the corresponding promoters. Although the networks had time-varying output trajectories for a fixed input and implemented sequential circuits, Guet et al. characterized their phenotypic behaviors as resembling binary logical/combinatorial circuits, with two chemical "inputs" and a fluorescent protein "output." Nevertheless, the biological experiments indicated a rich and diverse set of functions dependent on network connectivity and raised questions about how to design appropriate computational tools to analyze them.

In [7], the authors generated a combinatorial library composed of a small set of transcriptional regulatory genes and their corresponding promoters and varied their connectivity in a combinatorially exhaustive manner. They chose genes of three well-characterized

prokaryotic transcriptional regulators: *LacI*, *TetR*, and λ *cI*. The binding state of *LacI* and *TetR* can be changed with the small molecule inducers, isopropyl β-D-thiogalactopyranoside (IPTG) and anhydrotetracycline (aTc), respectively. In addition, they also selected five promoters regulated by these proteins (i.e. LacI, TetR, and λ CI), which span a rather broad range of regulatory characteristics—e.g., repression, activation, leakiness, and strength. Two of the promoters are repressed by lac (to be referred to as PL1 and PL2), one is repressed by tet(to be referred to as PT), and finally, the last two are regulated by λ ci, one positively ($P\lambda_+$) and one negatively ($P\lambda_-$). Their genetic assembly scheme ensured that each network in the library has the following structure: P_i-*lac*-P_j-λ *ci*-P_k-*tet*, where each P_i, P_j, and $P_k \in \{\text{PL1}, \text{PL2}, \text{PT}, P\lambda_+, P\lambda_-\}$ is implemented as any of the five promoters. Thus, the regulatory genes on each plasmid interact (i.e., activate or repress) with one another, generating networks with diverse connectivities. A separate plasmid consisting of a reporter *gfp* and repressed by λ *ci* is used to measure the biological activity of the synthetic network through the fluorescence of gfp.

In this paper, we will model all possible $5^3 = 125$ different networks and by examining their trajectories group them into various classes and examine how well this grouping coincides with the others based on topology. Since the networks constructed this way encompass a wide range of motifs (including negative and positive feedback loops, oscillators, and toggle switches) they present an interesting family of trajectories to NYU BioWave.

In summary the system to be analyzed consists of the following:

(i) There are combinations of four genes: *lac*, λ *ci*, *tet* and *gfp*, of which the first three interact with each other by pair-wise activation or repression and the last one (*gfp*) is used as an output. The corresponding proteins are denoted as lac, λ ci, tet and gfp. Their concentrations will be indicated by the notation $[x]$ (e.g., $[lac]$ is the concentration of *lac*-mRNA and [lac] is the concentration of lac-protein). The temporal rate of change of concentration will be denoted as $[\dot{x}]$.

(ii) The small molecule inducers IPTG and aTc act as the inputs to the system through their inactivation of the *lac* and *tet* genes, respectively.

(iii) There are five Operons: two lac-based: PL1, PL2; two λ ci-based: $P\lambda_-$, $P\lambda_+$; one tet-based: PT.

(iv) Total $5^3 = 125$ different combinatorial circuits are possible. A circuit is denoted as P_i-*lac*-P_j-λ *ci*-P_k-*tet*, indicating that P_i determines the transcriptional state of *lac*; P_j determines the transcriptional state of λ *ci* and P_k determines the transcriptional state of *tet*.

(v) For instance the circuit $P\lambda_+$-*lac*-PL1-λ *ci*-PL1-*tet* has the following connections:
   (a) *lac* is *activated* by λ ci.
   (b) λ *ci* is *repressed* by lac, and lac is inactivated by IPTG.
   (c) *tet* is *repressed* by lac, and lac is inactivated by IPTG.
   (d) *gfp* is *repressed* by λ ci.

In our analysis we will make several simplifying assumptions: (1) All genes have similar time constants; (2) mRNA's instantaneous concentration depends on the transcription process, its leakiness and its instability (i.e., how it degrades); (3) A protein's instantaneous concentration depends on the translation process and its degradation. Their dynamic state-evolution equations can be written in terms of two intrinsic parameters α (governing mRNA) and β (governing protein) as well as Hill-coefficient like terms (n and k), leakiness term (ρ) and saturation terms (θ).

If x denotes a gene and X its corresponding protein, we have the following equation

for x's transcription:

$$[\dot{x}] = -[x] + \alpha[\rho + f_x(\theta, [Y], [u_y])]$$

where $f_x(\theta, [Y], [u_y]) = \dfrac{1 + \theta[Y]^n + [u_y]^k}{1 + [Y]^n + [u_y]^k}$. In this equation, the transcription is activated or repressed by a protein Y and Y itself is modulated by a small molecule u_y. Note that, for small values of $[u_y]$, f_x shows a sharp transition from a value of 1 (when $[Y] = 0$) to a value of θ (when $[Y] = \infty$), as Y increases. However, for large values of $[u_y]$, f_x remains at 1 (when $[u_y] = \infty$), thus inactivating the effect of Y.

Similarly, we have the following equation for X's (corresponding proteins) translation: $[\dot{X}] = -\beta([X] - [x])$.

Going back to our example circuit Pλ_+-*lac*-PL1-λ *ci*-PL1-*tet*, we can write down in a straightforward manner the corresponding ODE's as shown below:

$$[\dot{lac}] = -[lac] + \alpha\rho + \alpha\frac{1 + \theta_a[\lambda\ \text{CI}]^n}{1 + [\lambda\ \text{CI}]^n} \qquad [\dot{tet}] = -[tet] + \alpha\rho + \alpha\frac{1 + \theta_s[\text{LAC}]^n + [\text{IPTG}]^k}{1 + [\text{LAC}]^n + [\text{IPTG}]^k}$$

$$[\dot{\text{LAC}}] = -\beta([\text{LAC}] - [lac]) \qquad [\dot{\text{TET}}] = -\beta([\text{TET}] - [tet])$$

$$[\dot{gfp}] = -[gfp] + \alpha\rho + \alpha\frac{1 + \theta_s[\lambda\ \text{CI}]^n}{1 + [\lambda\ \text{CI}]^n} \qquad [\dot{\lambda\ ci}] = -[\lambda\ ci] + \alpha\rho + \alpha\frac{1 + \theta_s[\text{LAC}]^n + [\text{IPTG}]^k}{1 + [\text{LAC}]^n + [\text{IPTG}]^k}$$

$$[\dot{\text{GFP}}] = -\beta([\text{GFP}] - [gfp]) \qquad [\dot{\lambda\ \text{CI}}] = -\beta([\lambda\ \text{CI}] - [\lambda\ ci])$$

The two top-left equations model the fact that *lac* is *activated* by λ CI. The two bottom-left equations model the fact that *gfp* is *repressed* by λ CI. The two top-right equations model the fact that *tet* is *repressed* by LAC, and LAC is inactivated by IPTG. The last two bottom-right equations model the fact that λ *ci* is *repressed* by LAC, and LAC is inactivated by IPTG. We used the following parameters and simulation functions:

$$[\dot{\text{IPTG}}](t) = -\exp(-t)[\text{IPTG}](0) \qquad \theta_s = 0 \quad \text{implying suppression}$$

$$[\text{IPTG}](0) = x_0 = 3 \qquad \theta_a = 2 \quad \text{implying amplification}$$

$$[\dot{\text{aTc}}](t) = -\exp(-1.1t)[\text{IPTG}](0) \qquad \alpha = 5 \quad \beta = 1 \quad \rho = 0.1 \quad n = 2 \quad k = 2$$

$$[\text{aTc}](0) = y_0 = 3$$

and note that in our normalized equations, α is the concentration of proteins per cell from *unrepressed* promoter; $\alpha\rho$ is the concentration of proteins per cell from *repressed* promoter; β is the protein:mRNA decay rate ratio; n is the Hill (cooperativity) coefficient of the repressor; and k is the Hill (cooperativity) coefficient of the small molecule.

1.2 The Yeast Cell Cycle Model

In [14,13], Tyson and colleagues have published some fundamental works on modeling *cell cycle genes*. In Tyson's model, the transition between phase G1 of the cell cycle to the S-G2-M phase was analyzed as a bifurcation of a dynamical system.

The "traditional" model used in Tyson's work may be described as follows: the main control element of the yeast cell cycle is the protein dimer of Cyclin B (CycB) and Cyclin-Dependent Kinase (Cdk). Other proteins, CKI, Cdh1, and SK, also play a role. At the beginning of a cell cycle (early G1), both CKI and Cdh1 levels are high. CKI can

bind to CycB/Cdk to inactivate the latter, while Cdh1 rapidly degrades CycB. Thus Cdk is in inactive state. As cell grows, cell mass increases over time, which makes CycB accumulate gradually in the cell nucleus. In late G1, SK raises sharply, degrading CKI, and CycB becomes more and more active. Eventually, CycB overcomes Cdh1 and get fully activated. Active Cdk/CycB triggers several events allowing the cell to enter S-phase and DNA synthesis to start. Cdk remains inactive until the cell enters M-phase, when Cdc20 is activated and destroys enough CycB to allow Cdh1 (as well as Cdk inhibitor, CKI) to become active. As a test for the model, in SK mutant cells, CycB/Cdk cannot be activated, thus preventing the cell from entering S-phase and resulting in a fatal explosive cell mass growth. However, in SK/CKI double mutant, the cell cycle behavior is restored.

The model is described in terms of a set of nonlinear differential equations with values for kinetic parameters that have been experimentally verified.

The analysis of the Yeast Cell Cycle model in [14,13] aims at comparing the behavior of different mutants *vis-a-vis* the wild-type control, to ascertain various hypotheses. We used this model to test a preliminary prototype of our "generate-and-test" framework.

2 The Simpathica System

The NYU/Courant Bioinformatics Group has developed and implemented a computational system, Simpathica, which allows a user to construct and rigorously analyze models of biochemical pathways composed out of a set of basic reactions. Each reaction is thought of as a module and belongs to one of many types: *reversible reactions*, *synthesis*, *degradation*, and *reactions modulated by enzymes and co-enzymes* or other reactions satisfying certain *stoichiometric constraints*. If the stochastics in these reactions are ignored (i.e., mass-action models), each of these reactions can be described by a first order algebraic differential equation whose coefficients and degrees are determined by a set of thermodynamic parameters. As an example, reaction modulated by an enzyme leads to the classical Michaelis-Menten's formulation of reaction speed as essentially differential equations for the rate of change of the product of an enzymatic reaction. The parameters of such an equation are the constants K_m (Michaelis-Menten Constant) and V_{max} (maximum velocity of a reaction). In a simple formulation, such as in S-system [15,16], this approach provides a convenient way of describing a biochemical pathway as a composition of several primitive reaction modules and then automatically translating them into a set of ODE's with additional algebraic constraints. Simpathica and XS-system [11,2,4] (an extension of the basic S-System) retains this modular structure while allowing for a far richer set of modules.

The Simpathica architecture is composed of two main modules and several ancillary ones. The first main module is a graphical front end that is used to construct and simulate the networks of ODE's that are part of the model being analyzed. Simpathica uses, among other, the SBML format [12] for exchange. The second module, XSSYS is an analysis module based on a branching time temporal logic, that can be used to formulate questions about the behavior of a system, represented as a set of *traces* (time course data) obtained from wet-lab experiments or computer simulations.

Starting with a trace of a bio-chemical pathway, (i.e. a time-indexed sequence of state vectors representing a numerical simulation of the pathway) as input, Simpathica can perform the following operations.

- Simpathica answers complex questions involving several variables about the behavior of the system. To this end we defined a query language based on a branching time temporal

logic formalism.

- Simpathica stores traces in an ancillary database module, NYUSIM, and allows easy search and manipulation of traces in this format. The analysis tools allow these traces to be further examined to extract interesting properties of the bio-chemical pathway.
- Simpathica classifies several traces (either from a single experiment or from different ones) according to features discernible in their time and frequency domains. Multi-resolution time-frequency techniques can be used to group several traces according to their features: steps, decreases, increases, and even more complex features, such as, memory. Simpathica contains a prototype subsystem (called NYU BioWave), which implements these classification procedures using Matlab.

With these tools, Simpathica provides an environment to suggest plausible hypotheses and then, refute or validate these hypotheses with experimental analysis of time-course evolution. It also allows investigating conditions or perturbations under which a metabolic pathway may modify its behavior to produce a desired effect (an instance of a control engineering problem).

2.1 Temporal Logic Model Checking

The XSSYS Simpathica back-end implements a specialized model checking labeling algorithm that, given a "model trace" and a temporal logic formula expressed in an extended CTL form, can state whether the formula is true or false, in which case a *counterexample* is provided: i.e. the system gives an indication at which point in time the formula becomes false.

A full description of the syntax and semantics of the temporal logic language manipulated by Simpathica/XSSYS is beyond the scope of this paper and hence, omitted. For the purpose of the present discussion, it suffices to assume that all the standard CTL operators are available in a somewhat anglicized form [3]. The main operators in XSSYS (and CTL) are used to denote *possibility* and *necessity* of *propositions* over time. In our case such proposition involve statements about the value of the variables representing *concentrations* of molecular species. 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)`.

We 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 asked by biologists in their daily data analysis tasks.

As an example, we can formulate queries like

```
eventually(not always(LacI < 1.3) or always(LacI > 4.0)).
```

The query expresses the fact that the value of the 'LacI' variable "oscillates" between the two values of 1.3 and 4.0. The system being analyzed is the *repressilator* system of Elowitz and Leibler. The analysis tool provides counter examples when input query fails to hold true or restricts the conditions under which the query can be satisfied.

A more through introduction to XSSYS and its capabilities can be found in [3,4].

[3] We have provided an English form to the standard operators in order to make the content of resulting language to be easily manipulated by the intended audience—in this case biologists with no exposure to the notations of Temporal Logic. We also note that, technically, we are missing EG, since we have only provided `always` as a representation of AG.

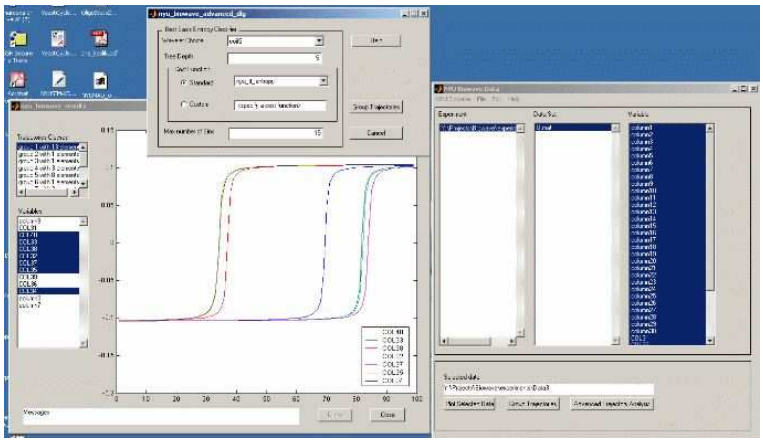


Fig. 1. A view of the NYU BioWave user interface. There are three windows visible. In the background there is the dialog showing the connection to NYUSIM, in the foreground there are the two windows that constitute the "classifier inspection tool". The group being reviewed comprises various *step* functions from a test data set of functions of various shapes (the functions are normalized before being plotted.)

2.2 Time-Frequency Analysis within NYU BioWave

Many biological experiments (especially *in silico* experiments) produce *time course data* which can be analyzed both in time and frequency domains to extract interesting functional properties. To this end we have constructed NYU BioWave, a tool that can find similarities in the 'shape' of time course data, that is, it can easily group together measurements of different quantities based on their time-course behavior. As an example, it can group together all trajectories that present a 'step' feature, thus easing the detection of relationships among observed variables. Moreover, it can do so across several datasets (e.g. datasets corresponding to different values of controlled parameters.)

The mathematical theory behind the NYU BioWave tool is primarily based upon *Multi-resolution Time-Frequency Analysis* through *Wavelet Decompositions* [10]. In Figure (1) we show a simple and artificial test case used to validate NYU BioWave capabilities, and the NYU BioWave user interface.

NYU BioWave utilizes a multi-scale basis selection algorithm. The first example in this class of algorithms, the *best basis algorithm* can be found in [5]. There, given bi-orthogonal wavelet filter denoted by $[v, w]$, the best basis algorithm defines a method for searching a subset of $O(M)$ (the set of orthogonal transformations in $\mathbb{R}^M$). $O(M)$ is generated by wavelet filter trees $[v, w]$, and has a number of interesting mathematical properties, which we do not discuss here (again, cfr. [5]).

NYU BioWave defines a method for searching a subset of $O(M)$ that uses a tree pruning algorithm whose operation is governed by the function α. The original best-basis algorithm is then an instance (with α being the entropy function) of the algorithm implemented in NYU BioWave.

NYU BioWave eventually associates a 'score' $s_i \in \mathbb{R}$ to each trajectory f_i examined, with $i = 0 \ldots n$. Currently, the 'score' is a value derived from the *entropy* of the trajectory. The set of scores is simply $S = \langle s_i \rangle$. These scores are then partitioned in groups, according to the characteristics of their distributions. At present, NYU BioWave implements a simple grouping scheme that optimizes *gaps* between the groups. The scheme is based on the computation of a "moving average" $\hat{\mu}$ and relative standard deviation $\hat{\sigma}$ of the "distances" $D_S = \langle s_{i+1} - s_i \rangle$ between the scores. Two scores s_i and s_j are grouped separately if $|s_j - s_i| > \hat{\mu} + 2\hat{\sigma}$. Of course, this method of clustering entropy scores is rather coarse

IPTG	*aTc*	*IPTG*	*aTc*	*IPTG*	*aTc*	*IPTG*	*aTc*
0.0	0.0	0.0	3.0	3.0	0.0	3.0	3.0

Table 1
Initial concentrations of the *input* molecules (to be interpreted as μMol) IPTG and aTc. The concentrations of IPTG and aTc decay exponentially in each experiment. Each set of inputs is fed in turn to the 125 circuits. Each simulation run until a steady state is reached.

and arbitrary and requires further research. However, we note that this approach works well when there is a known correlation among the f_i's (as is the case with the example described in Section 1.1).

An alternative and a more sophisticated way to assign a score to each trajectory would be to compute the set $\{\epsilon_{ij}\}$ defined as the "entropy of the coefficients of the representation of f_i, with respect to the best basis computed for f_j." We could then group f_i and f_j together, based on $\|\epsilon_{ij} - \epsilon_{ii}\| \leq \kappa$, for a given parameter κ. In other words, we consider a pair of functions similar, when they are 'close' with respect to their representation in terms of the optimal basis associated to the function [4].

Finally, we note that, this clustering problem is quite difficult to solve in a complete general way, and we will explore it in more detail in a different setting.

3 Analysis

We ran simulations for each of the 125 circuits with the inputs listed in Table 1. The simulations were run using Matlab standard Ordinary Differential Equations integrators In each run all 125 circuits were tested until a steady state was reached. The result was a set of 125 trajectories for each input pair ⟨IPTG, aTc⟩ (i.e. 4 sets). Two kinds of analysis were performed on the resulting sets of data: a *time-frequency* analysis using NYU BioWave and a classification of combinatorial circuits using Simpathica/XSSYS.

Analysis: Time-Frequency. The motivating example was taken from the work of Guet et al. We analyzed the ODE behavior using the non-linear projection discussed in Section 2.2. The results are 125 projection points in the range $[1.3905 \times 10^{-2}, 2.6561 \times 10^{-2}]$ which are then divided into four groups with our multi-resolution-adaptive binning algorithm. Figure (2) shows Group 4. There is consistency in the groups both in qualitative description of the element functions as well as the derived circuit topology. Thus the low-dimensional clustering of the 125 function encodes the underlying circuitry.

Analysis: Temporal Logic. As a simple test of our Simpathica/XSSYS system, we ran a non-traditional analysis of the four sets of trajectories using Simpathica Temporal Logic analysis tool: XSSYS. Simpathica/XSSYS sorted the circuits according to the following properties.

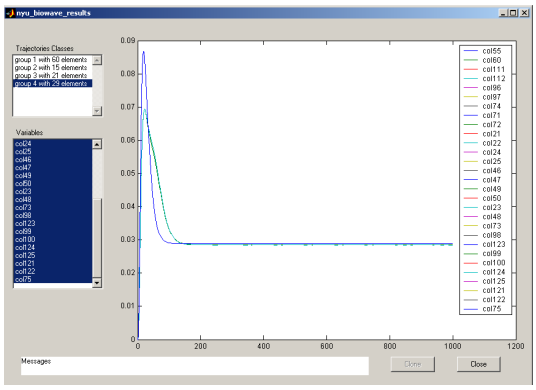
- Circuits exhibiting *switch*-like properties.
- Circuits exhibiting a *Boolean* behavior (i.e. showing a combinatorial function of the inputs).

We modified our tool to handle all these cases and proceeded in the following way.

(i) Find good *candidate* circuits, call this set C.
These are the circuits that present a variation in outputs given different inputs [5].

[4] We also note that the criteria we described is not symmetric. We will describe the detail of our approach in a different setting.

[5] This was not really necessary with respect to step 2, as the circuits eliminated would have been classified as one of the Boolean constants: either true, or false.)



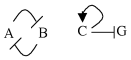
Circuit	Comment
	Circuit 71 $\langle \mathrm{PT}, \mathrm{P}\lambda_+, \mathrm{PL1} \rangle$

Fig. 2. Group 4 includes the trajectories whose shape is dominated by the topological arrangement of the plasmids in which λCI (C) activates its own transcription and neither LAC (A) nor TETR (B) have an effect on the transcription of λCI. This feature clearly eliminates the significance of the topological arrangement of the promoters before LAC and TETR. The sample diagram of this group shows λCI activating its own transcription, while the relationship is arbitrary, as long as they do not affect λCI. Circuit 71 is a sample of the diagrams representing these functions. The triple of promoters denotes the structure of the circuit.

(ii) Find which circuit $c \in C$ implements one of the basic 2-inputs Boolean functions [6].

(iii) Find which circuits admit more than 2 output values.

To test for the first property we used the following method. Each circuit was simulated given one of the input pairs in Table 1. The result is a quadruple of traces for each circuit. Next we ran a simple script testing whether the steady state value of each member of the quadruple was above or below a threshold. This corresponded to formulating the following TL query on each element of the quadruple.

```
eventually(always(c < threshold)).
```

`threshold` was varied in the range $[0.5 \ldots 5.0]$ with 0.1 increments. Any circuit c which failed the query for some element of the quadruple was marked as "*potential circuit*."

The next step was to test which of the potential circuits actually represented a Boolean function. This step immediately posed a problem, as certain circuits exhibit a two-valued response to the inputs from Table 1, while other exhibit three-valued response. Moreover, the choice of what constitutes a *high* and *low* response appeared rather arbitrary. To cope with this problem we devised a procedure that automatically constructs TL formulæ of the form

```
eventually(IPTG = 0 and aTc = 0 ==> eventually(always(low(c))))
and eventually(IPTG = 0 and aTc = 3 ==> eventually(always(high(c))))
and eventually(IPTG = 3 and aTc = 0 ==> eventually(always(high(c))))
and eventually(IPTG = 3 and aTc = 3 ==> eventually(always(high(c)))).
```

The formula checks whether circuit c represents an OR gate [7]. Mixing the `low` and `high` functions yields tests for all the other 15 two inputs Boolean functions. However, the results of the tests depend on a threshold which can be changed. Table 2 shows which

[6] Given two inputs i_1 and i_2 there are 16 possible Boolean functions: 0, 1, i_1, i_2, $\neg i_1$, $\neg i_2$, OR, AND, NOR, NAND, XOR, NXOR, IF_1_2, IF_2_1, NIF_1_2, NIF_2_1.

[7] The outer `eventually` operator is introduced mostly as a technicality to take into account the "start up" time of the simulated systems.

Boolean Function	*Circuit*
$\neg$ *IPTG*	51 52 56 57 76 77 78 79 80 81 82 83 85
aTc	14 39 64 89 114
aTc $\rightarrow$ *IPTG*	61 62

Table 2
The classification of potential Boolean circuits given a threshold of 1.3 μMol. Each number denotes one of the circuits described in [7].

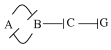
Circuit	Function	Comment
	$\neg$ *IPTG*	Circuit 85 $\langle \mathrm{P}\lambda_-, \mathrm{PL2}, \mathrm{P}\lambda_+ \rangle$
	*aTc*	Circuit 114 $\langle \mathrm{P}\lambda_+, \mathrm{PT}, \mathrm{P}\lambda_- \rangle$
	*aTc* $\Rightarrow$ *IPTG*	Circuit 61 $\langle \mathrm{PT}, \mathrm{PT}, \mathrm{PL1} \rangle$

Table 3
Some of the circuits implementing the logic-combinatorial circuits found with threshold parameter equal to 1.3μMol. Again the triple of promoters denotes the structure of the circuit.

circuits have been identified as which Boolean circuit, given a threshold of 1.3 μMol.

4 Generating Biological "Facts" from a Set of Traces

The analysis of Guet's circuits using our temporal logic analyzer suggested a generalization of the approach to analyze more complicated behavior in a combination of experiments. The natural generalization lies in the application of a time tested *generate and test* framework, where temporal logic formulæ are constructed in increasing order of complexity and tested against the data available (i.e. the set of traces.)

Given a set V_S of "variables" in a dataset, the generation algorithm must use several heuristics to constrain the size of the set $\mathcal{F}$ of formulæ to be analyzed, as a simple counting argument on the structure of the concrete syntax of CTL formulæ reveals that the number of formulæ of *syntactic depth* d is $\Omega\left(2^{2^d}\right)$: obviously too large a number to consider, even for the simple case of $d = 3$. Given a number of relatively straightforward heuristics, the formula generation and testing procedure can be kept under control, although the worst case scenario still applies.

The set of heuristics $\mathbf{H}_{\mathcal{F}}$ involved are based on (a) a (arbitrary) lexicographic ordering of the variables, (b) an accounting of the symmetries in the binary operators of the underlying temporal logic language, (c) a special interpretation of the *time* variable, (d) the singling out of a set of "special" formulæ, (e) imposing limits on the number of disjunction used, (f) a heuristic to check only the formulæ involving small connected components of variables (e.g. $v_i \rightarrow v_j \rightarrow v_k$) based on what is known about the structure of the model pathway, and, finally (g) allowing user intervention in establishing an *a priori* "ranking" of formulæ to be examined. Also, user supplied ranges for the values of the variables involved are taken into account. In essence the procedure performs the steps shown in Figure (3).

Because of the set of heuristics used, the resulting set of formulæ has limited size. Once the set of formulæ F has been generated, each of its members can be checked against the datasets comprising an experiment. The result will be a set of valuations for each $f \in \mathcal{F}$ with respect to each dataset considered; e.g. a dataset corresponding to the Guet's circuits.

Procedure Formula Generation:

(i) **Input**: a set of variable V_S from an experiment.

(ii) For each formula template from the set:
  (a) `Represses(P1, P2).`
  (b) `Activates(P1, P2).`
  (c) `Steady_state().`
  (d) `Constant(P, t1, t1).`
  (e) Formulæ representing the response of the system to a particular input at time `ti` (e.g. an *impulse* or a *sustained* input.)

generate the set $\mathcal{F}$ of all possible combinations of instantiated formulæ using only the elements of V_S and constrained by the heuristics set $\mathbf{H}_{\mathcal{F}}$.

Fig. 3. The formula generation procedure.

Report on "Test Experiment Tyson WT, 1 Mutant, 2 Mutants."
RESULTS
The results refer to the following datasets:

- The first dataset is named "Ian's Experiment/Tyson Yeast Dataset WT."
- The second dataset is named "Ian's Experiment/Tyson Yeast Dataset Mut1."
- The third dataset is named "Ian's Experiment/Tyson Yeast Dataset mut2."

...

84. CDH1 less than or equal to 1.0071783 will always hold until CDH1 activates CYCB, is true in the first dataset, is true in the second dataset, and is false in the third dataset.

85. CDH1 represses CYCB implies CYCB is greater than or equal to 0.65, is false in the first dataset, is true in the second dataset, and is true in the third dataset.

86. CDH1 greater than or equal to 1.0071783 will always hold until CDH1 activates CYCB, is false in the first dataset, is true in the second dataset, and is true in the third dataset.

87. eventually, CDH1 is less than or equal to CYCB, is false in the first dataset, is true in the second dataset, and is true in the third dataset

Table 4
A fragment of the "biologically interesting factoid" story produced by the generation system. The system actually produced 234 such sentences involving the species CDH1 and CYCB and a number of "interesting values" they can assume. Each "sentence" is a simple transliteration of a temporal logic formula into English, for the benefit of the reader.

We have some preliminary tests available for our formula generation procedure and they suggest several promising future developments, especially in conjunction with natural language based user interfaces [1]. Given a number of datasets and a set of "interesting" values for the variables in V_S, the "factoid generation" system produces an HTML formatted output. Table 4 shows an excerpt from the output produced by analyzing three datasets obtained by simulation of the Yeast Cell Cycle models described in [13,14].

5 Discussion

In this paper, we have described a set of tools within Simpathica system, specifically designed to perform temporal analysis of the trajectories of bio-chemical pathways and to classify them into groups for further characterization. The capabilities of these tools are illustrated through a detailed analysis of a combinatorial approach to bio-circuit design, following the scheme suggested by Guet et al. [7]. A generalization of the analysis approach that can semi-automatically *generate and test* a set of formulæ $\mathcal{F}$ was also briefly discussed in conjunction with Tyson's Yeast Cell Cycle model [13,14].

Arguably, much research remains to be done before biological circuit design can be fully and faithfully carried out in this manner, but this style of analysis may ultimately provide a better scheme over other competing approaches based on tedious hand design

or *in vitro* evolution. Furthermore, these ideas suggest that our approach will also allow one to study phenotypical properties of a genetic network in wild type, by concomitantly studying a family of mutants and double-mutants obtained by combinatorial knock-outs. Same approach also suggests that the functional properties of a novel gene can be studied by combinatorially mixing it with a family of artificial genetic networks that have already been characterized. Thus, such combination of biological experiments with computational and mathematical tools promises to open up new and exciting opportunities.

References

[1] M. Antoniotti, I. T. Lau, and B. Mishra. Naturally Speaking: A Systems Biology Tool with Natural Language Interfaces. CIMS-TR 853, Bioinformatics Group, Courant Institute of Mathematical Sciences, New York Universisty, June 2004.

[2] M. Antoniotti, F. C. Park, A. Policriti, N. Ugel, and B. Mishra. Foundations of a Query and Simulation System for the Modeling of Biochemical and Biological Processes. In *Proc. of the Pacific Symposium of Biocomputing (PSB'03)*, 2003.

[3] 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.

[4] M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Model Building and Model Checking for Biological Processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.

[5] R. R. Coifman and M. V. Wickerhauser. Entropy-based Algorithms for Best Basis Selection. *I.E.E.E. Transactions on Information Theory*, 38(2), 1992.

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

[7] C. C. Guet, M. B. Elowitz, W. Hsing, and S. Leibler. Combinatorial synthesis of Genetic Networks. *Science*, 296(5572):1466–1470, 2002.

[8] J. Keener and J. Sneyd. *Mathematical Physiology.* Springer-Verlag, 1998.

[9] P. Liò. Wavelets in bioinformatics and computational biology: state of the art and perspectives. *Bioinformatics*, 19(1):2–9, 2003.

[10] S. Mallat. *A wavelet tour of signal processing.* Academic Press, 1999.

[11] B. Mishra. A symbolic approach to modeling cellular behavior. In *Proceedings of HiPC 2002, Bangalore,* India, December 2002.

[12] System Biology Markup Language. `http://www.smb-sbml.org/sbml/docs/index.html`, 2002.

[13] J. J. Tyson, K. Chen, and B. Novak. Network dynamics and cell physiology. *Nature Reviews*, 2:908–916, 2001.

[14] J. J. Tyson and B. Novak. Regulation of the Eukaryotic Cell Cycle: Molecular Antagonism, Hysteresis, and Irreversible Transitions. *Journal of Theoretical Biology*, 210:249–263, 2001.

[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.