HAL
open science

Qualitative modelling and analysis of regulations in multi-cellular systems using Petri nets and topological collections

Jean-Louis Giavitto, Hanna Klaudel, Franck Pommereau

To cite this version:

Jean-Louis Giavitto, Hanna Klaudel, Franck Pommereau. Qualitative modelling and analysis of regulations in multi-cellular systems using Petri nets and topological collections. 4th Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2010), Aug 2010, Jena, Germany. 10.4204/EPTCS.40.11 . hal-00666662

HAL Id: hal-00666662

https://hal.science/hal-00666662v1

Submitted on 10 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Qualitative modelling and analysis of regulations in multi-cellular systems using Petri nets and topological collections

Jean-Louis Giavitto Hanna Klaudel Franck Pommereau
IBISC, University of Évry
Tour Évry 2, 523 place des terrasses de l'Agora, 91000 Évry, France
{giavitto,klaudel,pommereau}@ibisc.univ-evry.fr

In this paper, we aim at modelling and analyzing the regulation processes in multi-cellular biological systems, in particular tissues. The modelling framework is based on interconnected logical regulatory networks (*à la* René Thomas) equipped with information about their spatial relationships. The semantics of such models is expressed through colored Petri nets to implement regulation rules, combined with topological collections to implement the spatial information. Some constraints are put on the the representation of spatial information in order to preserve the possibility of an enumerative and exhaustive state space exploration. This paper presents the modelling framework, its semantics, as well as a prototype implementation that allowed preliminary experimentation on some applications.

1 Introduction

Regulation processes are the corner stone to understand many aspects of biological systems. Regulations occur at many levels: transcription and translation of the genetic material, protein modifications, etc. They define complex networks of interactions that cannot be easily understood without resorting to formal modelling and automated analysis. The generalized logical formalism initially proposed by René Thomas in the 70s [25, 26, 27], is a discrete modelling formalism that has proved to be an effective way to capture regulation processes and analyze them. It has been successfully applied to the study of a variety of regulatory networks comprising relatively large numbers of components [20, 21]. This formalism however does not provide any modelling device to specifically address the question of multi-cellular systems, where the regulatory networks of cells can influence each other in a way that is dependent on the spatial relationships between the cells.

This paper is a step toward providing a modelling framework for the regulation in multi-cellular systems, in particular tissues, taking into account cells migration, division and apoptosis (death). This framework will be applied to the analysis of systems such as developmental processes, invasive cancers, plant growth, etc.

In such a modelling framework, we would like to preserve the ability to perform model-checking based analysis in order to be able to assess causality-related properties and observe rare events, which usually cannot be obtained through simulation. This constrains the possible choices for modelling spatial information. In particular, their should be a finite number of possible spatial evolutions from a given configuration and they should be enumerable. Moreover, each spatial configuration should be represented in a normalized form, allowing the recognition of two identical configurations. For instance, floating-point positions are not a possible solution,

G.Ciobanu, M.Koutny (Eds.): Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2010)
EPTCS Proceedings, 2010.
EPTCS ??, 20??, pp. 1–16, doi:10.4204/EPTCS.??.??

© J.-L. Giavitto, H. Klaudel & F. Pommereau

instead, we shall use solutions based on discrete representations. Finally, we would like to be able to obtain easily a graphical representation of a cell population, *i.e.*, the chosen representation of spatial information should be compatible with existing graphical rendering techniques.

Contributions. Our starting point is a modelling approach developed in [2, 6] and extended later to introduce modularity [5, 4]. This approach itself is based on logical regulatory networks [25, 26, 27] that allow for modelling regulation within a biological system. A Petri net semantics is provided in order to analyze the various properties of so modelled systems. In [5, 4], modularity allows an easier modelling of multi-cellular systems, each cell being represented by a module. However, the spatial relationship between the modules within a model is represented in a very abstract way, with no link to any kind of geometrical or topological information.

Our main contribution in this paper is twofold. First, we propose a way to specify such information in a very flexible way, without significantly increasing the complexity of the original framework. Second, we define devices to model efficiently the spatial transformations related to the apoptosis, migration and division processes. The former goal is achieved by decoupling the regulation rules in the model from the spatial information and transformations. Both aspects being linked through a standardized interface, allowing the modeler for using various approaches to spatial representation. The latter goal is achieved by a careful design of this interface in order to enable a smooth bi-directional communication between the two parts of a model.

Then, we present two approaches to the modelling of spatial information. One is based on predefined grids and another is based on bounded-degree graphs. Both methods have pros and cons, depending on the application domain of interest. Finally, our framework is given a Petri net semantics that is implemented in a prototype, allowing to prove the feasibility of the approach and to run preliminary experiments on simple applications.

Notice that the approach presented in this paper is applicable only if one sort of module is considered at the same time, *i.e.*, when all the cells in a tissue are of the same kind. The extension to take into account multi-sorted systems is quite straightforward, but the resulting notations are much more complex. Our prototype implementation actually does not have any such limitation. However, for this paper, an intuitive and simpler presentation has been preferred.

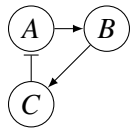
Outlines. The next section introduces the background of our work, in particular the modelling framework we start from. Our contributions are then presented in sections 3 and 4, the former being dedicated to the modelling of spatial information while the latter presents the extended modelling framework. The paper ends with a conclusion and a discussion about future works.

2 Background

2.1 Logical regulatory networks

A logical *regulatory network* is usually depicted as a graph whose nodes are *regulatory components*, for instance genes or proteins, and whose arcs indicate how each component is influenced by others. A simple regulatory network in depicted in the left part of Fig. 1.

More formally, a regulatory network is defined as a set $\mathscr{G}$ of regulatory components, each component $G \in \mathscr{G}$ being associated with a *regulatory function* K_G that provides the following information:



$$K_A : x_C \to \{0,1\} \stackrel{\text{df}}{=} 1 - x_C$$
$$K_B : x_A \to \{0,1\} \stackrel{\text{df}}{=} x_A$$
$$K_C : x_B \to \{0,1\} \stackrel{\text{df}}{=} x_B$$

Figure 1: A simple regulatory network with three components, A, B and C, such that A is an activator for B, B is an activator for C, and C is an inhibitor for A.

- its co-domain defines the range of values G can assume, the current value of G being denoted as x_G;
- its arguments define the regulatory components G depends on;
- its evaluation defines the level toward which G is called to evolve, by steps of ± 1.

For instance, the right part of Fig. 1 provides the formal definition of the regulatory network depicted in the left part. This example is a Boolean network in which all the components may have values in $\{0,1\}$. In order to model systems where different thresholds have different influences, more values may be used for component ranges.

A *state* of a regulatory network is a $\mathscr{G}$-indexed vector that provides for each component $G \in \mathscr{G}$ its current level x_G. Given a state s, and a component G, it is possible to evaluate K_G, yielding a target value x'_G for G. If $x'_G \neq x_G$, this defines a possible *evolution* of the system to a state s' that is such that $s'[H] \stackrel{\text{df}}{=} s[H]$ for all $H \in \mathscr{G} \setminus \{G\}$ and:

$$s'[G] \stackrel{\text{df}}{=} x_G \triangleright x'_G \stackrel{\text{df}}{=} \begin{cases} x_G + 1 & \text{if } x'_G > x_G \ , \\ x_G - 1 & \text{if } x'_G < x_G \ , \\ x_G & \text{otherwise.} \end{cases}$$

(Notation $\triangleright$ will be useful later on to define the Petri nets semantics.) Such an evolution is denoted by $s \to s'$. Given an initial state s_0 of a regulatory network, it is then possible to define the *reachable state space* as the smallest graph such that s_0 is a node and, whenever s is a node and $s \to s'$, then s' is also a node and there is an arc from s toward s'. Such a graph defines a *transition system* that is suitable to perform model-checking of various reachability- or causality-related properties.

For instance, consider state $[1,0,0]$ of the previous network (indexed as $[x_A, x_B, x_C]$). Only component B may evolve, because $K_B(x_A) \mapsto 1$, $K_A(x_C) \mapsto 1$ and $K_C(x_B) \mapsto 0$ in state $[1,0,0]$; this yields a new state $[1,1,0]$. The state space reachable from $[1,1,0]$ is depicted in Fig. 2.

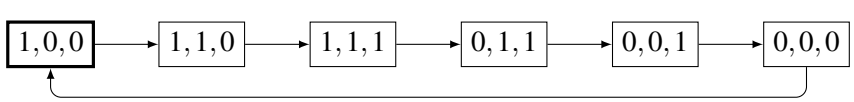
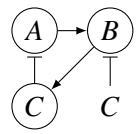


Figure 2: The state space of the regulatory network from figure 1 reachable from state $[x_A, x_B, x_C] = [1,0,0]$ (depicted in bold).



$$K_A : x_C \to \{0,1\} \stackrel{\text{df}}{=} 1 - x_C$$
$$K_B : x_A, \sigma_{C,B} \to \{0,1\} \stackrel{\text{df}}{=} \mathsf{max}(0, x_A - \sigma_{C,B}),$$
$$K_C : x_B \to \{0,1\} \stackrel{\text{df}}{=} x_B$$

Figure 3: A regulatory module based on the regulatory network of figure 1.

2.2 Modular extension

In [5, 4], a modular extension of logical regulatory networks is introduced in order to model regulatory networks encompassing several cells, each cell being modelled as a *regulatory module*. This extension consists of two parts:

- a regulatory module is defined as a regulatory network equipped with *inputs* and whose regulatory functions are adapted in order to take inputs into account;
- identified regulatory modules can then be connected, their spatial relationships being defined by a map δ associating a real value in the segment $[0;1]$ to every pair of module identifiers, where value 0 represents no influence of one module over another, and 1 represents a maximal influence. We call this value the *neighboring* between two modules, and the connected modules form a *regulatory bundle*.

For instance, Fig. 3 depicts a regulatory module based on the previous example. It specifies that component B is repressed by the presence of C outside the module. The uncircled node thus denotes an input of the module. Notice that no output needs to be specified since any component can be an input for any neighbor module. The regulation function K_B is adjusted in order to take into account the additional information about external C. It thus takes a new argument $\sigma_{C,B}$, called an *integration function*, whose unique argument is a set of pairs $(j, x_{C,j})$, where j is the identifier of a neighbor module and $x_{C,j}$ is the level of C in module j. Intuitively, for every module at non-zero neighboring to the current module, such a pair belongs to the argument of $\sigma_{C,B}$. This function is then responsible for computing a unique value within the admissible range of C, that integrates all the information from the neighborhood. The rationale behind the use of a $[0;1]$ map to define the neighboring is to take into account a notion of distance. For instance, diffusion of chemicals depends on a notion of distance and cannot be easily handled through the graph of neighborhood: two cells i and j can be in contact with a cell k, but because of their different sizes, the time used by a chemical diffusing from i to k will be different from the time taken for a chemical diffusing from j to k.

In the example, K_B evaluates to 0 when $x_A = 0$ or $\sigma_{C,B} \mapsto 1$ and to 1 otherwise. Function $\sigma_{C,B}$ has not been specified, for instance, it may be defined as

$$\sigma_{C,B} : (j, x_{C,j})_{j \geq 1} \mapsto \max_{j \geq 1}(\mathsf{round}(\delta(i,j) \cdot x_{C,j})) \ ,$$

where round returns the rounded value of its argument and i is the identifier of the evolving module. If $j = 0$, we define $\sigma_{C,B} \mapsto 0$ because there is no C in the neighborhood.

A bundle of such modules, each identified by a unique value (for instance an integer), can then be formed just by defining the neighboring relation. For instance, Fig. 4 shows such a bundle. In this example, the effective arguments of $\sigma_{C,B}$ in module 0 will be the singleton $\{(1, x_{C,1})\}$, while for module 1, it will be the 2-elements set $\{(0, x_{C,0}), (0.5, x_{C,2})\}$.

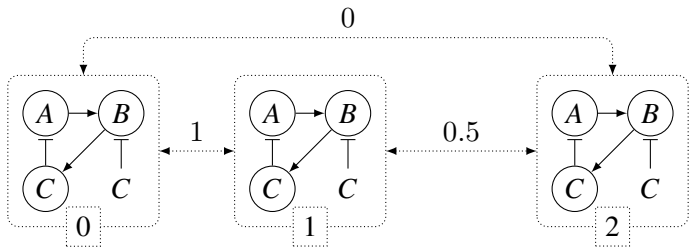


Figure 4: A bundle of regulatory modules encompassing three copies of the module from Fig. 3. The neighboring relation is depicted by the dotted double-arrowed arcs.

A state of a regulatory bundle is also defined as a vector of component levels, but that is indexed with the components as well as the module identifiers. For instance, states of the bundle in Fig. 4 may be indexed by $[x_{A,0}, x_{B,0}, x_{C,0}, x_{A,1}, x_{B,1}, x_{C,1}, x_{A,2}, x_{B,2}, x_{C,2}]$. The state space of bundles is computed similarly to the state space of regulatory networks, except that it involves the evaluation of integration functions, taking into account the neighboring relation δ.

2.3 Petri net semantics

In order to analyze regulatory bundles, a Petri net semantics has been defined [5]. For this purpose, a colored variant of Petri nets [13] has been used in which:

- *places* model containers of *tokens*, the latter being arbitrary values, *e.g.*, pairs of integers as in the following;
- *transitions* model activities that consume and produce tokens in places;
- *arcs* connect places to transitions (and vice-versa), indicating how tokens are consumed or produced by the transitions.

Fig. 5 shows an example of such a Petri net, that is actually the semantics of the regulatory bundle from Fig. 4. As usual, places are depicted by round nodes, transitions by square nodes and arcs by directed edges. In this net, places A, B and C are used to store the current level of each component in each module. For this purpose, they may be *marked* with tokens that are pairs $(i, x_{G,i})$ storing the level of a component G for a module identified by i. The marking thus corresponds to the state of the regulatory bundle. Then, transitions are used to implement the evolution of components. Consider for instance transition t_A, it consumes a pair $(i, x_{A,i})$ from place A, as well as a pair $(i, x_{C,i})$ from C; the latter is reproduced in the same place, as denoted by the double-arrowed arc, while the former is replaced by a pair $(i, x_{A,i} \triangleright K_A(x_{C,i}))$, that is, the evolution of A in module i. This evolution may happen for any module identifier, the consistency of consumed/produced values is ensured by the selection of a unique value for i during the execution, or *firing*, of t_A. Transition t_C behaves similarly. Transition t_B is slightly different because the evolution of B depends on A in the evolving module but also on the values of C in its neighborhood. So, t_B consumes $(i, x_{A,i})$ and $(i, x_{B,i})$ as expected, but also a set of pairs $(j, x_{C,j})$ that is computed dynamically on the arc from C to t_B, with respect to the function δ that encodes the neighboring relation. This set of pairs is bound to a variable $\vec{x_C}$ that is then used to compute $\sigma_{C,B}$ during the evaluation of K_B, allowing to produce in place B the new level of component B for module i.

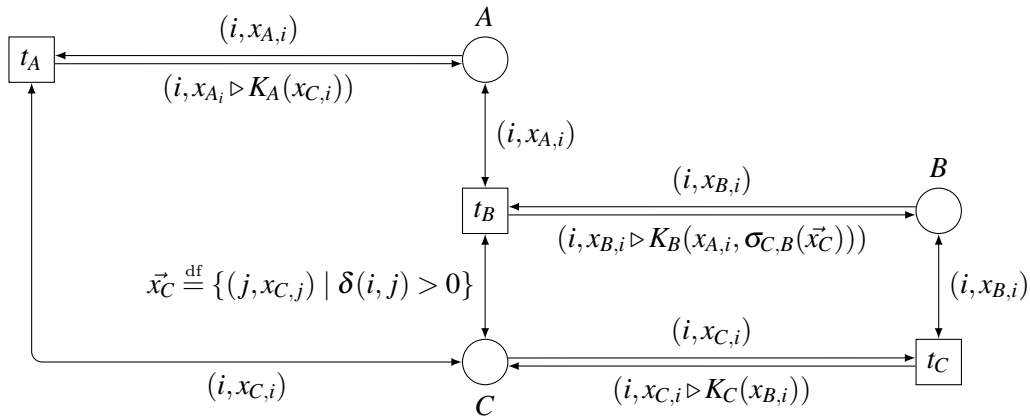


Figure 5: The Petri net defining the semantics of the regulatory bundle from figure 4.

The initial state of the regulatory bundle is rendered as the initial marking of the Petri net that defines its semantics: for each component G in a module i with the initial level $x_{G,i}$, one token $(i, x_{G,i})$ is added to the marking of place G.

2.4 Topological collections

We have just seen that the neighboring relation within a regulatory bundle can be encoded as a function δ that returns for each pair (i, j) of module identifiers (with $i \neq j$), the neighboring value of i with respect to j. This is a very general encoding but it is not related to any kind of geometrical, spatial or topological information. Our goal in this paper is to provide such an information in a way that allows for: spatial evolutions, enumerations of the possible evolutions, graphical rendering of spatial information. To do so, we resort to *topological collections*.

Topological collections have been introduced in [9] to describe arbitrary complex spatial structures that appear in biological systems [10]. They have been used to represents states of dynamical systems with a time varying structure [7, 12]. We will use them in this paper as a unifying framework to represent arbitrary neighborhood relationships. In this context, a neighborhood relationship between modules is represented by a labelled graph. Each vertex in this graph is labelled by a module identifier and a given identifier is the label of only one vertex. Two modules i and j are neighbors and $\delta(i, j) = \alpha$ if there is an edge labelled by α between the vertices associated with i and j. However, as we will see later on, such graphs will not be represented explicitly. Instead, relation δ will be computed dynamically with respect to some lower-level neighboring information.

Topological collections of various kinds have been implemented in an experimental programming languages called MGS [16]. This language is used here to implement a database that records the neighborhood relationships and that can be queried and updated efficiently.

3 Representing spatial information

In this section, we apply topological collections to the representation of spatial information in two ways. A first solution based on predefined grids is proposed: it is simple and efficient, but

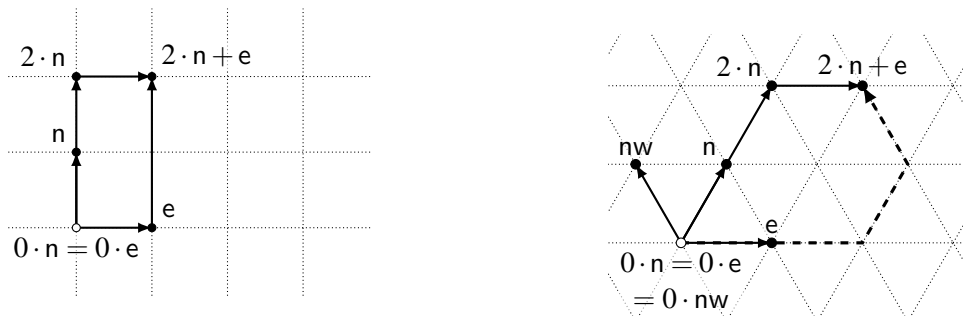


Figure 6: Left: a GBF defining a square grid, with two generators e and n. Right: a GBF defining a triangular grid with three generators e, n and nw, and a constraint $\mathsf{n} - \mathsf{nw} = \mathsf{e}$.

it is limited to represent systems in which modules may be inserted or moved only into existing "holes" in the grid. In order to overcome this limitation, we then propose another, more complex solution based on undirected graphs with limited degrees.

3.1 Placing modules on grids

Simple topological collections can be defined as *group based fields* (GBF), that can be considered as associative arrays whose indexes are elements in a group [8]. The latter is defined by a *finite presentation*: a set of generators together with some constraints on their combinations. Thus a GBF can be pictured as a labelled graph where the underlying graph is the Cayley graph of the finite presentation. The labels are the values associated with the vertices and the generators are associated with the edges.

For instance, in order to define a square grid, we may use two generators e (east) and n (north), supporting addition, difference and multiplication by an integer. This is illustrated in the left part of Fig. 6.

Similarly, a triangular grid can be defined by means of three generators n, e and nw (north-west) and a constraint $\mathsf{n} - \mathsf{nw} = \mathsf{e}$, as illustrated in the right part of Fig. 6. Notice that such a triangular grid is adequate to represent cells with a hexagonal shape, since the grid can be paved with hexagons centered at the positions in the grid. As shown by the dashed path, we have $2 \cdot \mathsf{n} + \mathsf{e} = 2 \cdot \mathsf{e} + \mathsf{n} + \mathsf{nw}$, which can be also checked in an algebraic way, by substituting nw with $\mathsf{n} - \mathsf{e}$ in this equality as allowed by the constraint.

The GBF structure is thus adequate to define the arrangement on a grid, in any number of dimensions. In such grids, a distance can be naturally defined as the minimum number of steps in order to reach one point from the other (this is the approach of *geometric group theory*). For instance, in the triangular grid of Fig. 6, points at e and n are at distance 1 because only one step in direction nw is required to reach the latter from the former; similarly, points n and $2 \cdot \mathsf{n} + \mathsf{e}$ are at distance 2. Let us denote by $\Delta(x,y)$ the distance between two points x and y. It is easy to check that $\Delta(x,y) = \Delta(y,x)$ for any x and y.

Such a distance can be used to implement our neighboring relation, for instance, for $x \neq y$, we could define:

$$\delta(x,y) \stackrel{\text{df}}{=} \frac{1}{\Delta(x,y)}$$

which matches the intuition that $\delta(x,y) = 1$ for two immediate neighbors while $\delta(x,y)$ converges

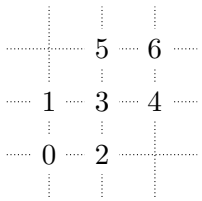


Figure 7: A square grid with modules (denoted by their identifiers) placed on it.

toward 0 when x and y become farther one each other.

The main drawback with grids is that inserting new elements is possible only if a hole is already present. Consider for instance Fig. 7 and assume that the modules represent cells forming a tissue. It would be difficult to model the division of cell 3, because there is no free position adjacent to 3. But in many biological system, we may expect that the division of cell 3 results in "pulling away" the neighboring cells. Similarly, if cell 3 is called to die, removing it from the grid will result in a disconnected tissue, which may be undesirable too.

However, the simplicity of grids is well adapted to model, *e.g.*, accretive growth that occurs at the borders of a tissue like leaves in plants [11]. Notice also that the graphical rendering of GBF is trivial to obtain because the grids are predefined.

3.2 A generalization of grids

In order to overcome the limitations of grids in the presence of cells migration, division or apoptosis, we now consider a topological collection based on undirected graphs, whose degree will be kept bounded. Modules will be located on the nodes of such graphs and edges will represent immediate neighboring. For instance, by bounding degree to 6, which we call a *6-bounded degree graph* (6-BDG), we shall obtain a result similar to triangular grids, as illustrated in Fig. 8.

Consider first the left part of Fig. 8: it depicts a 6-BDG with three nodes 0, 1 and 2, each being an immediate neighbor of the others, as depicted by the plain lines. The black dots and the dotted lines depict all the free locations next to node 0. Indeed, a new immediate neighbor next to 0 may be also a neighbor of 1, 2, both, or none of them. Notice that the geometrical position of nodes is not relevant, but only the edges between them are important. For instance,

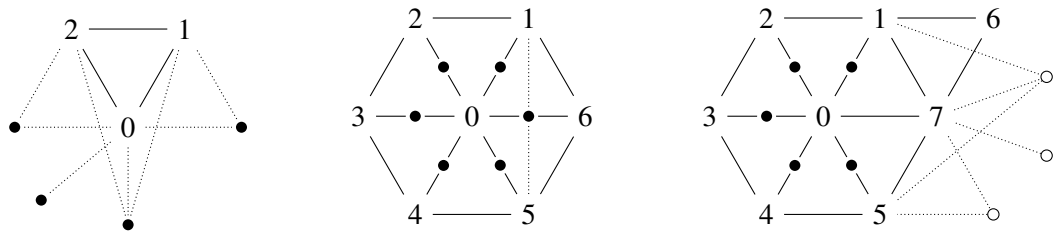


Figure 8: Left: a 6-BDG with 3 nodes and 4 possible insertion points from node 0. Middle: a 6-BDG with 7 nodes and 6 possible insertion points from node 0. Right: the same 6-BDG after the insertion of node 7 on edge 0—6. Dotted lines represent further possibilities to place node 6.

the black dot at the bottom may be equivalently depicted inside the triangle formed by nodes 0, 1 and 2. As long as node degrees is limited, the number of free locations next to a given node is limited as well.

When no new node can be inserted without violating the degree boundary, we proceed by splitting an existing edge. This is the case for instance in the middle of Fig. 8 where the black dots depict all the possible insertion points of a node next to 0. Dotted lines indicate the neighboring that would result from inserting a node on the edge from 0 to 6. This insertion results in "pulling away" node 6, which is depicted in the right part of the figure. White dots and dotted lines indicate alternate positions for the pulled node 6. In every case, there is also a limited number of possible graph transformations.

Deletion of nodes is handled in the most simple way, just by removing them in the graph. How the neighbors of a deleted node are then reconnected is modelled as a migration process. For instance, consider again the left part of Fig. 8, in particular the two left-most black dots, and assume a cell at the position of the lower dot (the one only connected to 0). This cell may be isolated from 2 because another cell, placed at the second dot, has just disappeared. The remaining cell may then migrate to fill this freshly freed position, or any of the positions depicted as black dots. In other words, a cell's death creates a hole that may then be filled by the migration of other cells around.

This system is thus much more flexible than the previous solution, but at the price of more complexity. However, this complexity is largely alleviated by MGS that provides almost ready-made solutions to implement such a system. An additional difficulty is to produce a graphical rendering of an n-BDG. Here also, MGS simplifies the problem by proposing graph layout algorithms that have been already proved relevant to represent graphically cells positions [19, 22].

In an n-BDG, a notion of distance can be defined as the length of the shortest path between two nodes. If such a path does not exist, *i.e.*, if the graph is not connected, the distance can be assumed infinite. Using this distance, we can then define the neighboring relation δ exactly as in GBFs.

4 Spatialized regulatory bundles

This section introduces an extension of regulatory bundles in order to take into account the spatial representations we have defined above. As a first step, we define a uniform interface for spatial information, that is independent of the actually chosen representation. The idea is that, at the level of semantics, this interface is queried from the Petri net part and, on the other hand, implemented on the top of a topological collection.

4.1 Spatial interfaces

Let us denote by $\mathbb{I}$ the set of module identifiers. We also define a set $\mathbb{L}$ of (abstract) *locations* that represent the positions in the chosen spatial representation. A *spatial interface* is a tuple (θ, δ, η) where:

- θ is a partial function $\mathbb{I} \to \mathbb{L}$ that maps some identifiers to locations, its domain being denoted by $\mathbb{I}_\theta$;
- $\delta : (\mathbb{I}_\theta)^2 \setminus \{(i,i) \mid i \in \mathbb{I}_\theta\} \to [0;1]$ is the *neighboring relation* as previously;

- $\eta : \mathbb{I}_\theta \to 2^{\mathbb{L}}$ returns for every module a set of *empty locations* in the immediate neighborhood of the module (this set may be empty).

In order to simplify notations, we may consider these functions as sets of pairs.

Intuitively, $\mathbb{I}_\theta$ defines the *allocated identifiers*, *i.e.*, those that correspond to actually existing modules, or living cells. On this allocated identifiers, δ encodes the neighboring relation. Finally, η is used when a cell is called to migrate or divide: in both cases, a new location next to the cell is needed, either as a new location for a migrating cell, or as a location for the newly created cell. Apoptosis, or freeing of a location after a migration, are simply modelled by restricting the domain of θ.

When a GBF-based representation is used, $\mathbb{L}$ is simply the set of all possible normalized linear combinations of generators, δ can be implemented as shown in section 3.1, and $\eta(i) \stackrel{\text{df}}{=} \{\ell \in \mathbb{L} \setminus \theta(\mathbb{I}_\theta) \mid \Delta(\theta(i), \ell) = 1\}$, *i.e.*, all the locations not already allocated and at distance 1 from the location of i. Exactly the same approach can be used for n-BDG, except that θ in the case must record positions is the graph, which is even simpler if the nodes are numbered consistently with the modules.

4.2 Regulatory modules with transformations

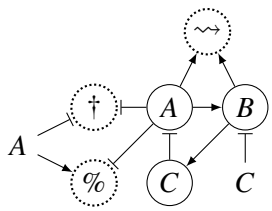
So far, a regulatory module defines the rules to change the levels of its components. This is not enough to model the migration, division or apoptosis processes. So we extend regulatory modules with three functions to specify these transformations:

- a function $K_\dagger$ for apoptosis, whose arguments are as for the regulatory functions, *i.e.*, levels of components within the module or its neighborhood. It returns a Boolean value indicating whether the cell modelled by the module should die or not;
- a function $K_\rightsquigarrow$ for migration, whose arguments are as for $K_\dagger$, plus an additional $L \subseteq \mathbb{L}$. It returns a subset of L that indicates the locations where the cell is allowed to migrate, taking into account information about the local and external levels of regulatory components;
- a function $K_\%$ for division, whose arguments are as for $K_\rightsquigarrow$ and that returns the set of locations where the new cell resulting from the division can be placed. In this paper, we assume that when a cell i divides, it yields the same cell i together with a new, identical cell associated with an unallocated identifier $j \in \mathbb{I} \setminus \mathbb{I}_\theta$. As shown later on, this policy can be easily replaced by another that is more suited to the biological system of interest.

How these transformation processes are regulated can be graphically rendered as special nodes, similar to regulatory components nodes. See for instance Fig. 9 that defines the transformations (right part) and depict them (left part):

- the cell may die whenever there is no more A expressed within the cell or its environment, which indeed means that A is a repressor for apoptosis;
- the cell may migrate to any available location if (and only if) there is some A or B expressed within the cell;
- the cell may divide, spawning to any available location, if (and only if) there is some A expressed in its environment but not inside the cell.

Notice that these new nodes have been depicted dotted because they can be regulated but cannot be used as regulators themselves.



$K_\dagger : x_A, \sigma_{A,\dagger} \to x_A = 0 \wedge \sigma_{A,\dagger} = 0$

$K_\rightsquigarrow : x_A, x_B, L \to L$ if $x_A + x_B > 0$ else $\emptyset$

$K_\% : x_A, \sigma_{A,\%}, L \to L$ if $\sigma_{A,\%} - x_A > 0$ else $\emptyset$

Figure 9: A regulatory module with transformations, based on the regulatory module of Fig. 3. Special nodes allow to depict how apoptosis (†), migration (⇝) and division (%) are regulated. Function $\sigma_{A,\dagger}$ and $\sigma_{A,\%}$ may be defined for instance as $\sigma_{C,B}$ previously.

Notice that we defined $K_\rightsquigarrow$ and $K_\%$ so that they can return a subset of their argument L. This has not been exploited in the example to keep it simple. But it is possible to implement complex functions that select among the locations in L only the "best" ones for migration or division. For instance, a function $K_\rightsquigarrow$ with an argument $\sigma_{G,\rightsquigarrow}$ may be designed to migrate in the directions of cells with a given level of component G among those proposed in its argument L.

4.3 Spatialized regulatory bundles

A *spatialized regulatory bundle* can be defined as a set of regulatory modules with transformations, together with a spatial interface. For example, we could build a bundle similar to that of Fig. 4 by using the module with transformations defined in Fig. 9 and equipping it with the following GBF-based spatial interface:

- $\theta \stackrel{\text{df}}{=} \{(0, 0\cdot\mathsf{n}), (1, \mathsf{e}), (2, 3\cdot\mathsf{e})\}$;
- $\delta(i,j) \stackrel{\text{df}}{=} 1/\Delta(\theta(i), \theta(j))$ if $\Delta(\theta(i), \theta(j)) \leq 2$ else 0;
- $\eta(i) \stackrel{\text{df}}{=} \{\ell \in \mathbb{L} \setminus \theta(\mathbb{I}_\theta) \mid \Delta(\theta(i), \ell) = 1\}$ as previously defined.

One can check that δ returns the same values as in the previous version of our example. (And this actually holds independently of whether square or triangular grids are used.)

The Petri net semantics is defined as before except that we need to add transitions to implement the spatial transformations. Moreover, the Petri net evolution is dependent on the spatial interface, that in turn may be modified when spatial transformations are executed. In the following, we consider that the interface elements are stored into some kind of global variables, that may be accessed from any part of the Petri net. (This could be implemented directly within Petri nets but would result in more complex notations.)

The semantics of the apoptosis transformation for our example is depicted in Fig. 10. It consumes all the current component levels for a module i, plus the levels of A in the neighborhood of i. The various levels of A are used to evaluate the Boolean expression $K_\dagger(x_{A,i}, \sigma_{A,\dagger}(\vec{x_A}))$ that appears as the *guard* of the transition, *i.e.*, a condition for its firing. Whenever the transition fires, it also updates θ by removing i from its domain; the transition also reproduce $\vec{x_A}$ in place A. The effect of this evolution is to remove from the places and from θ all the information related to module i, which corresponds to how its disappearing is rendered.

The Petri net semantics of the migration transformation for our example is depicted in Fig. 11. As usual, it consumes and reproduces the necessary levels in the adequate places. Then, thanks to the guard $\ell \in K_\rightsquigarrow(x_{A,i}, x_{B,i}, \eta(\theta(i)))$, a target location for migration is selected among the possible ones, and θ is updated accordingly upon firing.

Finally, the Petri net semantics of the division transformation for our example is depicted in Fig. 12. It reads (consume/produce) all the information from a module i and, thanks to its guard, binds to ℓ an acceptable location. The second part of the guard, $k = \mathsf{min}(\mathbb{I} \setminus \mathbb{I}_\theta)$, allows to bind to k, an unallocated identifier for the newly created cell. Its component levels are saved to the appropriate places as exact copies of the levels in module i so that the division creates two identical cells (at two adjacent locations), as required by the policy chosen above. If another rule is preferred, it can be implemented simply by changing the Petri net semantics of $K_\%$; for instance by computing new states and/or locations for the cells resulting from the division. Notice that we choose for k the smallest unallocated identifier (assuming that $\mathbb{I}$ is ordered) in order to reduce the combinatorial explosion during the evaluation of the guard. If min would not be used, every unallocated identifier could be bound to k, leading to as many successor states as the size of $\mathbb{I} \setminus \mathbb{I}_\theta$. On the contrary, it is important that the selection of ℓ is not constrained in the same way, which allows to explore all the possible evolutions of the spatial structure.

4.4 Implementation and applications

Spatialized regulatory bundles have been implemented as a prototype, using SNAKES [18, 17] for the Petri net part, and MGS [16] for the topological collections part. SNAKES is a general purpose Petri net library implemented in Python, that is particularly suitable for the quick development of prototypes. This implementation allows to define regulatory modules as special cases of Python classes in which methods implement the evolution and transformation rules. Bundles can then be populated with instances of these modules and their locations can be specified as MGS expressions, consistently with the chosen spatial interface. The Petri nets semantics can be automatically computed (and then exported or drawn) and its state space can be explored for analysis purpose.

For our implementation, we have actually reused a previous prototype developed in the context of [5] and extended it as follows:

- spatial interfaces have been implemented within MGS on the top of both GBFs and BDGs;
- a bridge to allow calls to MGS from Python has been realized by driving the MGS shell from Python through a pair of pipes;
- the original implementation of the neighboring relation (a simple map from pairs of identifiers to float values) has been replaced by queries to the spatial interface;
- the semantics of transformations for cells migration, division or apoptosis has been added.

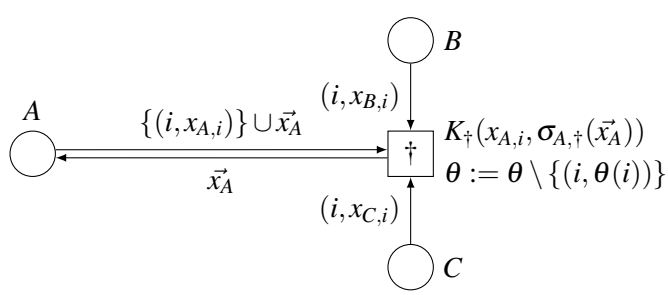


Figure 10: The Petri net semantics of transformation $K_\dagger$, where $\vec{x_A} \stackrel{\text{df}}{=} \{(j, x_{C,j}) \mid \delta(i,j) > 0\}$.

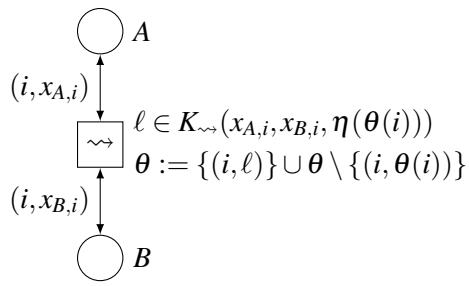


Figure 11: The Petri net semantics of transformation $K_{\rightsquigarrow}$.

This prototype implementation has been applied to the study of two simple biology-inspired examples, allowing to test various features of our framework [3]:

1. A system with transcription signal cascading resulting in a cell division allowed to observe the alternation of growth phases and divisions, which can be interpreted as a mutual influence between components evolutions and spatial transformations. Moreover, by controlling a chosen set of growth factors, the duplication of the modelled cells could be controlled. This example used a GBF representation for the spatial information.

2. A simplified model of epithelial cellular healing has been defined with cells arranged on a ring defined as a 2-BDG. A ring has been chosen in order to simulate an infinitely long chain of cells, avoiding the healing process to be triggered at the borders of the chain. When one cell of the ring is removed (or dies), the regulation results in the duplication of one of the cells at distance 2 from the removed cell. This is consistent with the observed biological behavior where division does not occur on the edges of a wound but one "layer" of cells farther.

5 Related and future works

Integrative modelling of biological processes (*e.g.* in system biology) relates different models that operate on different levels of abstraction and various spatial and time scales. For instance,

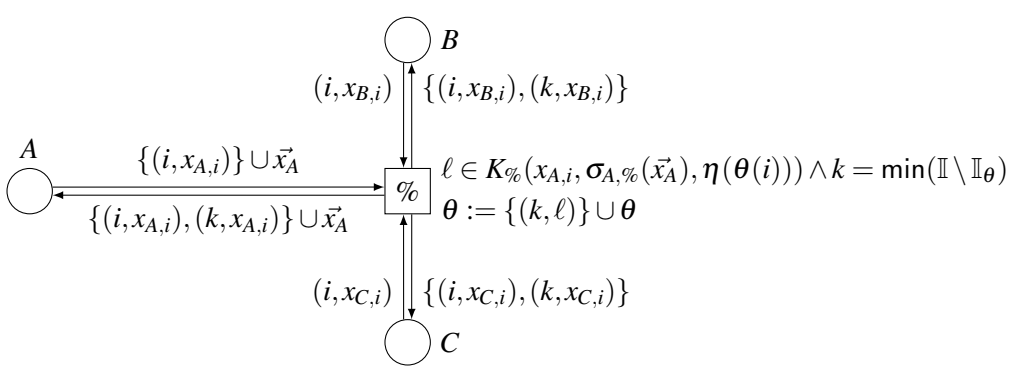


Figure 12: The Petri net semantics of transformation $K_{\%}$, where $\vec{x_A}$ is the same as previously.

in developmental biology, the spatial organization of cells is especially crucial and the description of the morphogenetic processes at a cellular level [19] implies the integration of molecular mechanisms such as cell-cell signaling, mechanical stresses and genetic regulation on a complex dynamic geometry. Various formalism have been proposed to face these needs relying on various explicit representations of space [24]: molecular dynamics, spatial Gillespie [23], partial differential equations (PDE), lattice based methods (*e.g.*, cellular automata), etc. See Fig 13 for a brief taxonomy of space representation in biological modelling.

In this context, we want to develop a modelling framework suitable to represent and study the regulations in multi-cellular systems, taking into account the spatial relationships between the cells as well as the spatial transformations resulting from cells divisions, migrations, or apoptosis. Discrete algebraic formalisms like P systems, process algebra or Petri net are very relevant because the automated tools that can be used to help both the modelling and the systematic analysis of the system behavior. Such formalisms take into account spatial relationships. For instance, classical membrane systems rely on membranes inclusion to abstract the spatial organization of cellular processes. The limitations of this structure has been recognized [9] and leads to the development of several variants: tissue P systems [28], population P systems [1], etc. Some process algebra (*e.g.*, used to study mobility or variants of π-calculus used for biological modelling) rely also on a notion of localization but often the spatial relationships are not explicitly exposed (the algebra of locations is coded into identifiers) or too limited (nesting structures).

The framework presented in this paper is based on the well-known formalism of logical regulatory networks and extends it with spatial information and a definition of the spatial transformations. A Petri net semantics of the resulting formalism has been proposed in order to allow for the simulation and the analysis of the modelled systems (*e.g.*, through model-checking,

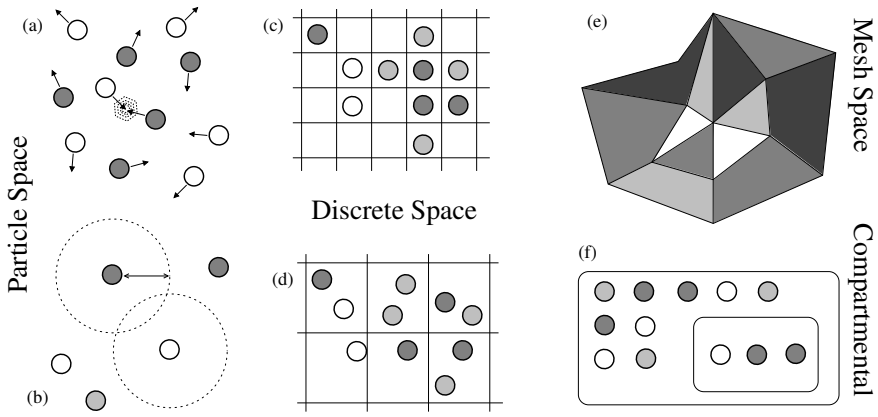


Figure 13: Various representation of space in systems biology according [24]. (a) In particle space, molecules are represented as individual particles with positions in a continuum space. (b) Such methods accommodates a discrete dynamics when particles jump in time and space by calculating the maximum distance that the particle can travel and interact in the time slot. Lattice discretizes regularly the space: in microscopic lattices (c) at most one particle is allowed to occupy a lattice site while mesoscopic methods (d) allows several. (e) Mesh spaces are usually used to solve PDE (*e.g.*, in reaction-diffusion systems). (f) Compartmental spaces focus on the molecular transfers between compartments.

invariant extraction, etc.). Our solution is particularly flexible and general thanks to a *decoupling* of various concerns: in particular the regulation aspects are grasped by regulatory networks while the spatial aspects are handled through topological collections which unify several spatial representations. These different aspects are addressed using their own adequate tools. They are then related by means of a general interface allowing a smooth bi-directional interaction while preserving the decoupling.

The current presentation is limited to systems in which only one kind of cell can exist. Our prototype implementation does not have such a limitation since this is not a technical limitation but a deliberate presentation decision to in order to focus on the intuition about our approach. Future works will provide a generalized formal definition of the framework presented in the paper. We also intend to run several case studies in order to assess the relevance of our proposal: in particular we shall consider models of *Drosophilia* embryo as in [5] and extend them with cells divisions. Another aspect will be addressed in the future: allowing that cells division immediately results in the possibility of infinite state space systems. This can be tackled by limiting the number of living cells at a given time (*i.e.*, use a finite set of module identifiers); another possibility is to resort to abstraction, for instance by applying to module identifiers a state space reduction technique originally designed for systems with dynamic process creation [14, 15]. Finally, we intend to study an alternative approach for spatial information in order to address systems such as blood-cells populations. The idea is to implement the spatial relation as a purely stochastic relation reflecting the idea that such cells are in constant movement but may meet occasionally.

Acknowledgements. The authors thank F. Delaplace at the University of Évry for fruitful discussions and the anonymous reviewers for their valuable comments. This work is partially supported by the ANR research projects AutoChem and Calamar.

References

[1] Francesco Bernardini & Marian Gheorghe (2004): *Population P systems. Journal of Universal Computer Science* 10(5), pp. 509–539.

[2] C. Chaouiya, E. Remy & D. Thieffry (2006): *Qualitative Petri Net Modelling of Genetic Networks. Trans Comp Syst Biol* VI(4220), pp. 95–112.

[3] S. Castagnet (2009): *Modélisation de tissus biologiques : localisation et abstraction.* Master's thesis, IMBI, University of Évry.

[4] C. Chaouiya, H. Klaudel & F. Pommereau (2009): *A Petri net based framework for a qualitative modelling of regulatory networks encompassing inter-cellular signalling.* In: A. Navarro & A. Oliveira, editors: *JB'09*, pp. 1–5.

[5] C. Chaouiya, H. Klaudel & F. Pommereau (to appear): *A modular, qualitative modelling of regulatory networks using Petri nets*, chapter 12 of *Modeling in Systems Biology – the Petri Net Approach.* Springer.

[6] C. Chaouiya, A. Naldi, E. Remy & D. Thieffry (to appear): *Petri net representation of multi-valued logical regulatory networks. Natural Computing* .

[7] J.-L. Giavitto (2003): *Topological Collections, Transformations and Their Application to the Modeling and the Simulation of Dynamical Systems.* In: *14th International Conference on Rewriting Techniques and Applications (RTA 2003)*, LNCS 2706, Springer, pp. 208–233.

[8] J.-L. Giavitto & O. Michel (2001): *Declarative definition of group indexed data structures and approximation of their domains.* In: *PPDP '01: Proceedings of the 3rd ACM SIGPLAN international*

conference on Principles and practice of declarative programming, ACM Press, New York, NY, USA, pp. 150–161.

[9] J.-L. Giavitto & O. Michel (2002): *The Topological Structures of Membrane Computing.* Fundamenta Informaticae 49, pp. 107–129.

[10] J.-L. Giavitto & O. Michel (2003): *Modeling the topological organization of cellular processes.* BioSystems 70, pp. 149–163.

[11] J.-L. Giavitto, O. Michel & J. Cohen (2002): *Accretive Rules in Cayley P Systems.* In: *Selected paper from the International Workshop on Membrane Computing (WMC-CdeA 2002)*, LNCS 2597, Springer, pp. 319–338.

[12] J.-L. Giavitto & A. Spicher (2008): *Topological rewriting and the geometrization of programming.* Physica D: Nonlinear Phenomena 237(9), pp. 1302 – 1314. Novel Computing Paradigms: Quo Vadis?

[13] K. Jensen & L. M. Kristensen (2009): *Coloured Petri Nets: Modelling and Validation of Concurrent Systems.* Springer.

[14] H. Klaudel, M. Koutny, E. Pelz & F. Pommereau (2009): *An Approach to State Space Reduction for Systems with Dynamic Process Creation.* In: *ISCIS'09*, IEEE digital library 15777, IEEE, pp. 1–6.

[15] H. Klaudel, M. Koutny, E. Pelz & F. Pommereau (to appear): *State Space Reduction for Dynamic Process Creation. SACS* .

[16] O. Michel, J.-L. Giavitto, J. Cohen & A. Spicher. *The MGS homepage.* Available at `http://mgs.spatial-computing.org`.

[17] F. Pommereau. *SNAKES is the net algebra kit for editors and simulators.* Available at `http://pommereau.blogspot.com/2010/01/snakes.html`.

[18] F. Pommereau (2008): *Quickly prototyping Petri nets tools with SNAKES.* Petri net newsletter October, pp. 1–18.

[19] P. Barbier Barbier de Reuille, I. Bohn-Courseau, K. Ljung, H. Morin, N. Carraro, C. Godin & J. Traas (2006): *Computer simulations reveal properties of the cell-cell signaling network at the shoot apex in Arabidopsis.* Proceedings of the National Academy of Sciences of the United States of America 103(5), pp. 1627–1632. Available at `http://www.pnas.org/content/103/5/1627.abstract`.

[20] J. Saez-Rodriguez, L. Simeoni, J. A. Lindquist, A. Hemenway, U. Bommhardt, B. Arndt, U-U. Haus, R. Weismantel, E. Gilles, S. Klamt & B. Schraven (2007): *A logical model provides insights into T cell receptor signaling.* PLoS Comput Biol 3(8), p. e163.

[21] L. Sánchez, C. Chaouiya & D. Thieffry (2008): *Segmenting the fly embryo: a logical analysis of the segment polarity cross-regulatory module.* Int. J. Dev. Biol. 52(8), pp. 1059–75.

[22] A. Spicher & O. Michel (2005): *Using Rewriting Techniques in the Simulation of Dynamical Systems: Application to the Modeling of Sperm Crawling.* In: Fifth International Conference on Computational Science *(ICCS'05)*, 3514-I, pp. 820–827.

[23] A. B. Stundzia & C. J. Lumsden (1996): *Stochastic Simulation of Coupled Reaction-Diffusion Processes.* Journal of Computational Physics 127(1), pp. 196 – 207.

[24] K. Takahashi, S. N. V. Arjunan & M. Tomita (2005): *Space in systems biology of signaling pathways - towards intracellular molecular crowding in silico.* FEBS Letters 579(8), pp. 1783 – 1788.

[25] R. Thomas (1973): *Boolean formalization of genetic control circuits.* J. Theor. Biol. 42, pp. 563–85.

[26] R. Thomas (1991): *Regulatory networks seen as asynchronous automata: A logical description.* J. Theor. Biol. 153, pp. 1–23.

[27] R. Thomas, D. Thieffry & M. Kaufman (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. 247–76.

[28] C.M. Vide, J. Pazos, G. Paun & A.R. Patón (2003): *Tissue P systems.* Theoretical Computer Science 296, pp. 295–326.