Qualitative Analysis of Regulatory Graphs: A Computational Tool Based on a Discrete Formal Framework

Claudine Chaouiya[1], Elisabeth Remy[2], Brigitte Mossé[2], and Denis Thieffry[1]

[1] LGPD-IBDM, Campus de Luminy, Case 907, 13288 Marseille Cedex 9, France, {chaouiya, thieffry}@ibdm.univ-mrs.fr
[2] IML, Campus de Luminy, Case 907, 13288 Marseille Cedex 9, France, {remy, mosse}@iml.univ-mrs.fr

Abstract. Building upon the logical approach developed by the group of R. Thomas in Brussels, we are defining a rigorous mathematical framework to model genetic regulatory graphs. Referring to discrete mathematics and graph-theoretic notions, our formal approach supports the development of a software suite in Java, GIN-sim, which allows the qualitative simulation and the analysis of the dynamics of regulatory graphs, under either synchronous or asynchronous updating assumptions.

1 Introduction

Our formal approach roots in the logical formalism previously developed by R. Thomas and colleagues [5, 6]. Combining graph-theoretic and discrete mathematical notions, we propose a series of definitions enabling a proper mathematical description of genetic regulatory graphs, as well as of the corresponding qualitative dynamical behaviour (Sections 2 and 3) (see [2] for a recent review of this field). This formal framework serves as a basis for the study of formal properties of regulatory graphs (Section 3.4), as well as for the development of a simulation software, GIN-sim (Section 4). [3]

2 Regulatory graphs

2.1 Definitions

A **regulatory graph** is a labeled graph where vertices represent genes, whereas edges represent interactions; when oriented (*e.g.* transcriptional regulation), an interaction is represented by an **arc**, possibly signed (positively for

[3] We thank H. de Jong for his suggestions concerning a previous version of this manuscript. We further acknowledge the financial support of the *French Action inter-EPST bioinformatique*.

L. Benvenuti, A. De Santis, and L. Farina (Eds.): Positive Systems, LNCIS 294, pp. 119-126, 2003.

an activation, negatively for a repression). Note that we mainly refer to interactions between genes, though these interactions may involve various types of molecular mechanisms. On each arc, a label indicates the conditions under which the interaction is functional, together with the sign of the interaction. Finally, we consider the following data:

- A finite set $\mathcal{G} = \{g_1, \ldots, g_n\}$ constituted by n elements, called **genes**.
- A set of positive integers $\{max_1, \ldots, max_n\}$, where, for each i, max_i is the **maximum expression level** of gene g_i. Therefore, the different expression levels allowed for g_i are the integers $\{0, \ldots, max_i\}$.
- A labeled oriented graph $\mathcal{R} = (\mathcal{G}, \mathcal{L})$, where $\mathcal{G}$ is the set of vertices (genes) and $\mathcal{L}$ is the set of arcs, which represent interactions between genes. A label (A, q) is associated to each arc, specifying the conditions under which the interaction takes place, and the nature of this interaction:
  i) A is an integer interval included in $\{1, \ldots, max_i\}$. If several arcs join g_i to g_j, then the different intervals are mutually disjoined.
  ii) $q \in \{-1, 0, 1\}$ is the **sign of the interaction**, denoting an **activation** $(q = +1)$, an **inhibition** $(q = -1)$, or undetermination$(q = 0)$.

Interaction from g_i (source) to g_j (target) is a tuple $T = (g_i, g_j, A(T), q(T))$ where $(A(T), q(T))$ is the label of the arc from g_i to g_j. Interval $A(T) = [s_{inf}(T), s_{sup}(T)]$, with $s_{inf}(T) > 0$, is the set of consecutive expression levels of g_i for which T is **functional**. Integer $q(T)$ is the sign of the interaction.

Interactions are subjected to the following conditions:

For any g_i in $\mathcal{G}$, for any l in $\{1, \ldots, max_i\}$, there exists an interaction T with source g_i such that $l = s_{inf}(T)$; consequently, any non trivial expression level of gene g_i corresponds to a threshold from which an interaction (with source g_i) becomes functional (thus for each gene, the maximum level equals at most the number of interactions exerted by this gene).

Let $\mathcal{I}_j$ be the set of incoming interactions (or inputs set) of g_j. For any gene g_j, a subset X of $\mathcal{I}_j$ is **admissible** if it does not contain interactions having the same source.

When expression levels of the genes are given, we know which interactions are functional, and we would like to describe their action. This is done by means of logical parameters:

- for any gene g_j, the application K_j, called **logical function for gene** g_j, associates an integer $K_j(X)$ $(0 \leq K_j(X) \leq max_j)$ to any admissible subset X of $\mathcal{I}_j$. This integer is called **logical parameter** $K_j(X)$ and corresponds to the expression level to which gene g_j tends, when the set of functional incoming interactions is equal to X.

Remark 1. For a gene g_j, absence of inhibition can lead to increase its level of expression of g_j, and consequently, parameter $K_j(\emptyset)$ may be greater than zero.

Remark 2. When X is not an admissible subset of $\mathcal{I}_j$, $K_j(X) = 0$.

2.2 A toy example

As a toy example, we consider the regulatory graph defined by $\mathcal{G} = \{a, b, c\}$, $max_a = 2$, $max_b = max_c = 1$, and by the labeled graph $\mathcal{R}$ (see Figure 1). Gene a is a dual regulator of gene b (it activates or inhibits b depending on the context); gene b activates itself and gene c; finally, gene c inhibits gene a.

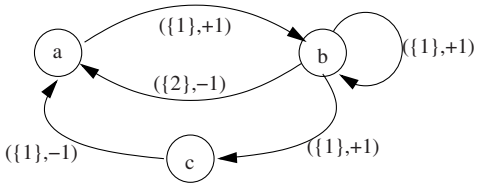

Fig. 1. The regulatory graph of the toy example.

There are five interactions: $T_1^a = (a, b, \{1\}, +1)$, $T_2^a = (a, b, \{2\}, -1)$, $T_1^b = (b, b, \{1\}, +1)$, $T_2^b = (b, c, \{1\}, +1)$ and $T_1^c = (c, a, \{1\}, -1)$. The logical parameters are given by Table 1.

Inputs sets are : $\mathcal{I}_a = \{T_1^c\}$, $\mathcal{I}_b = \{T_1^a, T_2^a, T_1^b\}$ and $\mathcal{I}_c = \{T_2^b\}$.

Table 1. The logical parameters of the toy example.

K_a :		K_b :		K_c :	
$\emptyset$	$\mapsto 2$	$\emptyset$	$\mapsto 0$	$\emptyset$	$\mapsto 0$
$\{T_1^c\}$	$\mapsto 0$	$\{T_1^a\}$	$\mapsto 1$	$\{T_2^b\}$	$\mapsto 1$
		$\{T_2^a\}$	$\mapsto 0$		
		$\{T_1^b\}$	$\mapsto 1$		
		$\{T_1^a, T_1^b\}$	$\mapsto 1$		
		$\{T_2^a, T_1^b\}$	$\mapsto 0$		

3 Dynamical graphs

Consider $((\mathcal{G}, \mathcal{L}), (K_j)_{1 \leq j \leq n})$ a regulatory graph. To characterise the dynamics of the system, we have to address the following question: given an initial state $x^0 = (x_1^0, \ldots, x_n^0)$ (where x_i^0 is the initial expression level of gene g_i), what are the following consecutive states (possibly) reached by the system? Let us denote by $\mathcal{E}$ the set of all possible states:

$$\mathcal{E} = \{x = (x_1, \ldots, x_n); \forall i = 1, \ldots, n,\ 0 \leq x_i \leq max_i\}\,. \tag{1}$$

For any given state x and for a gene g_j, we call $\mathcal{I}_j(x)$ the set of all incoming interactions which are functional at state x. It is an admissible set defined by:

$$\mathcal{I}_j(x) = \{U = (g_i,\, g_j,\, A(U),\, q(U)) \in \mathcal{I}_j\,;\; x_i \in A(U)\}\,. \tag{2}$$

A tuple $\mathcal{I}(x) = (\mathcal{I}_1(x), \ldots, \mathcal{I}_n(x))$, called **instruction** at state x, defines for each gene which incoming interactions are functional. Using applications $(K_j)_{1\leq j\leq n}$, we obtain all the values of the parameters which describe the evolution of the system. In order to represent the discrete dynamics of the system, we define a **dynamical graph**, where vertices represent states, each labeled by a tuple of n integers representing the actual levels of the genes. In our toy example, in state $x = (2, 1, 0)$ gene a is at its maximum level ($x_a = 2$), as well as b ($x_b = 1$), while c has no signifiant expression ($x_c = 0$).

In dynamical graphs, arcs represent spontaneous transitions between pairs of states. For instance, an arc between $x^0 = (0, 0, 0)$ and $x^1 = (1, 0, 0)$ corresponds to a transition from x^0 to x^1, as a consequence of the definition of the corresponding parameters $K_a(\mathcal{I}_a(x^0))$, $K_b(\mathcal{I}_b(x^0))$ and $K_c(\mathcal{I}_c(x^0))$. We have still to define an updating method to specify the temporal ordering of the transitions. We successively consider a fully synchronous versus a fully asynchronous assumptions.

3.1 Synchronous dynamical graphs

Under the synchronous assumption, at each time step, all update orders (*i.e.* calls for changes of expression level for a subset of genes at a given state) are executed simultaneously. As a result, each state has exactly one successor. From a biological point of view, this frequently used assumption implies that all macromolecular processes are realised in identical amounts of times (or "delays"), which is clearly unrealistic and often at the origin of simulation artefacts.

We denote by $\xi_s = (\mathcal{E}, \mathcal{F}_s)$ the **synchronous graph**, where $\mathcal{E}$ is defined by (1), and $\mathcal{F}_s$ is the set of arcs defined as follows. There exists a unique arc from x to $y \in \mathcal{E}$ defined by $y = (y_1, \ldots y_n)$ and for all $j \in \{1, \ldots n\}$:

$$y_j = \begin{cases} x_j & \text{if } K_j(\mathcal{I}_j(x)) = x_j, \\ x_j - 1 & \text{if } K_j(\mathcal{I}_j(x)) < x_j, \\ x_j + 1 & \text{if } K_j(\mathcal{I}_j(x)) > x_j. \end{cases} \tag{3}$$

In other words, the dynamical synchronous graph corresponds to an application of $\mathcal{E}$ on itself, which associates to a state x a unique state y obtained by a simultaneous update of all coordinates of x, following instruction $\mathcal{I}(x)$.

Remark 3. Our definition forbids jumping over integer values, something which may occur when using the simple definition $y_j = K_j(\mathcal{I}_j(x))$ in a multi-level context.

Given a set of initial states, a sub-graph corresponding to a particular pathway can be extracted from ξ_s. We denote by $\xi_s(x^0)$ the sub-graph which represents the pathway of the system when initial state is x^0, under a synchronous updating (note that $\xi_s(\mathcal{E}) = \xi_s$).

3.2 Asynchronous dynamical graphs

Under the asynchronous assumption, when multiple update orders occur at a given state, additional information is needed to select a specific transition (*i.e.* the values of relevant time delays or some ordering relationships). Here, specific time-delays are associated to each reaction (synthesis, degradation, activation, inhibition). As we have no information about these time delays, all possible transitions are generated. As a consequence, each state x has a number of successors equals to the number of update orders in this state.

Let us denote by $\xi_a = (\mathcal{E}, \mathcal{F}_a)$ the **asynchronous dynamical graph**, where the set of vertices is $\mathcal{E}$, and $\mathcal{F}_a$ is the set of arcs. Let x be a state; $\forall j \in \{1, \ldots, n\}$ such that $K_j(\mathcal{I}_j(x)) \neq x_j$, there exists an arc between x and

$$y = \begin{cases} (x_1, \ldots, x_{j-1}, x_j - 1, x_{j+1}, \ldots, x_n) & \text{if } K_j(\mathcal{I}_j(x)) < x_j, \\ (x_1, \ldots, x_{j-1}, x_j + 1, x_{j+1}, \ldots, x_n) & \text{if } K_j(\mathcal{I}_j(x)) > x_j. \end{cases} \quad (4)$$

Therefore, two linked states x and y differ by at most one coordinate. Moreover, in an asynchronous graph, an arc represents a unique update order.

We denote by $\xi_a(x^0)$ the sub-graph of ξ_a which represents all possible pathways when initial state is x^0, under an asynchronous updating.

3.3 Illustration through our toy example

The example of Section 2.2 is small enough to enumerate all possible states with the corresponding instructions and parameters values (Table 2).

Table 2. States and corresponding instructions

States x	$\mathcal{I}_a(x)$	$\mathcal{I}_b(x)$	$\mathcal{I}_c(x)$	$K_a(\mathcal{I}_a(x))$	$K_b(\mathcal{I}_b(x))$	$K_c(\mathcal{I}_c(x))$
$(0,0,0)$	$\emptyset$	$\emptyset$	$\emptyset$	2	0	0
$(0,0,1)$	$\{T_1^c\}$	$\emptyset$	$\emptyset$	0	0	0
$(0,1,0)$	$\emptyset$	$\{T_1^b\}$	$\{T_2^b\}$	2	1	1
$(0,1,1)$	$\{T_1^c\}$	$\{T_1^b\}$	$\{T_2^b\}$	0	1	1
$(1,0,0)$	$\emptyset$	$\{T_1^a\}$	$\emptyset$	2	1	0
$(1,0,1)$	$\{T_1^c\}$	$\{T_1^a\}$	$\emptyset$	0	1	0
$(1,1,0)$	$\emptyset$	$\{T_1^a, T_1^b\}$	$\{T_2^b\}$	2	1	1
$(1,1,1)$	$\{T_1^c\}$	$\{T_1^a, T_1^b\}$	$\{T_2^b\}$	0	1	1
$(2,0,0)$	$\emptyset$	$\{T_2^a\}$	$\emptyset$	2	0	0
$(2,0,1)$	$\{T_1^c\}$	$\{T_2^a\}$	$\emptyset$	0	0	0
$(2,1,0)$	$\emptyset$	$\{T_2^a, T_1^b\}$	$\{T_2^b\}$	2	0	1
$(2\ 1\ 1)$	$\{T_1^c\}$	$\{T_2^a\ T_1^b\}$	$\{T_2^b\}$	0	0	1

Using states and corresponding instructions in Table 2, for an initial state, *e.g.* $x^0 = (0,0,0)$, we can generate the dynamical pathway(s) of the system. Figure

2(A) illustrates the synchronous dynamical sub-graph $\xi_s((0,0,0))$, leading to a 3-states cycle. Figure 2(B) illustrates $\xi_a((0,0,0))$, leading to two alternative stable states. Recall that in asynchronous graphs, all possible updates are represented. Superscripts (+ or -) indicate whether the instruction tends to increase or decrease the level of expression of a gene. Absence of superscript denotes a stationary level.

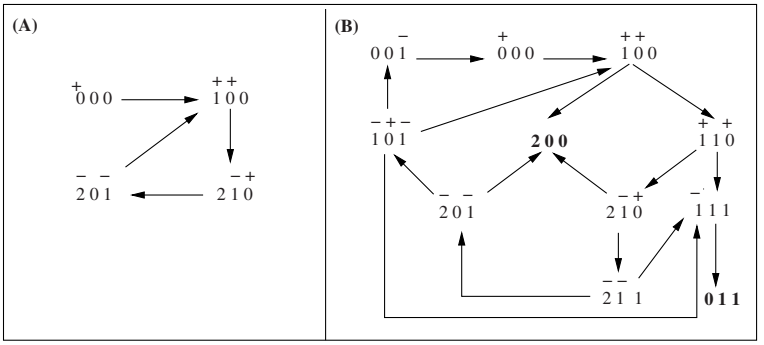

Fig. 2. (A) Synchronous (B) asynchronous dynamical sub-graphs for the toy example. Note that loops are omitted on terminal nodes.

3.4 Dynamical properties

A state $x = (x_1, \ldots, x_n)$ is **stationary** if, for all $i = 1, \ldots, n$, $x_i = K_i(\mathcal{I}_i(x))$. Using previous definitions and Remark 2, it is possible to show that stationary states satisfy the following equations: for $j = 1, \ldots, n$,

$$x_j = \sum_{X \subset \mathcal{I}_j} K_j(X) \left[\prod_{U \in X} \mathbf{1}_{A(U)}(x_i) \prod_{U \in \mathcal{I}_j \setminus X} (1 - \mathbf{1}_{A(U)}(x_i)) \right] ,$$

where U stands for interactions (g_i, g_j, $A(U)$, $q(U)$) and $\mathbf{1}_A$ denotes the characteristic function of set A, *i.e.* $\mathbf{1}_A(x) = 1$ if $x \in A$, $\mathbf{1}_A(x) = 0$ otherwise.

Stationary (stable) states are easily identified as final vertices. In our toy example, states $(2,0,0)$ and $(0,1,1)$ are stationary in the asynchronous sub-graph $\xi_a((0,0,0))$ (Figure 2(B)). The synchronous graph of Figure 2(A) presents no stationary state, but contains a **dynamical cycle**. Note that the notion of stationary state is independant of the updating method. Nevertheless, the choice of a specific updating method can considerably change the connectivity between the states (compare Figure 2(A) and (B)).

Stationary states or dynamical cycles correspond to the notion of **attractors** in the field of dynamical systems, though dynamical cycles may be followed only transiently. More generally, attractors are related to strongly connected components of dynamical graphs. Using the same analogy, the notion

of **basin of attraction** of an attractor is encompassed by the set of vertices having a path to a given strongly connected component (or "attractor").

Having defined a regulatory graph, we focus on the circuits of the graph. When the signs of the interactions are determined, a circuit is said to be **positive** if the product of these signs is positive, **negative** otherwise. These circuits can generate **differentiative** (positive circuits) or **homeostatic** (negative circuits) properties [5]. Forming strongly connected components of the regulatory graph, intertwined circuits can be related to the biological notion of cross-regulatory modules.

With this mathematical framework, we aim at establishing formal links between regulatory graphs and the corresponding dynamical graphs. We have already precisely defined the structure of the dynamical graphs corresponding to elementary regulatory circuits. This structure depends only on the sign and length of the circuits. Furthermore, the complex structure of ξ_a can be simply described on the basis of the simpler structure of ξ_s.

4 GIN-sim

From a computational perspective, our approach takes the form of a series of Java classes, collectively called GIN-sim. This simulation tool is part of a wider software project, which provides a series of modules covering the integration, the processing, and the modelling of functional regulatory data [1]. In GIN-sim, both synchronous and asynchronous simulations have been implemented. Graphical interfaces are currently under development, as well as algorithms to exhibit structural properties of both regulatory and dynamical graphs.

Given a set of initial states, GIN-sim generates a dynamical graph, qualitatively representing all allowed spontaneous state transitions corresponding to the model encoded in the original regulatory graph. The initial states and the parameter values can be defined by the user or by default (including the number of distinct levels for each regulatory product, and the qualitative weights of the different combinations of interactions on each gene). The user can progressively refine his model, depending on simulation results.

Given a regulatory graph, a set of parameters, and a set of initial states, our simulation algorithm is essentially a variant of the standard depth-first traversal algorithm. For each current state, relevant parameters are determined to generate the successor(s) of this vertex.

5 Discussion and conclusion

Leaning on the logical method previously developed by R. Thomas [5], we have introduced a rigorous, discrete, dynamical formalisation of genetic regulatory graphs. The originality of our approach lies in : (1) the coverage of multi-arcs in regulatory graphs (labelled by non overlaping intervals); (2) a

generic representation of all kinds of logical relationships when multiple interactions are exerted on a given gene. Note that the corresponding logical parameter values constraint the signs attached to the interactions involved. In other words, the determination of the sign of an interaction (+ or -) imposes inequalities on relevant parameters to insure consistency.

This discrete mathematical framework opens the way to a systematic analytical study of the link between regulatory and dynamical graphs as well as between synchronous and asynchronous dynamical graphs. GIN-sim implements this formal framework and allows the validation of analytic results, as well as biological applications. Up to now, this approach has been applied to the dynamical modelling of the networks involved in the control of the cell cycle, cell differentiation, and pattern formation during *Drosophila melanogaster* embryonic development (see *e.g.* [4]).

As the number of genes and interactions of regulatory graphs increases, the size of the corresponding dynamical graphs may grow exponentially. However, there are at least three ways to cope with this problem: (1) using features of genetic regulatory networks such as modularity and limited values for in/out degrees of vertices; (2) focusing on relevant part of dynamical graphs (partial exploration); (3) exploiting analytical results, for example concerning the role of feedback circuits [6] or the location of all stationary states [3].

Other analytical tools are available for the modelling of regulatory graphs [2]. Often complementary, these approaches should be combined to cope with the complexity and the variety of biological networks.

References

1. Chaouiya C., Sabatier C., Verheecke-Mauz C, Jacq B. and Thieffry D. (2002): *GIN-tools: Towards a software suite for the integration, the analysis, and the simulation of Gene Interaction Networks.*, Proceedings of JOBIM 2002. Saint-Malo, France, June 2002, pp. 17-26.
2. de Jong H.(2001): *Modeling and simulation of genetic regulatory systems: A literature review*, J. Comp. Biol. 9, pp.69-105.
3. Devloo V., Hansen P. and Labbé M. (2003): *Identification of all steady states in large biological systems by logical analysis*, Bull. Math. Biol., in revision.
4. Sánchez, L. and Thieffry D.(2001): *A logical analysis of the gap gene system*, J. theor. Biol. 211: 115-141.
5. Thomas R. (1991): *Regulatory networks seen as asynchronous automata: a logical description*, J. theor. Biol. 153:pp. 1-23.
6. Thomas R, Thieffry D, Kaufman M. (1995): *Dynamical behaviour of biological regulatory networks, I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state*. Bull. Math. Biol. 57, pp.247-276.