Artificial Gene Regulation:
A Data Source for Validation of Reverse Bioengineering

Christian Knüpfer [1,2]   Peter Dittrich[1]   Clemens Beckstein [2]

[1] Bio Systems Analysis Group
Jena Centre for Bioinformatics (JCB) and
Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena
Ernst-Abbe Platz 1-4, 07745 Jena, Germany

[2] Artificial Intelligence Group
Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena
Ernst-Abbe Platz 1-4, 07745 Jena, Germany

Abstract

In systems biology a large number of new algorithms are developed that extract information from high throughput genomics and proteomics data. In order to evaluate these new algorithms, we have developed an artificial gene expression data source, which is able to deliver an arbitrary large amount of artificial gene expression data, where the underlying network is precisely known in all details. Here, we describe the system, which can generate artificial gene expression networks including protein-protein interaction dynamics, transcriptional and post-transcriptional regulation, explicit modeling of RNA and protein concentration, and dimerization (RNA-RNA, protein-protein, RNA-protein). Dimers and RNA can function as regulators like proteins. We demonstrate the structural properties and dynamical behavior of the generated networks. Depending on the parameter settings given by the user, our method generates regulatory systems that exhibit a wide spectrum of network structures and dynamical behaviors.

1   Introduction

High-throughput genomics and proteomics experimental methods generate a growing amount of systematic data of biological cells under controlled conditions. This data allows to derive automatically models of dynamical cellular processes, such as gene regulation, by using computer based data analysis and hypothesis generating methods. Within the field of systems biology, a growing amount of algorithms for the analysis and reconstruction of genetic regulatory networks have recently been proposed, such as boolean based algorithms [5, 7], continuous approaches [2, 3, 16], and probabilistic frameworks [10, 17]. These algorithms take experimental

data (such as RNA abundance measures from gene expression profiling, protein concentrations from protein 2-D gels, or further information from bioinformatics databases) as input and deliver estimates on the regulatory interaction among genes.

Besides the fact that most of them lack a precise formal specification of what they are supposed to do in the first place, it is also very difficult to evaluate their explicative power based on real biological or artificial data [3, 9, 13]. The matter appears to be even more difficult, because there is not enough real data available for a satisfying evaluation. Nevertheless, the performance of a method on a small number of real world problems is often used as an argument for its quality, which is dangerous from a methodological point of view. On the other hand, artificial data, such as data generated from random boolean networks, is criticized for not reflecting the real biological problem [6].

Our aim therefore is to generate artificial data that are similar to biological data. Our strategy to achieve this is to create *realistic* (i.e., biologically plausible) "life-like" artificial regulatory networks as a data source. We assume that if we build a dynamical network model that incorporates many mechanisms found in natural regulatory networks, then the resulting data will also reflect the nature of genetic regulatory systems.

In the first part of this paper we introduce a new architecture (framework) that allows to generate a large variety of artificial gene regulatory networks. The architecture is flexible, such that it allows to integrate new biological knowledge easily. In the second part, the properties of our method are investigated by looking at the structure of the networks generated with the new architecture and demonstrating the dynamical behavior of the resulting systems.

2 Framework

Our goal is to build a generator for complex dynamical networks that should be regarded as synthetic replicas of real biological genetic regulatory systems. Besides being biological plausible, the generator should be modular, flexible, and extendable. Furthermore, the generated networks have to show sufficient complexity and heterogeneity in order to exhibit realistic dynamical behavior. Modularity and extendibility are required to integrate new biological knowledge, easily.

Therefore, our architecture is implemented as an object-oriented system: All regulatory effects are fully encapsulated in one class. In order to introduce a new model of transcriptional regulation, only one class has to be rewritten. Moreover, it is possible to replace the implemented deterministic ODE approach by a stochastic simulation.

Our general strategy is to generate the artificial regulatory network in three steps: (1) generate the structure of the network (a hyper graph), (2) generate the functional behavior (an algebraic formula), and (3) generate the kinetic parameters.

The result is, both, a description of the network and a function f describing the dynamics of the artificial regulatory network, e.g., in terms of differential equations of the form $\dot{x} = f(\mathbf{x})$. Currently, the format of the network description follows the XML specification, and the graphics is produced in portable description format (PDF). The dynamical function is generated as source code for C, C++, and *Octave* [4].

2.1 Generation of the Network Structure

The generated artificial systems consist of two types of components: genes and substances. A *substance* is an *RNA*, a *protein*, or a *dimer*. Genes produce RNA. RNA produces proteins. RNA and proteins can react to form dimers. The production of RNA (transcriptional regulation) and proteins (translational regulation) can be regulated by RNA, proteins, and dimers. In the dynamical model, as described in the following section (Sec. 2.2), the concentration of RNA, proteins, and dimers can change. Genes are considered as persistent entities, which serve as *transcription units* for RNA. For each coding RNA a *translation unit* exists representing the translation of this RNA to its respective protein. Here, we assume that each gene codes for only one RNA. Two different genes can code for the same RNA. In this case, the transcriptional regulation can be different for each gene, but the translational regulation of the protein production of this RNA is the same. In our graphical representations (e.g., Fig. 1) the transcription of a gene (RNA production) and the translation of RNA (protein production) are subsumed in one node, called *production unit*.

The network structure is generated in the following way (for a detailed formal description see ref. [6]): (1) After creating RNAs and proteins more complex substances are created from these. For each gene, an RNA is chosen as its primary transcript and for some RNA a protein is chosen as product of the translation. It is possible for an RNA to have no product. The fraction of these *non-coding RNA* can be specified by the user. (2) For each gene, the in-degree is randomly selected and divided into two portions: in-degree for transcriptional regulation and translational regulation, respectively. (3) The algorithm finds appropriate regulator substances for each transcription and translation unit, respectively, according to the parameter settings specified by the user, e.g. in-degree and out-degree distribution.

This algorithm apparently depends on quite a large number of parameters: The number of genes N, the number of RNA n_{rna}, the number of proteins n_{prot}, the fraction of non-coding RNA α, the number of homo-dimers n_{hom}, the number of hetero-dimers n_{het}, the probability that two RNA form a dimer, the fraction of genes that self-regulate α_{self}, the fraction of external substances that are not gene products α_{ext}, the in-degree and out-degree distributions $K_{\mathrm{in,out}}$, the probability that an RNA acts as a regulator p_{rna}, and the fraction of transcriptional regulation α_{tkr}. Fortunately, the parameters have an intuitive biological meaning so that they can be chosen relatively easily, cf. [6]. The parameterization of the dynamical components will be more difficult, as we will see in the following section.

2.2 Dynamical Behavior

The dynamical behavior of the artificial gene expression network is described by a set of ordinary differential equations (ODEs) of the form $\dot{\mathbf{x}} = f(\mathbf{x}), \mathbf{x} \in \mathcal{R}^n, \mathbf{x} \geq \mathbf{0}$ (n: number of substance). There is one scalar state variable x_s for each substance s describing its current concentration. The ODE system is built according to the network structure as generated by the algorithm described in the previous section. There is a unique mapping from the network structure to the ODE system, the details of which are shown in ref. [6]. The general form of the equation for an RNA or protein s reads:

$$\frac{dx_s}{dt} = \sum_{\pi} \left(\frac{dx_s}{dt} \right)_{\pi} - \sum_{\langle s,q \rangle} \left(\frac{dx_s}{dt} \right)_{\langle s,q \rangle} + \left(\frac{dx_s}{dt} \right)_{\mathrm{ext}} - \left(\frac{dx_s}{dt} \right)_{\mathrm{decay}}. \tag{1}$$

That is, we assume that the change in the concentration of a substance is the sum of changes through production of this substance (represented by π) reduced by the sum of all changes through building dimers $\langle s, q \rangle$ with this substance and by decay $\left(\frac{dx_s}{dt}\right)_{\text{decay}}$. Additionally, there can be an external inflow or outflow $\left(\frac{dx_s}{dt}\right)_{\text{ext}}$. The equation for a dimer $\langle s, q \rangle$ follows analogously.

Equation (1) defines the framework to formulate the dynamical behavior of the system. To get the final ODE system, all terms in the equation have to be formulated as concrete kinetic descriptions of the corresponding processes. The decay terms, e.g., can be written as first order reactions $\left(\frac{dx_s}{dt}\right)_{\text{decay}} = \delta_s x_s(t)$ with a constant decay rate δ_s. In our approach, biological knowledge about the regulation of gene expression can be incorporated by defining the production terms for RNA and proteins. There are many approaches to model this regulatory process, e.g., by sigmoid functions of the weighted sum of the concentrations of the regulator substances [3], by S-systems [8], or by a switch-like mechanism [11]. We implemented two types of dynamics: **Type 1:** a sigmoid function of the weighted sum of inputs like in ref. [3], and **Type 2:** a generalized version of a switch-like mechanism described by ref. [11]. The latter allows to formulate boolean-like interactions with continuous, steep sigmoid functions.

Type 1 Dynamics: In this case, for the transcription of an RNA s, each of its production terms $\left(\frac{dx_s}{dt}\right)_{\pi}$ is defined as follows:

$$\left(\frac{dx_s}{dt}\right)_{\pi} = d_{\max} \times \text{sig}\left(b + \sum_{i=1}^{l} w_i \times x_i(t)\right). \tag{2}$$

Here, $x_1, \ldots, x_l$ are the concentrations of the regulators, $w_i \ldots, w_l$ the respective weights of the regulatory influence in the production π, $d_{\max}$ is the maximal transcription rate, b is the basal transcription rate, and the function sig is defined as a sigmoid:

$$\text{sig}(u) = \frac{1}{1 + e^{-u}} . \tag{3}$$

For the translation where an RNA m acts as a template to form a protein s, the production term is similar, but with an additional factor $x_m(t)$, which models the essential dependency of the translation process from the concentration of the translated RNA:

$$\left(\frac{dx_s}{dt}\right)_{\pi} = d_{max} \times x_m(t) \times \text{sig}\left(b + \sum_{i=1}^{l} w_i \times x_i(t)\right). \tag{4}$$

Type 2 Dynamics: For the production terms in Type 2 dynamics we filter each weighted input x_i by applying a sigmoid h_i:

$$x_i' = \begin{cases} h_i(w_i, x_i) & \text{if } w_i \geq 0, \\ 1 - h_i(w_i, x_i) & \text{otherwise,} \end{cases} \tag{5}$$

$$h_i(w_i, x_i) = \frac{|w_i| x_i^{q_i}}{|w_i| x_i^{q_i} + \phi_i^{q_i}} \tag{6}$$

where q_i and ϕ_i define the shape of the sigmoid. For suitable choices of q_i and ϕ_i the shape of the sigmoid is steep enough for the x_i' to behave like a $0 - 1$ (boolean) switch approximation

of the x_i. Using filtered inputs we can build arbitrary boolean-like production terms, where the basic boolean-like operators using the already filtered inputs x'_1, x'_2, are defined as follows:

$$\text{NOT}(x'_1) = 1 - x'_1, \tag{7}$$
$$\text{AND}(x'_1, x'_2) = x'_1 \times x'_2, \tag{8}$$
$$\text{OR}(x'_1, x'_2) = x'_1 + x'_2 - x'_1 \times x'_2. \tag{9}$$

This definition ensures that the result of applying an operator to filtered inputs, is again usable as filtered input for another operator.

2.3 Generation of Artificial Data

After creating the network and its dynamics in terms of an ODE system, the ODEs have to be numerically integrated in order to generate data. Two types of data are usually used in large scale analysis of the gene expression machinery: time series and single measurements after a defined time interval, e.g., at steady state. Both types of data are delivered by our system. In general, there are various ways for carrying out experiments. For example, *knockout experiments*, where we deactivate a gene; or *over-expression experiments*, where we amplify the production of a gene product. Technically, this kind of experiments can easily be performed with the generated artificial networks. It is just a matter of setting variables representing rate constants of RNA production to zero (knockout) or to higher values (over-expression).

3 Properties of the Generated Regulatory Systems

Given parameters for the number N of genes and for the distribution K of in-degrees (uniform or normal distribution), the algorithm delivers an artificial regulatory system where the graphical representation has N nodes and the distribution of the in-degrees follows K.

The algorithm delivers an artificial regulatory system, i.e., a set S of substances and a set 𝔊 of genes (the genome). It assigns a primary transcript to each gene (an RNA) and a protein to each coding RNA. For every transcription and translation process a set R ⊆ S of suitable regulators is computed.

There is a unique mapping from this regulatory system to a simplified graph (V, E), called *influence graph*, describing only regulatory influences among genes. The influence graph is the target of most gene network reconstruction methods and is defined as follows: There is an edge $(g_1, g_2) \in E$ between two genes $g_1, g_2 \in V$ iff there is a regulatory influence from gene g_1 to the transcription of gene g_2 or from g_1 to the translation of the primary transcript of gene g_2. An edge of this type signals that gene g_1 produces a regulatory substance that is involved in the transcription process of g_2 or in the translation process of the primary transcript of g_2. The regulatory substance can be a primary transcript (RNA), a protein, or a dimer.

In addition to the influence graph, our algorithm derives an ODE system from the regulatory system. There is one equation of the form Eq. (1) for each substance describing the evolution of its concentration over time. The regulatory substances of transcription and translation appear in the production terms described in Eq. (2) and Eq. (4), respectively.

3.1 Structural Properties

The algorithm that generates the network structure is a greedy algorithm. Therefore, it does not guarantee that all user settings (e.g., fraction of self-regulators) are met precisely. But an analysis of the in-degree and out-degree distribution of a large amount of randomly generated networks showed that even when the number of genes is small (e.g., $N < 4$), the expected deviation from the user settings is below 5% (see ref. [6] for detailed results).
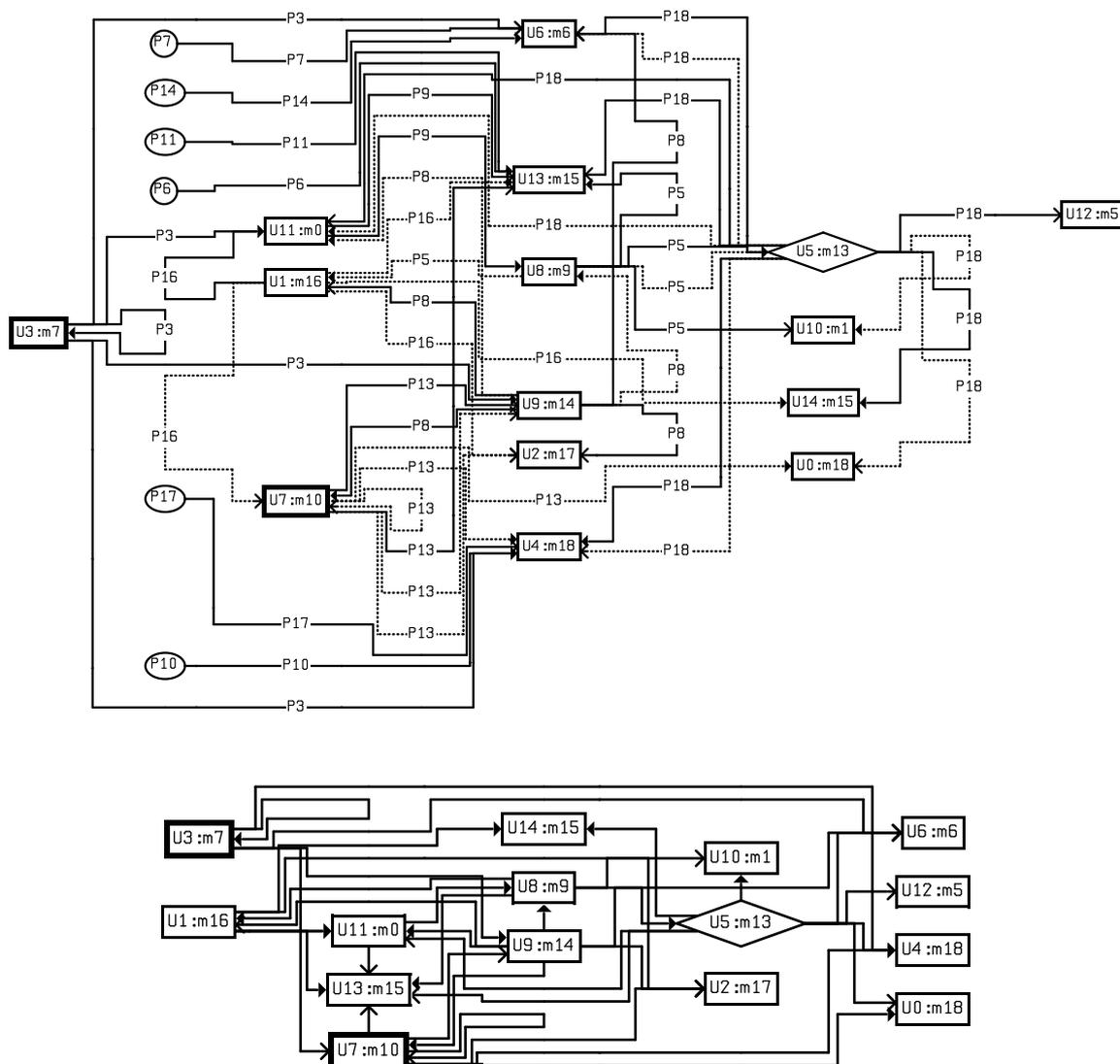

Figure 1: **Top:** Example of a generated network. The £gure gives an impression of the structure of a generated regulatory network with 15 genes. Genes are represented by squared shaped nodes (nodes with many outputs, hubs, are diamond shaped). Circular nodes represent external substances (proteins). A label like U3:m7 denotes the *production unit* U3, which produces RNA m7 *and* protein P3. Solid lines denote transcriptional regulation, dotted lines denote translational regulation. An edge is labeled with the respective regulator substance. **Bottom:** The corresponding in¤uence graph where only genes and their interactions are shown. Usually, the aim of a reconstruction method for regulatory networks is to £nd this kind of structure. Note that two different genes can produce the same RNA.

Figure 1 shows an example of a generated network that consists of 15 genes and the corresponding influence graph, which consists only of the genes and their interactions. Our system delivers the generated network and its influence graph in a format, which can be easily visualized using *xvcg* [14].

3.2 Dynamical Properties

In order to study the dynamical properties of our artificial gene networks, we generated networks for different parameter settings, simulated their behavior by numerically integrating their ODEs (using *lsode* [12]) and investigated the sensitivity of the networks to deletion and over-expression experiments.

Using Type 1 dynamics (sigmoid function of the weighted sum of inputs), we observed that the system almost always approached an asymptotically stable fixed point, where genes are expressed at different levels ranging over several magnitudes of order (Fig. 2). A similar behavior has been observed in other artificially created reaction networks, such as random catalytic networks [15] or binary string artificial chemistries [1].
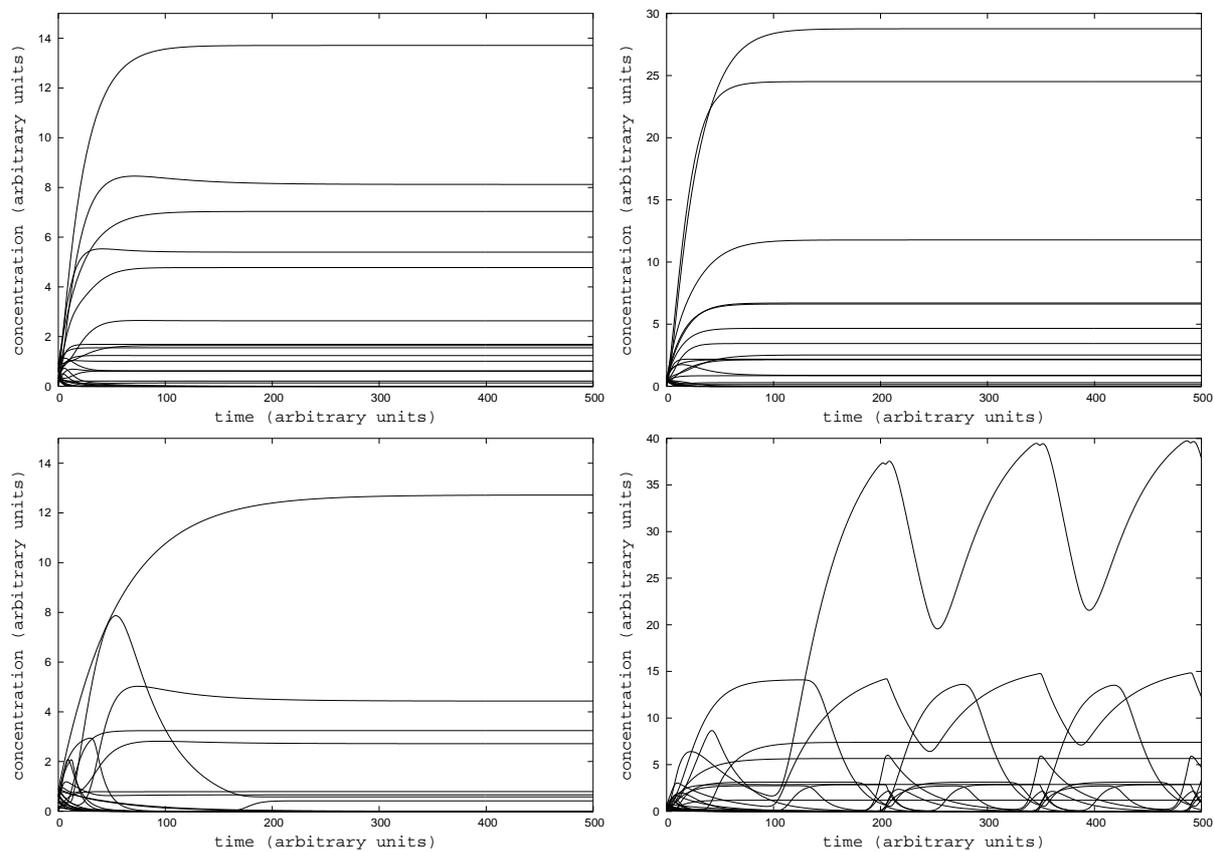

Figure 2: Examples of the dynamical behavior of Type 1 dynamics (**top**) and Type 2 dynamics (**bottom**). For all regulatory systems the same parameter settings (beside the definition of the dynamic type) were used. All systems consisted of 15 genes.

Using Type 2 dynamics (switch-like mechanism), about one third of the generated networks exhibited more complex, oscillatory dynamics as exemplified by Fig. 2. Furthermore, our experiments showed that Type 2 dynamics lead to networks that are more sensitive to perturbations, such as the deletion of a single gene as demonstrated by Fig. 3. Because of a lack of sensitivity

of Type 1 dynamics, Type 2 dynamics should be preferred when perturbation experiments are carried out.

3.3 Example: Deletion Experiment

To give an impression of how to produce an artificial regulatory system with our architecture and in order to show how to use this system to create artificial gene expression data, we now describe a deletion experiment. A deletion experiment can be conducted in five steps:

1. Define the parameters for generating the network structure and the kinetic description. This can be done by using our web-interface or — for fine-tuning and batch-jobs — via a parameter-file in XML.

2. Apply our generator (JAVA-program) to those parameters. It will output the artificial regulatory system and the description of the ODE system (as c- or octave-source). A typical resulting network structure is depicted in Fig. 1.

3. Give the ODE system to an ODE solver by compiling the c-source code and linking the result to the ODE solver's runtime system [12]. This can be done via the web-interface. Alternatively you can solve the ODE system within *octave* [4].

4. Define the experiment. For this purpose, the user provides a plain text-file, e.g., `experiment.dat`:

```
start 0
end 1200.0
inc 1
t 440 a_m0 0
t 1040 a_m0 1
```

The meaning of these five lines is: solve the ODE system from $t = 0$ to $t = 1200$ in steps of 1, with RNA $m0$ (and thus all genes producing this RNA) being deactivated from $t = 440$ until $t = 1040$.

5. Carry out the experiment:

```
solve -ts experiment.dat
```

where `solve` is the ODE solver program compiled in Step 3 and the parameter `-ts` signals that a time-series according to the definition in the file `experiment.dat` shall be computed. The results are saved in plain text-files and can be visualized as exemplified by Fig. 3.

In summary, the above procedure produces a regulatory network, its influence graph, graphical visualizations of both, an ODE system, an executable simulator for conducting experiments, and time series or steady state data.
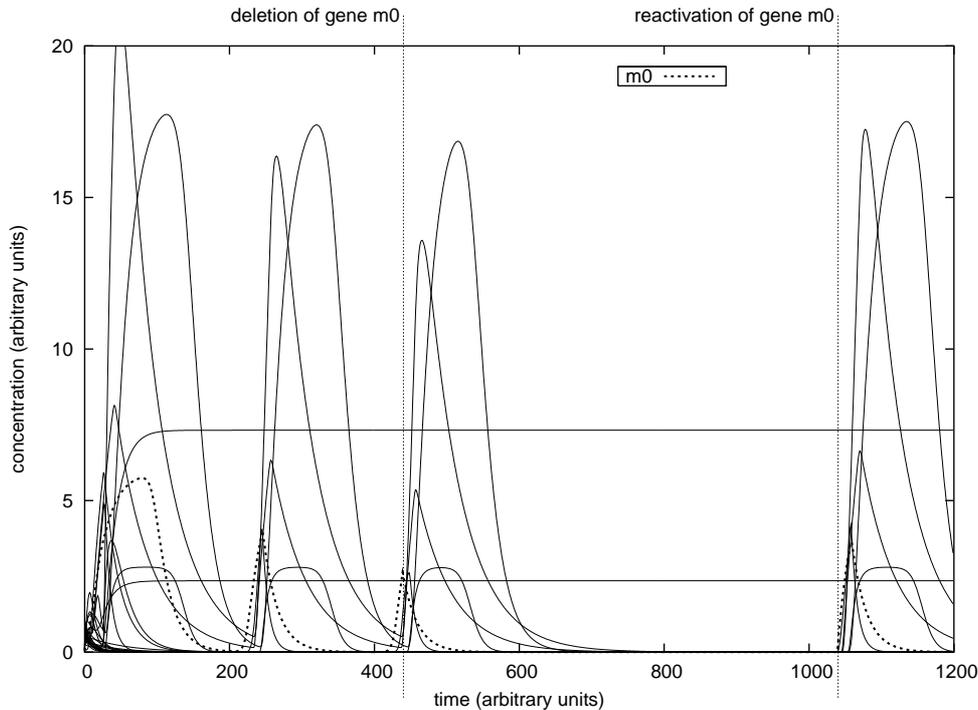
Figure 3: Example of oscillatory dynamics and a deletion experiment. At $t_0 = 440$ gene $m0$ is "deleted" by inhibiting the production of its RNA. Thus its concentration (denoted by the dashed line) vanishes slowly due to protein degradation and the dilution flux. At $t_1 = 1040$ the inhibition of the gene is released. (Type 2 dynamics)

4   Summary and Conclusion

We have presented a new system that is able to create complex artificial regulatory networks exhibiting high biological detail. Nevertheless, the underlying architecture is modular and extensible. The resulting regulatory networks are represented as computer programs, which can be executed in order to obtain concentration profiles of simulated gene activity. These data can provide a solid base for validation and evaluation of reverse bioengineering methods that aim at extracting information from gene expression data.

Our simulation results showed that it is not easy to randomly create a regulatory system that behaves like a biological system. We have carefully modeled many relevant details of the biological system, such as protein-RNA dimerization and post-transcriptional regulation, which usually are not considered in other systems for test data generation. Even if all single components are modeled accurately, the overall behavior of a randomly created network often does not exhibit the complexity of the natural system.

The basic structure of our model seems to be appropriate. But it remains an open question how to orchestrate the right interplay between the components in order to reach a nature-like dynamics. For that we have to find the structural principles behind this interplay. Two different approaches appear to be feasible for this endeavor: We can follow a more engineering-type approach by identifying conditional probability functions where the distribution for a rate-constant depends on other already selected rate-constants. Alternatively, we can imitate the natural evolutionary process that has lead to biological regulatory systems. Following this approach, artificial regulatory systems are evolved according to a fitness function that depends on the dynamical behavior of the networks. In our opinion, the latter approach shows the most promise for creating artificial test systems that can serve as appropriate representatives of natural regulatory systems.

References

[1] Wolfgang Banzhaf. Self-replicating sequences of binary numbers – foundations I and II: General and strings of length n = 4. *Biol. Cybern.*, 69:269–281, 1993.

[2] Ting Chen, Vladimir Filkov, and Steven S. Skiena. Identifying gene regulatory networks from experimental data. In *ACM-SIGACT The Third Annual International Conference on Computational Molecular Biology (RECOMB99)*, pages 94–103, 1999.

[3] Patrik D'haeseleer. *Reconstructing Gene Networks from Large Scale Gene Expression Data*. PhD thesis, The University of New Mexico, 2000.

[4] J.W. Eaton. *GNU Octave Manual*. Network Theory Limited, Bristol, UK, 2002.

[5] Trey E. Ideker, Vesteinn Thorsson, and Richard M. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. In *Proc. Paci£c Symp. Biocomputing*, pages 305–316, 2000.

[6] C. Knüpfer. Testdatengenerierung für die Inferenz von Genregulationsnetzen. Master's thesis, Friedrich-Schiller-Universität Jena, D-07743 Jena, 2003.

[7] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Proc. Paci£c Symp. Biocomputing*, volume 3, pages 18–29, 1998.

[8] Y. Maki, D. Tominaga, M. Okamoto, S. Watanabe, and Y. Eguchi. Development of a system for the inference of large scale genetic networks. In *Proc. Paci£c Symp. Biocomputing*, volume 6, pages 446–458, 2001.

[9] Dennis J. Michaud, Adam G. Marsh, and Prasad S. Dhurjati. eXPatGen: generating dynamic expression patterns for the systematic evaluation of analytical methods. *Bioinformatics*, 19(9):1140–1146, 2003.

[10] Dana Pe'er, Aviv Regev, Gal Elidan, and Nir Friedman. Inferring subnetworks from perturbed expression pro£les. *Bioinformatics*, 1(1):1–9, 2001.

[11] Erik Plahte, Thomas Mestl, and Stig W. Omholt. A methodological basis for description and analysis of systems with complex switch-like interactions. *Journal of Mathematical Biology*, 36(4):321–348, 1998.

[12] Krishnan Radhakrishnan and Alan C. Hindmarsh. Description and use of lsode, the livermore solver for ordinary differential equations. Technical report UCRL-ID-113855, Lawrence Livermore National Laboratory, 1993.

[13] Dirk Repsilber and Jan T. Kim. Developing and testing methods for microarray data analysis using an arti£cial life framework. In *Advances in Arti£cial Life - 7th European Conference (ECAL-2003)*, pages 686–695. Springer, Berlin, 2003.

[14] Georg Sander. VCG – visualization of compiler graphs. Technical Report A01-95, Universität des Saarlandes, FB 14 Informatik, 1995.

[15] Peter F. Stadler, Walter Fontana, and John H. Miller. Random catalytic reaction networks. *Physica D*, 63:378–392, 1993.

[16] D.C. Weaver, C.T. Workman, and G.D. Stormo. Modeling regulatory networks with weight matrices. In *Proc. Paci£c Symp. Biocomputing*, volume 4, pages 112–123, 1999.

[17] C. Yoo, V. Thorsson, and G.F. Cooper. Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational DNA microarray data. In *Proc. Paci£c Symp. Biocomputing*, volume 7, pages 498–509, 2002.