

UNIVERSIDADE DE LISBOA  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE INFORMÁTICA



## **Update schemes and other extensions to support logical modelling of multicellular systems**

Sofia de Sousa Torres

**Mestrado em Bioinformática e Biologia Computacional**

Dissertação orientada por:  
Prof. Doutor Pedro Monteiro  
Prof. Doutor Francisco Pinto



# Acknowledgements

This thesis would not have been possible without the help of a few people, thus I'd like to express my gratitude to:

Claudine Chaouiya and Pedro Monteiro, my two main supervisors, for welcoming me into this project and allowing me to learn from them. The countless hours spent advising and helping me has been extremely valuable to my growth as a student and as a person.

Francisco Pinto, my internal supervisor, for his sympathy and availability.

Finally, I'd like to show my thanks to my parents for their unconditional love and support, as well as my siblings and both grandmothers, without whom I would not have been able to succeed.



# Resumo alargado

**Keywords:** Modelos lógicos, Autómato celular, Esquemas de atualização, Sistemas multicelulares, Formação de padrões

O estado de uma célula é controlado por vários componentes biológicos, como genes, proteínas, metabólitos, que interagem entre si, criando grandes redes regulatórias. Os componentes dessas redes não só regulam os outros componentes, como também a si próprios, fazendo-o através de fatores de transcrição, fosforilação, entre outros. Vários formalismos foram usados para modelar este tipo de redes, entre eles está o formalismo lógico, que representa uma rede regulatória como um grafo, onde os vértices correspondem a componentes biológicos e as arestas às interações entre estes. Este formalismo é especialmente bem sucedido no estudo deste tipo de redes devido à abstração do componente num vértice que permite que este represente qualquer tipo de componente biológico, refletindo assim a complexidade biológica inerente a estas redes. Também o tipo de interação é abstraído, considerando apenas o impacto da interação, se é positiva ou negativa. O modelo lógico é caracterizado por tradicionalmente considerar apenas dois valores para os seus componentes, a sua ausência (0) ou a sua presença (1). O valor de cada componente é regulado por uma função lógica dependente dos componentes que o regulam, esta função lógica representa as interações antagónicas, acumulativas, entre outras, no componente regulado.

O estudo dos sistemas biológicos estende-se também à consideração de sistemas multicelulares. A dinâmica destes sistemas leva à formação de padrões devido à interação (sinalização) entre células. Estes padrões resumem-se a organizações espaciais de células que atingem estados estáveis (estados de onde qualquer célula não consegue sair).

O EpiLog (*Epithelium Logical modelling*) é uma ferramenta informática desenvolvida em Java, com interface gráfica, utilizada para estudar a formação de padrões sobre um epitélio. Este epitélio é modelado por um autómato celular, composto por uma matriz de células hexagonais, onde o estado de cada célula é controlado por um modelo lógico. A implementação e simulação dos modelos lógicos é feita pela biblioteca bioLQM, utilizado pelo EpiLog.

O autómato celular, *framework* usada pelo EpiLog, é um formalismo discreto usado para estudar propriedades de auto-organização emergentes das interações entre os autómatos. Consiste numa matriz de autómatos onde o valor zero ou um associado a cada célula (ou autómato) é regulado pelas interações com os seus vizinhos próximos. O EpiLog estende esta definição ao associar modelos lógicos a cada célula, o que permite uma representação mais complexa do estado da célula. Também a definição de vizinhança é alterada para permitir a modelação de sinalização entre células não adjacentes, representando a sinalização parácrina.

A dinâmica de um modelo lógico é influenciada pela escolha do esquema de atualização. O esquema de

## 0. RESUMO ALARGADO

atualização determina a ordem pela qual os valores dos componentes dos modelos são atualizados, isto é, a ordem pela qual a função de regulação é aplicada. O EpiLog faz a atualização das suas células do epitélio considerando “classes de prioridade”. Este esquema define que os componentes dos modelos se dividam em classes com *ranks* associados, de tal modo que os componentes pertencentes a classes com *ranks* mais baixos não são atualizados enquanto existirem componentes de classes com *ranks* mais elevados que sejam atualizáveis. As classes em si são atualizadas de forma síncrona, isto é, os componentes pertencentes à mesma classe são atualizados simultaneamente, gerando um único sucessor. Do ponto de vista biológico, o esquema de atualização síncrono não faz sentido, visto implicar que todos os vários processos biológicos aconteçam com a mesma taxa de velocidade. Este pressuposto leva a artefatos de modelação que não correspondem ao observado *in vivo*. Por oposição ao esquema de atualização síncrono, o assíncrono atualiza cada componente individualmente, criando tantos sucessores quanto o número de componentes que possam ser atualizados.

O foco principal desta tese é a implementação de novos esquemas de atualização que introduzem alguma assincronicidade dentro das classes de prioridades, com o objetivo de mitigar e explorar as fragilidades do esquema de atualização síncrona. A implementação é feita no bioLQM, onde as classes são compostas por grupos, sendo que estes tem um esquema de atualização associado, os sucessores da classe são a união dos sucessores dos grupos da classe. O EpiLog limita uma classe a apenas um grupo, tornando os dois essencialmente sinónimos. Os novos esquemas de atualização dividem-se em sucessores múltiplos e sucessores únicos, EpiLog usa apenas os segundos, sendo estes o uniformemente aleatório e não uniformemente aleatório.

O esquema uniforme aleatório corresponde à escolha aleatória de um sucessor entre os sucessores resultantes de um esquema assíncrono, enquanto o esquema não uniforme aleatório implica que seja atribuído a cada componente uma probabilidade de ser atualizado, e o sucessor final é escolhido dos sucessores assíncronos com base nessa probabilidade. Ambos os esquemas permitem que um maior número de trajetórias seja explorado, as resultantes do esquema assíncrono, o esquema de atualização não uniforme aleatório permite também integrar conhecimento cinético sobre as interações do sistema, levando à exploração de trajetórias em teoria com maior significado biológico.

A utilidade dos novos esquemas foi testada através de um caso de estudo, um modelo lógico já publicado do módulo de genes *segment polarity*, que consolida os segmentos do embrião da *Drosophila*. É mostrado que os novos esquemas de atualização permitem obter a maioria dos padrões estáveis obtidos na publicação original ao contrário do esquema síncrono inicial. São testadas várias mutações além do caso *wild-type*, e são considerados dois epitélios: um único segmento de seis células, e um epitélio de 12 por 12 células. O primeiro epitélio corresponde à situação modelada na publicação original, e a replicação dos resultados foi conseguida com ambos os esquemas de atualização “duas classes com uniforme aleatório” e apenas “não uniforme aleatório” (ou seja uma classe). Este resultado demonstra a necessidade da introdução da assincronicidade nos esquemas de atualização. No epitélio de 12 por 12 células devido ao grande número de células ( $12 \times 12$ ) e conseqüentemente ao número elevado de estados possíveis, foi necessário definir um esquema de atualização mais restrito, o “duas classes com não uniforme aleatório” para obter os mesmos resultados. Este último resultado mostra a utilidade da combinação dos novos esquemas de atualização com as classes de prioridade, que oferecem ao modelador maior flexibilidade e controlo sobre que trajetórias explorar. Também é mostrado que com a implementação dos novos esquemas, o EpiLog está mais apto para modelar grandes epitélios, sendo assim uma melhor ferramenta no estudo de formação de padrões em sistemas multicelulares.

Uma vez que os novos esquemas de atualização foram implementados no bioLQM, favorecem não só o EpiLog, como também todas as ferramentas dependentes do bioLQM. Estas ferramentas ganham agora classes de prioridades que permitem a definição de vários grupos numa classe, e esquemas de atualização de grupos que aceitam esquemas de atualização com sucessores múltiplos além de únicos.

Além da implementação dos novos esquemas de atualização, foram adicionadas duas outras funções ao EpiLog, com o objetivo de tornar mais completa e fácil a experiência de modelação do utilizador.

A primeira, chamada “phenotype tracking”, foi motivada pela necessidade de uma visão alternativa e complementar ao output do EpiLog, que se foca ao nível do epitélio, dando uma representação gráfica do estado do mesmo (onde o estado das interno células pode ser consultado ao ser clicado). Esta nova ferramenta gera uma série temporal dos estados internos das células ao longo de uma simulação, o que melhora o entendimento do utilizador sobre os resultados de uma simulação. É também possível definir sobre que estados a informação será gerada, isto é, a definição de fenótipos que consistem em conjuntos de estados da célula (do modelo lógico) que caracterizam uma célula. Os fenótipos são definidos pelo utilizador dependente do interesse biológico que lhe atribui.

Foi também implementada uma maneira alternativa de editar as definições de um epitélio, que permite ao utilizador fazê-lo de forma textual, evitando o uso da interface gráfica. Esta adição é indicada para utilizadores mais experientes, permitindo assim uma edição mais rápida e eficiente.





# Abstract

**Keywords:** Logical modelling, Cellular automata, Updating Schemes, Multi-cellular systems, Pattern formation

The state of a cell is controlled by a regulatory network of biological components. The logical formalism is a powerful discrete framework to model these networks, as it abstracts the type of biological components capturing the nature, positive or negative, of their interactions.

Cell-cell signalling results in pattern formation through the acquisition of distinct cellular phenotypes. This work focuses on the logical modelling of multi-cellular systems, accounting for cell-cell signalling. EpiLog is a Java tool for studying pattern formation of an epithelium, modelled by a cellular automaton, composed of a two-dimensional grid of hexagonal cells, where logical models govern their internal states. The dynamics of the cellular logical model are simulated by the bioLQM toolkit, used by EpiLog.

The model dynamics are influenced by updating schemes, which set the order of the model components updates. EpiLog uses priority synchronous classes, where all the class components are updated simultaneously. Synchronous updating unrealistically assumes that all processes occur at the same time, producing modelling artefacts.

This thesis implements new updating schemes to mitigate the drawbacks of the synchronous update, by introducing some asynchronicity. The usefulness of the new updating schemes is confirmed with a case study, a published logical model of the segment polarity module, which consolidates the fly embryo segments. We show that the new updating schemes capture most of the stable patterns obtained in the original publication, and that EpiLog with these new updaters is thus more suitable to study pattern formation in multi-cellular systems.

Additionally, two features were implemented. The user can now better assess simulation results thanks to the generation of data concerning the cell states along the simulation. Furthermore, the possibility to edit model definitions rather than doing so through the GUI facilitates the work of the advanced users.



# Table of contents

<b>Acknowledgements</b>	<b>iii</b>
<b>Resumo alargado</b>	<b>v</b>
<b>Abstract</b>	<b>ix</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Goals and thesis overview . . . . .	2
<b>2 State of the Art</b>	<b>5</b>
2.1 Logical modelling of cellular networks . . . . .	5
2.1.1 Model definition . . . . .	5
2.1.2 Model dynamics . . . . .	6
2.1.3 Perturbations . . . . .	8
2.2 Multi-cellular modelling . . . . .	8
2.2.1 Existing approaches . . . . .	8
2.2.2 Cellular automata . . . . .	9
2.2.2.1 CA updating scheme . . . . .	9
2.2.2.2 CA - integration . . . . .	10
<b>3 Modelling Tools</b>	<b>11</b>
3.1 Logical modelling tools . . . . .	11
3.1.1 BioLQM . . . . .	11
3.1.2 Other logical modelling tools . . . . .	12
3.2 Multi-cellular modelling tools . . . . .	13
3.2.1 EpiLog . . . . .	13
3.2.1.1 Model definition . . . . .	13
3.2.2 Other multi-cellular modelling tools . . . . .	14

# TABLE OF CONTENTS

<b>4</b>	<b>BioLQM extensions</b>	<b>15</b>
4.1	Previous Priority Updater . . . . .	15
4.2	Priority updater integration with other updating schemes . . . . .	15
4.3	Approach and Priority updater architecture . . . . .	17
<b>5</b>	<b>EpiLog extensions</b>	<b>21</b>
5.1	BioLQM extensions integration with EpiLog . . . . .	21
5.2	Tracking simulation phenotype data . . . . .	21
5.2.1	Proposed changes . . . . .	22
5.2.2	Interface approach and phenotype architecture . . . . .	25
5.3	Textual mode to define the parameters of an Epithelium . . . . .	25
<b>6</b>	<b>Evaluation of new updating schemes with the Segment Polarity model</b>	<b>27</b>
6.1	Background . . . . .	27
6.2	Segment polarity multi-cellular model in EpiLog . . . . .	30
6.2.1	Introduction . . . . .	30
6.2.2	Single para-segment: stripe of six cells . . . . .	31
6.2.2.1	Wild-type: no mutations . . . . .	31
6.2.2.2	Mutants . . . . .	32
6.2.3	Multiple para-segments: grid of 12 by 12 cells . . . . .	32
6.2.3.1	Wild-type . . . . .	32
6.2.3.2	Mutants . . . . .	33
<b>7</b>	<b>Conclusions and further work</b>	<b>37</b>
	<b>Appendices</b>	<b>45</b>
<b>A</b>	<b>Graphical user interface manual</b>	<b>47</b>
A.1	Priority Panel - bioLQM . . . . .	47
A.2	EpiLog . . . . .	48
A.2.1	Cellular model update: Priority Panel . . . . .	48
A.2.2	Phenotype definition . . . . .	49
A.2.3	Edit by text . . . . .	50
<b>B</b>	<b>JUnit tests</b>	<b>53</b>
B.1	Models . . . . .	53
B.2	JUnit tests . . . . .	53
B.2.1	Random uniform . . . . .	54
B.2.2	Random non uniform - Random with rates . . . . .	54
B.2.3	Priority Updater . . . . .	55
B.2.4	Simulation: Priority Updater . . . . .	56

# List of Figures

2.1	Logical toy model . . . . .	7
2.2	Lateral Inhibition illustration . . . . .	10
4.1	Flow diagram of the Priority Updater . . . . .	16
4.2	Scheme representation of the Priority Updater implementation . . . . .	16
4.3	Scheme of BioLQM updaters classes . . . . .	18
4.4	Priority class panel, belonging to bioLQM . . . . .	19
5.1	EpiLog: Cellular model update tab . . . . .	22
5.2	EpiLog: Phenotypes GUI operations . . . . .	23
5.3	EpiLog: Scheme phenotype architecture . . . . .	24
5.4	EpiLog: Edit by text phenotype definitions dialog . . . . .	26
6.1	Segment polarity logical model by Sanchez 2008 . . . . .	28
6.2	Initial conditions of segment polarity six cell model . . . . .	30
6.3	Stable states for six cell stripe reachable from realistic initial conditions . . . . .	31
6.4	Stable states for grid of cells reachable from realistic initial conditions . . . . .	34
6.5	One stable state for grid of cells with some irregularities . . . . .	35
A.1	Priority panel - Multiple groups manual . . . . .	47
A.2	EpiLog manual: Cellular model update - priority panel . . . . .	48
A.3	EpiLog manual: Phenotype definition . . . . .	49
A.4	EpiLog manual: Edit by text mode . . . . .	50
A.5	EpiLog manual: other edit by text dialogs . . . . .	51
B.1	JUnit: Logical Model #2 . . . . .	53



# List of Tables

- 4.1 bioLQM priority updater, before and after extensions . . . . . 17
- 6.1 Segment polarity single cell model stable states . . . . . 29
- 6.2 Initial conditions for segment polarity model wild-type . . . . . 30
- 6.3 Segment polarity six cells stable states with no perturbations . . . . . 31
- 6.4 Segment polarity six cells stable states with perturbations . . . . . 35
- 6.5 Segment polarity random non uniform updating scheme for grid of 12 by 12 cells . . . . . 36





# Chapter 1

## Introduction

### 1.1 Motivation

In the field of biology, more specifically in molecular biology, research has been focused mainly in retrieving information about interaction between systems components and cataloguing it. With the advances of high-throughput screenings the amount of data has skyrocket in the last few years (D'Argenio, 2018).

However, large quantities of data do not translate into knowledge of the underlying mechanisms of a system. To do so, it is necessary to formulate theoretical hypotheses, such as models, that are able not only to explain the data that helped formulate it, but also to make predictions that can be experimentally tested. As a result, theory and experimentation work best together, with model predictions directing the work of experimenters. Predictions that are empirically demonstrated to be incorrect, on the other hand, indicate a misunderstanding of the system, prompting a revision of the original hypothesis (Palsson, 2000).

With the advances of computational power, modelling became a popular tool to study biological systems as it allows the consideration of many variables that are typical of such systems. Particularly, molecular networks have been modelled by multiple computational strategies (Smolen, 2000; Jong, 2002), which differ on the level of detail and information type they support.

A cell's behaviour is influenced by the interaction between many biological molecules, be genes, proteins, metabolite, etc. These constitute molecular networks, in which the components regulate each other and sometimes themselves (through transcription factors, phosphorylation, and so on).

Systems of ordinary differential equations (ODEs) have extensively been used to study molecular networks. They permit a fine grain analysis as they consider continuous time and continuous variables (e.g. concentration levels). (Le Novère, 2015).

A great limitation of this type of approaches is that it requires multiple variables and parameters. These parameters prove to be a big obstacle as they are hard to determine and rarely found in literature. However, certain properties of biological systems, such as steady-state behaviour, have been proven to be robust to different levels of gene activity and kinetic parameters (Alon et al., 1999; Dassow et al., 2000).

Logical models take advantage of the absence of parameterization and intrinsic robustness of regulatory networks. The regulation of gene transcription is explained by a sigmoidal Hill-function. Due to the switch-like nature of gene expression regulation, this function can be approximated to a step-function, which equates to a boolean approximation. As a result, a logical model component has only two possible states,

## 1. INTRODUCTION

present/active "1", and absent/inactive "0". Glass et al., 1973 showed that this approximation conserves the stable states found on the continuous model counterpart.

This framework is commonly depicted by a directed graph, with nodes representing system components and edges representing regulatory interactions. A node can indicate a gene product or any other biological component that affects the genetic regulatory network. An interaction between two nodes is either positive or negative (Abou-Jaoudé et al., 2016).

Despite the high level of abstraction, this approach is able to capture features like attractors (steady states or cyclic attractors) and has been showed to successfully model biological systems (Abou-Jaoudé et al., 2016).

The formalisms described above are used to study the intra-cellular mechanisms that control cell behaviour. However, there are several biological events of interest that depend on cell-cell communication, which eventually leads to pattern formation, *i.e.*, a structured arrangement of cells, in either state or form, on a spatial component.

These patterns are either the consequence of inter-cellular signalling, which alters the cells internal states, or of morphological events that change the spatial distribution of the cells, additionally, a combination of the two events may also occur (Salazar-Ciudad et al., 2003).

Cellular automata constitute a discrete modelling framework extensively used to study self-organisation of individual units into macro-scale patterns. Originally, these units, were logical units whose state, either zero or one, was decided by their local interactions with neighbouring units. Therefore, this framework is an ideal candidate to model biological systems at the macro scale, where the outcomes depend on the interactions between cells (Schiff, 2007).

EpiLog, which is at focus on this thesis, is a software tool for multi-cellular logical modelling. It captures biological patterns on an epithelium, which is modelled by a cellular automaton, where each unit is an hexagonal cell associated with its own logical model. EpiLog uses the Logical Qualitative Modelling toolkit (bioLQM) to handle the logical models associated with the individual cells (Varela et al., 2019).

The logical model components have their values updated according to the updating scheme they employ. Succinctly, the synchronous updating scheme, where all model components are updated at once, is known to create artefacts that lead to outcomes not seen *in vivo*, nor in continuous models (Glass et al., 1973). Alternatively, the asynchronous updating updates all model components individually, creating as many successors as there are updatable components. EpiLog intra-cellular updating scheme, called "priority updater", divides its model components in classes that are each updated synchronously, as a consequence, it is still the application's major shortcoming.

### 1.2 Goals and thesis overview

This thesis main focus is to further the development of the tool EpiLog, enabling the modelling community to more accurately capture the dynamics of a grid of cells that embody an epithelium. More precisely, three extensions were implemented:

- **Extension of bioLQM updating schemes;** EpiLog relies on the bioLQM toolkit to handle individual cellular logical models. Currently, the updating scheme used by EpiLog does not allow for non synchronous updating within a class of the priority updater. The modification of bioLQM thus consisted

## 1.2 Goals and thesis overview

in supporting alternative updating schemes.

- **Addition of a tool to track cell states in EpiLog;** EpiLog users can visually follow the simulation, through the epithelium grid. A new tool to analyse the simulations results has been implemented. It outputs statistics about specific cell states along the simulation; these cell states are defined by the user.
- **Addition of a textual mode to define an epithelial model;** EpiLog relies on a Graphical User Interface (GUI). This tool enables the users that do not wish to use the GUI, to save time by textually defining the parameters of an epithelium and simulation.

The thesis document has the following structure: chapter 2 describes the fundamentals of logical modelling at the cellular and multi-cellular levels, followed by chapter 3 about the existing logical modelling tools, including bioLQM and EpiLog. The chapter 4 and chapter 5 describe the implementation work done in bioLQM and EpiLog respectively. In the chapter 6 the modifications done to the updating schemes in bioLQM are tested within the EpiLog environment with a real case study, namely the segment polarity module, and its impact assessed. Finally, the last chapter 7, discusses the work done throughout this thesis and presents the conclusions and potential future extensions of EpiLog.



# Chapter 2

## State of the Art

### 2.1 Logical modelling of cellular networks

**Logical modelling** was first thought of by S.A. Kauffman (Kauffman, 1969) and its advances were shortly followed by R. Thomas (Thomas, 1973). The basis of this framework is the abstraction of the level of activity of biological components by factoring only binary values, which allows the simplification of molecular parameters that are commonly difficult to find in the literature. The value of each component is controlled by a regulatory function, which depends on the current state of the whole model, the component logical function.

This paradigm can successfully be used to simulate on-or-off signalling (e.g. the transcription or lack off, of a gene or the presence or absence of a gene product). Its implicit simplicity means that it can be used to model large regulatory networks. Furthermore, multi-valued logical modelling was proposed by Thomas and d'Ari, 1990, which allows for finer models that consider different effects for e.g., low, medium or high levels of a component. Multi-valued models may be necessary if different gene activity levels have opposing effects on a target or the target differs with the level of the gene.

Despite the simplicity of the framework, it is still capable of capturing complex properties of regulatory networks (Abou-Jaoudé et al., 2016).

#### 2.1.1 Model definition

A logical model is represented by a directed graph (Abou-Jaoudé et al., 2016), called **Logical Regulatory Graph (LRG)**, composed by a set of  $n$  nodes ( $\mathbf{N}$ ) representing the biological components and a set of edges ( $\mathbf{E}$ ) representing the interactions (regulations) between the nodes. The set of nodes is divided into set of internal nodes ( $\mathbf{I}$ ) and set of input nodes ( $\mathbf{U}$ ). The latter represent external stimuli and are not regulated by any other component, whereas the former nodes influence one another. Each element  $\mathbf{N}_i$  of  $N$ , is associated with:

1. A discrete variable  $\mathbf{X}_i$ , that represents the level of activity of the component  $N_i$ . It can take the values 0 and 1 for classic Boolean models, or any integer between 0 and a fixed maximum ( $\mathbf{Max}_i$ ) if multi-valued.
2. A logical regulatory function  $\mathbf{F}_i$  that defines the evolution of the level of the node  $N_i$ . If the node is in-

## 2. STATE OF THE ART

ternal,  $N_i \in I$ , the regulatory function depends on its regulators (other nodes). This function represents the regulatory interactions that influence the  $N_i$ . If the node is an input,  $N_i \in U$ , the regulatory function is the identity function, the input level remains constant.

### 2.1.2 Model dynamics

The dynamics of a logical model is captured by a state transition graph (STG), (see figure 2.1). The STG shows all trajectories possible between states, within the state space:  $S = \{0, \dots, Max_1\} \times \dots \times \{0, \dots, Max_n\}$ . Each element of the state space set,  $S$ , is a state, which is a vector of  $n$  values,  $X = (X_1, \dots, X_n)$ , where  $n$  is the number of components in the model.

Given a state, the successor states are identified by applying the regulatory functions to each component,  $N_i, i = 1, \dots, n$ . When  $F_i(X) > X_i$ ,  $X_i$  is increased by one, and symmetrically, if  $F_i(X) < X_i$ ,  $X_i$  is decreased by one. A state is stable if for all the components, the regulatory function returns no change, *i.e.*,  $F_i(X) = X_i, i = 1, \dots, n$ .

Due to the discrete and finite nature of the STG, it will always be trapped into a state or a set of states - an attractor. The attractor is either a stable state or a strongly connected component (SCC), a sub-graph wherein no transitions point outwards it. The states that exclusively lead to the SCC but are not part of it, form the basin of attraction of the attractor. Both types of attractors are of immense biological interest as they often correlate with cell phenotypes.

The size of the state space (and thus of the complete STG) grows exponentially with the number of components of the model ( $2^n$  for a Boolean model of  $n$  components). However, depending on the updating strategy, the number of trajectories in the STG can be restricted.

The different strategies to update the values of the model components are referred to as **updating schemes** (also called updaters in the remainder of this thesis), the two most common are:

1. **Synchronous:** every component is updated at the same time step, granting a deterministic system where each state has at most one successor (see figure 2.1c).
2. **Asynchronous:** the components are called to update independently, producing as many successors as the number of updatable components, therefore yielding a non-deterministic dynamics with concurrent transitions (see figure 2.1d).

The simplest attractor, the fixed point (stable state), is easily found as it is a fixed point of the regulatory functions and is independent of the updating schemes, unlike cyclic attractors (SCCs). Since asynchronous dynamics are more complex than the synchronous ones (due to concurrent transitions), the analysis of their properties is harder.

While synchronous updating makes STG simpler to analyse, it is only a meaningful approximation for biological phenomena that occur on the same time scale. However, this is rarely the case, as the different interactions that take place in a biological system show great diversity on the time scales on which they occur, for example, translation and transcription are much slower events than the binding of a transcription factor to its DNA site or protein modification. As such, to realistically model a system that encompasses both events, time separation is necessary, *i.e.*, asynchronicity.

## 2.1 Logical modelling of cellular networks

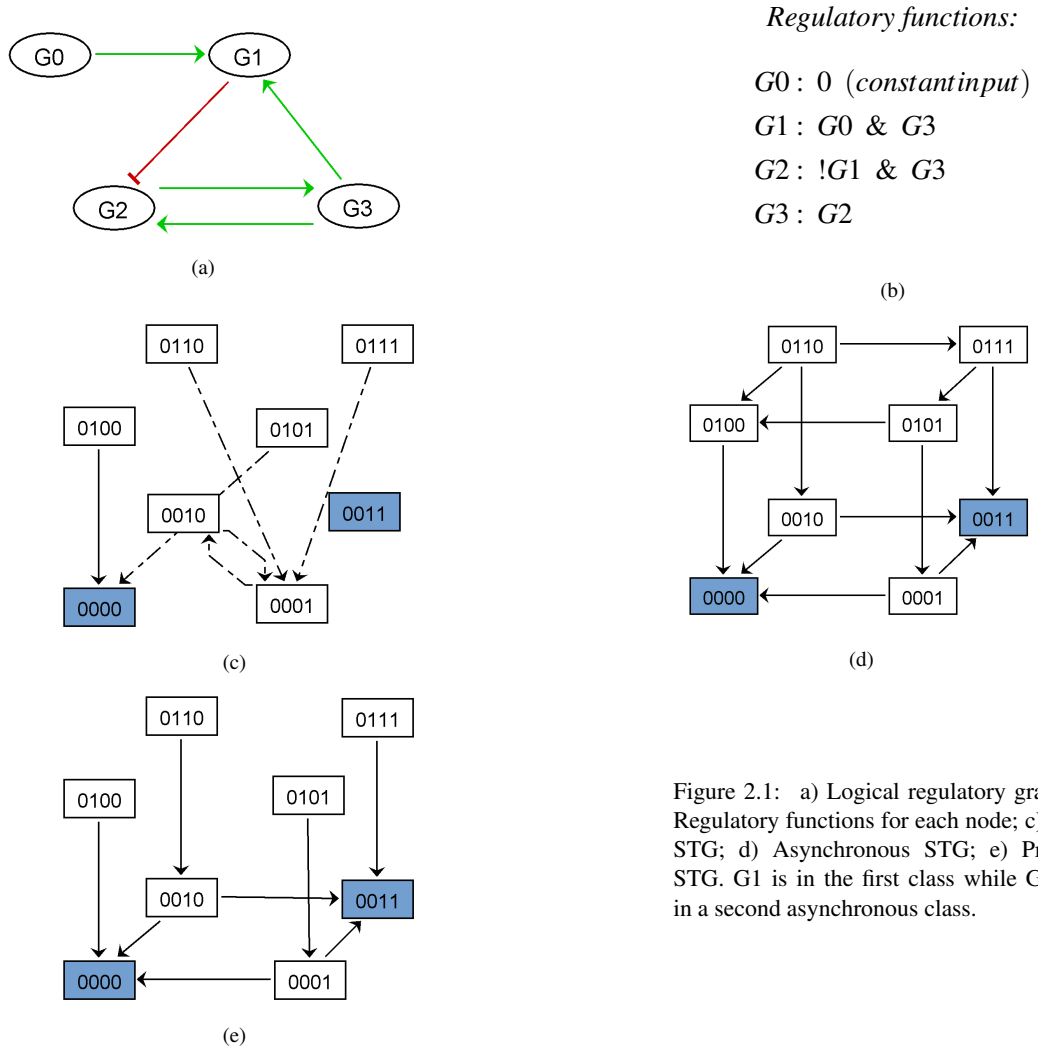


Figure 2.1: a) Logical regulatory graph (LRG); b) Regulatory functions for each node; c) Synchronous STG; d) Asynchronous STG; e) Priority classes STG. G1 is in the first class while G2 and G3 are in a second asynchronous class.

Complete asynchronicity, on the other hand, implies the exploration of biologically non significant trajectories, as not all are possible or observed *in vivo*. This corresponds to the edge case that considers all the concurrent transitions, causing an unnecessary complexity in the ensuing dynamics.

To address the need of breaking synchronicity, while only exploring biologically viable trajectories, some updaters integrate the knowledge of the rate of interactions between components:

1. **Priority classes** (Faure et al., 2006) - amount to the creation of  $n$  classes populated by the model components. Each class has an associated rank and an updating scheme (asynchronous or synchronous), that defines how the class components are updated. If multiple classes have updatable components, the ones belonging to the highest rank are the ones that are updated. The consideration of priority classes greatly simplifies the asynchronous STG, removing non realistic trajectories by integrating knowledge about the interactions kinetics (see figure 2.1e).
2. **Asynchronous updating scheme, random with rates or random non uniform** (Stoll, Viara, et al., 2012) - each component has an associated rate, that mirrors the real rate at which the transition occurs. If a transition is much faster than others, it will have a higher rate. The rate correlates with the

## 2. STATE OF THE ART

probability of a component to be chosen to update. The probability is the ratio between the component rate and the sum of the rates of the other updatable components. Thus, this updating scheme does not simplify the asynchronous STG, but makes certain trajectories less probable. In the sequel, these updating schemes will be called simply random non-uniform or random with rates.

### 2.1.3 Perturbations

Logical models facilitate the definition and the study of the impact of perturbations (mutations), e. g., knockout or over-expression.

A perturbation to a component ( $N_i$ ) is defined by fixing the value of  $X_i$  to a constant, if the component is over-expressed this means setting  $X_i = X_{max_i}$ , otherwise, if it is knocked out,  $X_i = 0$ . Since multi-valued models are considered, it is also possible to fix the value of  $N_i$  to a range contained in  $[0, X_{max_i}]$ .

Any perturbation will restrict the initial conditions and consequently the possible trajectories, basin of attractions and final attractors.

## 2.2 Multi-cellular modelling

Organism tissues show intrinsic orderly complexity that is obtained along the development of the organism. This order corresponds to an arrangement of cells that show patterns of either gene expression (cell state) or cell morphology. These patterns are obtained by multiple biological mechanisms like cell mitosis, growth, apoptosis, signalling, migration, etc. Most of these mechanisms depend on cell signalling, where a state of a cell has a direct impact on the state of their neighbouring cells by secretion of molecules through gap junctions or paracrine signalling. Patterns are also obtained by morphogenesis, where the cell position is altered while its state is not, as for example cell migration or growth. (Salazar-Ciudad et al., 2003).

To study this processes, multi-cellular modelling is a natural progression of intra-cellular modelling.

### 2.2.1 Existing approaches

Different modelling approaches to multi-cellularity have been proposed, the main difference being the use (or lack thereof) of supporting lattices. Lattice models (D. Drasdo et al., 2018) are characterised by the division of the space into homogeneous compartments. A comprehensive revision on discrete approaches was published by Osborne et al., 2017.

Cellular automata (Deutsch et al., 2005) and cellular Potts models (Graner et al., 1992) are two examples that fall in the **lattice based** category. The former limits a cell to a single lattice unit whereas the latter allows for a cell to occupy multiple lattice units, enabling the modelling of morphogenic phenomena such as growth, migration or division, whereas cellular automata require further modifications to deal with these scenarios and are better suited to deal with inter-cellular signalling.

**Lattice free models**, often agent-based models, include cell centred models (Dirk Drasdo et al., 2005) and vertex models (Meineke et al., 2001) where both allow for continuous cell movement and use energy functions that are dependent on neighbouring cells that restrain the cell shape. The former monitors the cell, by the cell center while the latter monitors the cell boundaries. These types of models are ideal to model events dependent on morphogenesis.



### 2.2.2 Cellular automata

This section focuses on Cellular Automata, which is the framework used by EpiLog.

Cellular automata (CA) represent discrete and local dynamic systems. They were first thought of by von Neumann and Stanislaw Ulam in 1940, motivated by the question of self-reproduction of biological organisms. CAs have three main properties, locality, synchronicity and uniformity. CAs are spatially finite, composed of units (automata or cells) that evolve according to transition rules. These adhere to the locality and synchronicity principle, which dictate that a rule depends only on the adjacent cells (neighbourhood) and are applied simultaneously to all the cells. The locally principle specifically defines the neighbourhood as the contacting units. Finally, the uniformity principle asserts that all the cells follow the same set of rules (Schiff, 2007).

Nonetheless, even limited by these three principles, CAs capture extremely well self-emergent complex patterns. Notably, Stephen Wolfram considered an elementary cellular automaton (two-states one-dimensional lattice), and by observing over 256 (Wolfram, 1983) possible updating rules, was able to categorise them in four different patterns of self-organisation – homogeneous, stable or periodic, chaotic and complex (Wolfram, 1984).

To model biological system, this framework can be adapted to consider different cell shapes, neighbourhoods and non synchronous updating schemes. CAs are, for the characteristics listed above, an optimal candidate framework to model large systems such as groupings of cells, *i.e.*, an epithelium.

#### 2.2.2.1 CA updating scheme

As it happens with intra-cellular modelling (see sub-section 2.1.2), at the inter-cellular level, synchronous updating schemes also lead to less realistic dynamics and simulation artefacts. Figure 2.2 illustrates this problem. Consider the simple Delta-Notch model, where the receptor Notch becomes an inhibitor of the expression of its Delta genes when it binds to the extracellular Delta ligand. The simulation of this model with synchronous and asynchronous updating schemes leads to two completely different results: as the first leads to an oscillatory behaviour between all cells expressing Delta or Notch and the second updating scheme leads to a "salt and pepper" pattern observed *in vivo*.

Although CAs are traditionally updated in a synchronous manner, if we would like to model biological systems, alternative updating techniques must be considered to address the previously stated problem.

Fatès (2014) suggested the  $\alpha$ -asynchronous updating scheme, where  $\alpha$  is the probability of each cell to update at each time step. This is the updating scheme used by EpiLog at the multi-cellular level. Bouré et al. (2012) proposed  $\beta$ -synchronism and  $\gamma$ -synchronism, which both consider information loss between neighbouring cells.  $\beta$ -synchronism is characterised by the probability ( $\beta$ ) of each cell to transmit its new state to its whole neighbourhood, the cells are updated synchronously.  $\gamma$ -synchronism is similar to  $\beta$ -synchronism, but  $\gamma$  is the probability of each cell to transmit its new state to each neighbours independently.

By considering the CA Wolfram's rules, and looking at how the updating schemes change the outcome behaviour of each rule, both papers note that the updating scheme choice can have a determinating effect (Bouré et al., 2012; Fatès, 2014). As a result, identifying update schemes that better represent the modelling problem is critical.

## 2. STATE OF THE ART

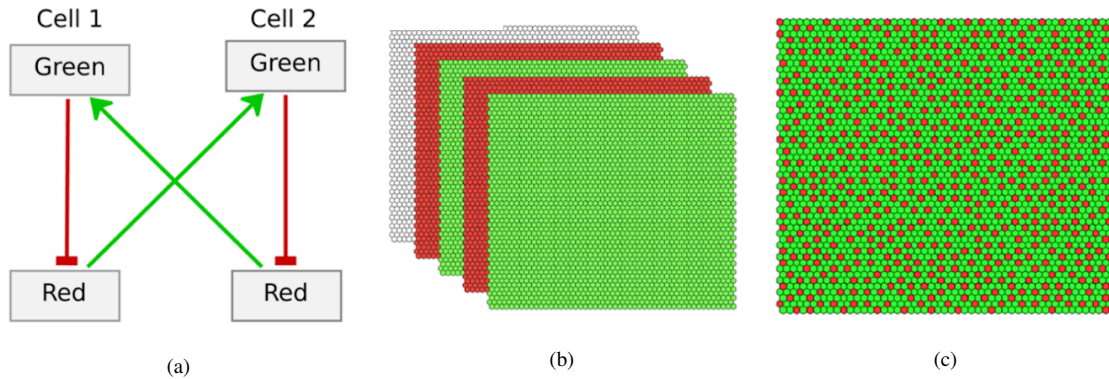


Figure 2.2: Figure taken from (Varela et al., 2019); (a) Simplified Delta-Notch model: when Notch (a cell receptor, here as "Green") interacts with Delta (a ligand, here as "Red"), the Notch intra-cellular domain (NICD) is released and inhibits the expression the ligand Delta. This sequence of events leads to a lateral inhibition mechanism, the "salt and pepper" pattern. (b) Synchronous update leads to an oscillatory behaviour, a cyclic attractor. (c) Asynchronous update leads to the realistic "Salt and pepper" pattern.

### 2.2.2.2 CA - integration

The dynamics of a traditional cellular automata is driven by a local rule system that defines the cell neighbourhood as the cells at a maximum distance of one. However, to model biological systems, it is unrealistic to consider only the adjacent cells. In fact cell-cell communication is often done at a distance, *i.e.*, paracrine signalling. Therefore, to realistically model these systems, the concept of the neighbourhood must be expanded to encompass neighbourhoods with a bigger radius than one.

The rules of a traditional CA depend on the states of the cells, either 0 or 1. With the association of a logical model to each cell, rules can be more complex and depend on one, multiple or even all components of the cell logical model. Furthermore, the number of neighbouring cells required to influence the state may be specified in the rule. The number of cells is a crucial point, as in cellular systems one cell signalling might not be enough and a cumulative effect of multiple cells sending the same signal is needed to create a change in the system.

Fauré et al. (2014) described and modelled the *Drosophila* eggshell patterning. As the patterning is highly affected by cell-cell communication, the authors considered an epithelium grid, where each cell includes a logical model, which is affected by its neighbouring cells. This is, the intra-cellular logical model was extended to the multi-cellular level by including an **integration input** which is directly dependent on the states of the neighbouring cells. This definition was the basis for the development of EpiLog.

EpiLog defines integration nodes, see section 3.2.1.1 that either receive a signal or not. The input receives the signal if the rule associated with the signal is true, this rule can depend on the radius of the neighbourhood, on the expression levels of components in the neighbouring cells, and also on the number of cells in a given state. All three parameters can have a minimum and a maximum threshold. EpiLog relies on combinations of constraints, where a constraint can depend on everything that was described above.

## Chapter 3

# Modelling Tools

There are numerous logical modelling tools available. These tools differ in terms of software environment, modelling strategies, and final information provided to the modeller. Because they have different features, these tools are frequently complementary. As a result, a need arose for modellers more easily exchange models, consequently, the exchange format Systems Biology Markup Language (SBML) was proposed.

The SBML (Hucka et al., 2003) is a XML-based format designed to represent models of biological processes, allowing the exchange of models across tools.

More recently the Consortium for the development of Logical Models and Tools (CoLoMoTo) (Naldi, Hernandez, Levy, et al., 2018) has created the SBML qual (Chaouiya, Bérenguier, et al., 2013a) as an extension of SBML to support logical modelling. CoLoMoTo has also created the LogicalModel library or bioLQM, a Java toolkit, presented in the next section.

### 3.1 Logical modelling tools

#### 3.1.1 BioLQM

**BioLQM** (Naldi, 2018), <http://www.colomoto.org/biolqm/>, is a Java toolkit that was developed to unify and facilitate cooperation in the qualitative modelling community (Naldi, Hernandez, Levy, et al., 2018). The toolkit enables the definition of logical models as well as possible modifications, e.g., perturbations, model booleanization (for multivalued models), etc. Multiple updating schemes are also implemented in bioLQM, facilitating the retrieval of successors states when a state, the regulatory functions and the updating mode are given as input. Thus it does not fully simulate the dynamics of a model but can be used for that end by other applications. This is the case of GINsim (Naldi, Hernandez, Abou-Jaoudé, et al., 2018) and EpiLog (Varela et al., 2019). The identification of stable states is achievable with bioLQM by analytical means using decision diagrams (Naldi, Thieffry, et al., 2007).

The **bioLQM package** is divided in sections, each with a specific objective:

- **Model modifiers** section, includes the implementations of model booleanization, reduction, perturbation, reversal and subsampling.
- **Service** section, allows for bioLQM to be used through the API or through the command line.

### 3. MODELLING TOOLS

- **Tool** section, has the implementation of fixed points, and attractors computation, as well as trap spaces identification. It also includes the simulation tools, all the updating schemes and simulation engines.
- **IO** section, supports the import and export of models between different formats, including Petri nets, truth tables, SBML format, and multiple other formats of modelling tools.
- **Widgets** section, includes the support GUI for the Priority Classes definition.

#### 3.1.2 Other logical modelling tools

All the tools listed below support logical model definition. As these tools are often complementary, and have distinct characteristics, the descriptions that follow focus on their strengths, rather than being exhaustive.

**MaBoSS** (Markov Boolean Stochastic Simulator) is a command-line logical modelling tool that implements the random with rates updating scheme. It employs the Gillespie algorithm, which approximates a continuous-time Markov process (Stoll, Viara, et al., 2012) where the probabilities are determined by rates associated with the transitions affecting the nodes (Stoll, Caron, et al., 2017).

**BoolNet** is a R package that allows the manual definition of large logical models and offers methods to generate models from gene measurements over time. It allows the detection of attractors and state trajectories. It supports the synchronous, asynchronous and probabilistic updating schemes (Müssel et al., 2010).

**The Cell Collective** is a logical modelling tool that distinguishes itself for being a server-based online tool. Its main objective is to allow modellers to share biological data and models. It includes a repository for modellers to share data of biological processes, the "Knowledge Base" component. This information can then be more easily integrated to create new models and compare results with old ones. The tool also allows for "real time simulations", where it is possible to observe the average activity for each biological species (node). It is also possible to do a post-simulation dynamical analysis, obtaining dose-response curves between two model components (Helikar et al., 2012).

**SQUAD** (Standardized Qualitative Analysis of Dynamics) is a Java software that can compute attractors, stable states and outputs components concentration over time. It also enables the definition of model perturbations such as knockouts and gain of function. This tool takes a Logical Regulatory Graph (LRG) and computes the system's steady states. The LRG is then transposed into ODEs. The steady states are computed using the ODE continuous model, which uses information about the logical model to help compute them (Di Cara et al., 2007).

**GINsim** (Gene Interaction Network simulation) is a Java software with a powerful GUI that embeds the bioLQM package. Among other features, it can define multi-valued logical models, it can also booleanize and reduce models. Aside from the asynchronous and synchronous updating approaches, this tool supports priority classes. It outputs a state transition graph, and can compute a strongly connected graph and runs simulations to determine attractor reachability. It also allows for perturbation analysis (Naldi, Hernandez, Abou-Jaoudé, et al., 2018).

## 3.2 Multi-cellular modelling tools

### 3.2.1 EpiLog

**EpiLog** (Varela et al., 2019) (Epithelium Logical modelling), <http://www.epilog-tool.org>, is a free Java software with a graphical interface that implements the CA and logical modelling frameworks allowing the study of epithelial pattern formation. The lattice is formed by hexagonal cellular automaton, and each cell behaviour is dictated by its logical model and by the input signals from the neighbouring cells and/or from other positional cues.

EpiLog uses the Java library bioLQM (Naldi, 2018), to handle the intra-cellular network associated with each cell. The intra-cellular models are not defined in EpiLog and have to be imported from a SBML file (Chaouiya, Bérenguier, et al., 2013b) that can be defined with other tools, for instance, GINsim.

The epithelium in EpiLog is represented as a lattice of hexagonal shaped cells, implying a direct neighbourhood of six cells, emulating observed cell behaviour *in vivo* (Gibson et al., 2006). EpiLog extends CA to allow the simulation of biological signals from non-adjacent cells. Biological noise is also considered since different signal thresholds can be defined for the inputs, in the form of either cell states or number of cells. This is achieved by defining different (signalling) integration functions to each input node.

#### 3.2.1.1 Model definition

The steps to define a model in EpiLog are described below. For further details, the documentation may be consulted at <http://www.epilog-tool.org/documentation>.

**Topology:** The first step in defining a multi-cellular model (epithelium) is to specify its topology: dimensions – size of the 2D grid - and boundary conditions - if borders are disconnected or connected in both or in one of the dimensions.

**Intra-cellular modelling:** Having the model grid defined, it is possible to associate different intra-cellular logical models to different cells, allowing for the representation of biological variation. The different intra-cellular models are loaded as SBML files. The intra-cellular model must have at least one input node to allow for the integration of external signalling.

**Input definitions:** The inputs of the intra-cellular model must have their types defined, either as: **positional**, representing positional cues (from the environment) that remain constant, or as **integration input**, serving as signalling connections between the cells of the grid.

For every possible value,  $X_i$ , of the integration input,  $N_i$ , an integration function  $G_{i,j}(X)$  is defined.  $j$  is between  $[0, Max_i]$ , where  $Max_i$  is the maximum value that  $N_i$  can take. These logical functions define the conditions in which the integration node changes its value to the target value  $j$ . The first integration function of  $N_i$ ,  $G_{i,j}(X)$ , to be evaluated is the one with the highest target level, where  $j = Max_i$ . If it returns false, then  $j = Max_i - 1$  will be evaluated and so forth until the function returns true, if none returns true than the integration input will be set up to 0.

### 3. MODELLING TOOLS

Integration functions are logical combinations of cardinality constraints (CC), where a CC describes the minimum and/or maximum numbers of cells with a minimum and/or maximum value at some minimum and/or maximum distance.

Consider this example:  $(\{g_1 : 2, \max = 2\} \& \{g_2 : 1[ : 4], \min = 2\})$ ; For the integration function to return true, the following conditions have to be met: at most 2 cells at a distance 1 with  $g_1$  at minimum level of 2, and at least 2 cells at a distance at most 4 with  $g_2$  at the minimum level 1.

**Initial conditions:** The initial values of the components need to be specified for each cell, which by default are 0.

**Perturbations:** This consists in setting a component value, either to a fixed number or to a range, allowing for knock-out simulations (setting the value to 0) and ectopic activity (setting the value to the maximum).

**Cellular updating mode:** The updating scheme for the internal components can be specified, by default it is synchronous. EpiLog supports priority class updating, where the class components are updated synchronously. Additionally, it allows a component to be split into increase and decrease updates. However, within each class, the updating mode is synchronous.

**Epithelium model update:**  $\alpha$ -asynchronicity is used, where  $\alpha$  is the ratio of cells to be updated. If  $\alpha$  is 1, this corresponds to a fully synchronous dynamics while  $\alpha$  equal to 0 represents a fully random asynchronous dynamics, *i.e.*, a single updatable cell is randomly selected at each simulation step.

#### 3.2.2 Other multi-cellular modelling tools

Besides EpiLog, at least two other tools make use of logical modelling at the cellular level to model multi-cellular systems, in different ways:

**PhysiBoSS** (PhysiCell-MaBoSS) (Letort et al., 2019) is a multi-cellular modelling tool that combines the MaBoSS software for intra-cellular Boolean modelling with the PhysiCell package for multi-cellular agent-based modelling. As this tool does not depend on a lattice, it is useful to model growth and cell mobility.

**CoGNac** (Chaste and Gene Networks for Cancer) (Rubinacci et al., 2015) is a tissue modelling tool that makes use of the Chaste plugin (Mirams et al., 2013), which enables both continuous cell and cell-based modelling. CoGNac uses the latter framework to apply random noisy Boolean networks, to represent gene regulation networks in on or off lattices. It focuses on the study of cell differentiation and cell cycle length to model colonisation of tissues by different types of cells which are characterised by different attractors.

While both tools model population of cells, they focus on the physical properties that are emergent from the attractors of the logical model of each cell. EpiLog distinguishes itself by introducing inter-cellular signalling and capturing multi-cellular dynamics that emerge from the interconnection of cells.

## Chapter 4

# BioLQM extensions

The bioLQM version that resulted from the extension described in this chapter, can be found at: <https://github.com/fistorres/bioLQM>.

### 4.1 Previous Priority Updater

EpiLog uses the priority classes updating scheme described in section 1. The implementation of bioLQM, used by EpiLog, is illustrated in figure 4.1, where it is called the priority updater. In contrast to the original definition (in which a class equated a single group), each priority class is possibly composed of several groups. Each class can be updated by the asynchronous and synchronous updating schemes.

As noted in the previous 2.1.2 section, the synchronous updating cannot truly simulate biological events because they do not occur on a single time scale. While asynchronous updating solves the problem, it is an exhaustive, non-optimal solution since it generates a large number of concurrent trajectories that do not occur *in vivo*, thus is computationally costly. As such, it becomes important to introduce new updating schemes that incorporate biological knowledge in the ranked classes, hence limiting the number of trajectories in a biological coherent way.

### 4.2 Priority updater integration with other updating schemes

**Proposed solution:** Modify the priority classes so that each group accepts an updating scheme, and the priority updater to accept these groups and use their updating scheme to compute the group successor, providing the end user more flexibility when defining model dynamics. This extension makes use of the bioLQM updaters and the previously created ranked class groups. It entails assigning each group an existing updating scheme, such that the rank successors are the union of the group's successors (see figure 4.2).

Table 4.1 summarises the differences between the current available version of bioLQM and the bioLQM with the proposed changes implemented.

BioLQM updaters are classified into three types: deterministic updaters, multiple successors updaters and random updaters. Note that all multiple successors updaters can be made random when a filter is added that randomly selects one of the successors (RandomUpdaterWrapper class included in figure 4.3).

## 4. BIOLQM EXTENSIONS

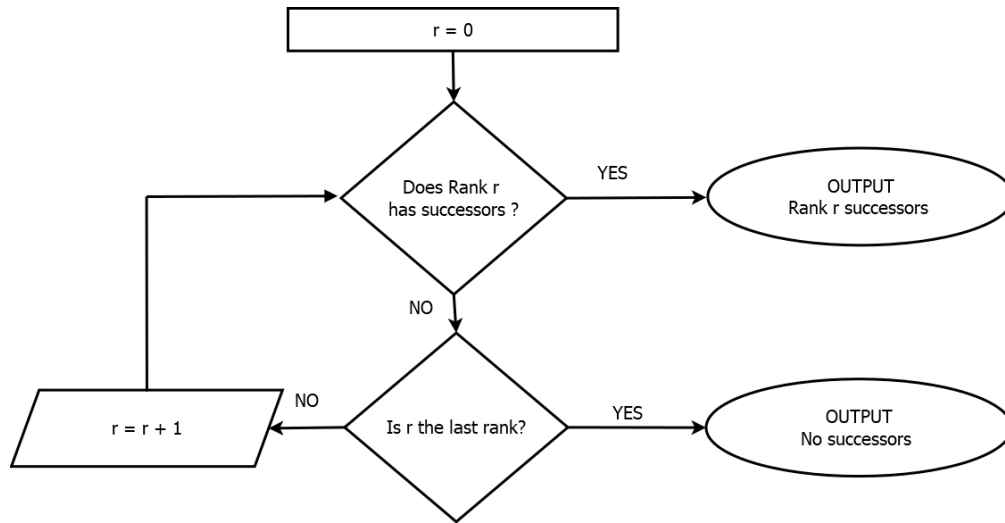


Figure 4.1: Flow diagram of the priority updater, ranks are visited in order. If a rank has valid successors, the priority updater stops and outputs them.

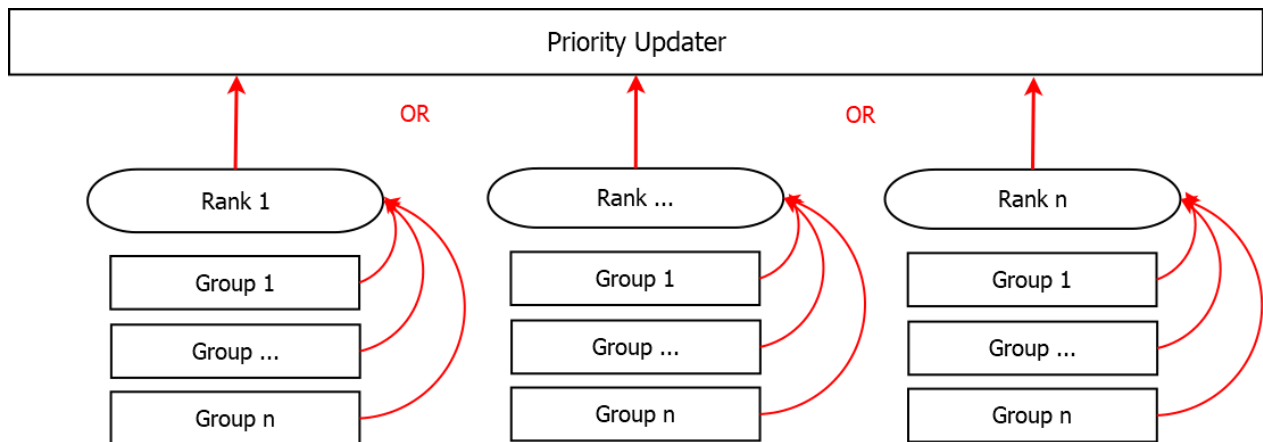


Figure 4.2: Scheme representation of Priority classes implementation in bioLQM. The ranked classes successors are the union of their groups successors when multiple groups are available. The priority updater outputs only the successors of one ranked class, the one with highest rank and valid successors.



### 4.3 Approach and Priority updater architecture

Table 4.1: Differences in bioLQM priority updater before and after the implementation of the proposed modifications and EpiLog restrictions on the use of post implementations bioLQM. Previously, the available updating schemes were coupled with each rank, whereas now they are coupled with the groups.

		bioLQM v0.7.1	bioLQM v0.8.1	EpiLog use
Priority's classes structure	Ranks	✓	✓	✓
	Groups		✓	
Single successor updaters	Synchronous	✓	✓	✓
	Random uniform		✓	✓
	Random non uniform		✓	✓
Multiple successor updaters	Asynchronous	✓	✓	
	Complete		✓	

### 4.3 Approach and Priority updater architecture

The **PriorityUpdater** is the java class in charge of outputting successors in accordance with the priority classes updating scheme, this class receives an instance of the **PCRankGroupsVars** class, which represents the priority classes.

The **PCRankGroupsVars** java class is responsible for assembling the ranked classes and managing their ranks. The ranked classes are implemented in the **RankedClass** java class and in turn manage the groups, which are implemented under the java class name **RankedClassGroup**. The relationship between these three java classes and the **PriorityUpdater** java class is depicted in the figure 4.3. The bioLQM **PCRankGroupsVars** class also supports split components, *i.e.*, the decrease and increase of a component value can be dealt with individually, the split components information is dealt by each updater, the figure 4.4 shows the components split.

Because each updater in bioLQM inherits directly or indirectly from the **LogicalModelUpdater** (Java) interface (as shown in figure 4.3), it becomes easy to assign one updater to each group. The **RankedClassGroup** can be made to accept an instance of the **LogicalModelUpdater**.

The modifications in bioLQM that were needed to implement the assignment of one updater to each group were made considering not only the need to maintain compatibility with the **PriorityUpdater** class that uses the **PCRankGroupsVars** object to compute successors but also with the **PriorityPanelClass**.

The **PriorityClassPanel** is the java class that corresponds to a graphical user interface. This interface is supported by Swing, a java toolkit for the development of GUIs. Its purpose is to aid the user to define ranked classes and their groups. This class also suffered major changes, which are depicted in the figure 4.4 that shows this panel post implementation, and includes a short description of its usage. Although bioLQM does not focus on the development of GUIs, its implementation in bioLQM allows any tool that is developed with Swing to integrate this panel, such is the case with GINsim and EpiLog.

**Multiple successors and single successor updating schemes** With the extension of the priority updater to accommodate additional updaters, it becomes necessary to give bioLQM users the choice of restricting the groups updaters to multiple successors updaters or single successor updaters. This is accomplished in the **PriorityClassPanel** constructor, where the programmer can choose between multiple successors updaters, single successor updaters or both.

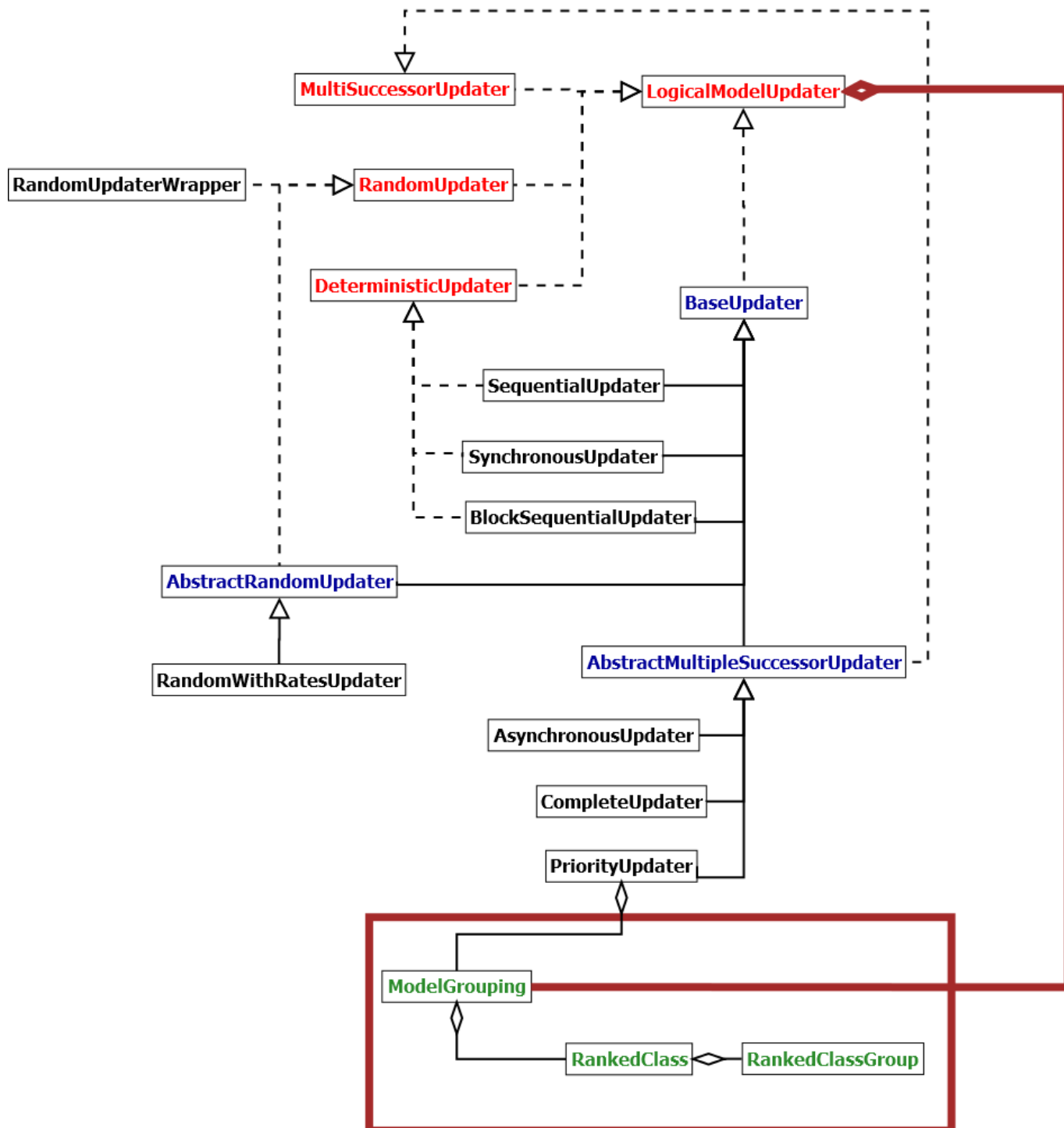


Figure 4.3: Representation of the updaters implementation in bioLQM. Blue are (Java) abstract classes, red are (Java) interfaces, black are updaters (Java) classes and in green is the model grouping class and its subclasses. The diamond shape means that source class is accepted as an argument of the target class (or interface). Triangle shape with full/dotted lines means the target is a class/interface and the source extends/implements it. The dark red line represents what was added in this thesis, and the dark red squared shows the classes that were modified to support the modifications in the PriorityUpdater.

### 4.3 Approach and Priority updater architecture

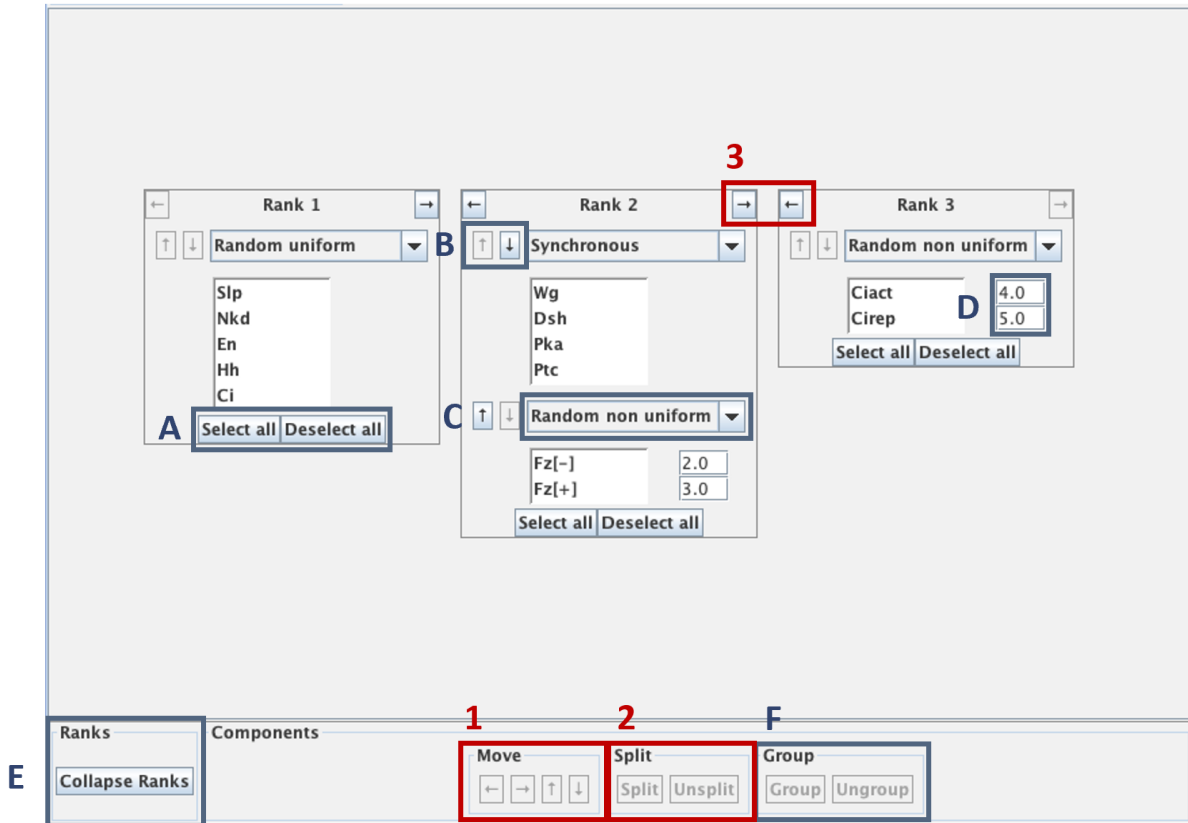


Figure 4.4: View of the PriorityClassPanel. Boxes and numbers in red highlight the GUI components already present in bioLQM before the changes. 1 - Move operations, the selected components can be moved to different ranks and/or groups; 2 - Split operations, the selected components can be split or unsplit; 3 - Ranked Class move operations, allow for the changing of rank order. The boxes and letters in blue highlight the GUI components that were added or greatly modified. A - Select and unselect all components in the ranked class; B Group move operations, allow for the changing of group order (note this is for aesthetic purposes only); C - Group updating scheme combo box, allows the selection of the group updating scheme; D - When random non uniform updating scheme is selected, a box appears which allows the definition of the rates for each group component. E - Collapse Ranks button, collapses all ranked classes in a single one, keeping the groups structure.

To support this, the UpdaterFactory class was added, it is a helper class for both the PCRankGroupsVars and PriorityClassPanel classes. The updaters objects are generated given the updater name and/or PCRankGroupsVars object and/or rates (if RandomWithRates updater). The updaters are defined in the "Updaters.properties" file, which separates the "multiple successor updaters" and "single successor updaters", ensuring consistency between the PCRankGroupsVars and PriorityClassPanel classes.



## Chapter 5

# EpiLog extensions

The EpiLog version that resulted from the extensions described in this chapter, can be found at: <https://github.com/fistorres/epilog>.

### 5.1 BioLQM extensions integration with EpiLog

The bioLQM extensions presented in the previous chapter were motivated by the use of the Priority Updater and its related widget by the EpiLog tool. Post bioLQM extensions, EpiLog had to be altered to support the new version.

As shown in the table 4.1, EpiLog uses the extended updater in a “single successor updaters” mode and a “single group ranked class” mode. Indeed the goal is to extended the cellular updating scheme beyond the synchronous one. However since EpiLog deals with a potentially high number of cells, it has to be limited to single successors updaters as multiple successors would render the simulation intractable.

The integration of the new version of the PriorityUpdater and its related classes as well as the GUI, was easily achieved since most changes occurred in bioLQM. In EpiLog, the definition of the priority updating scheme is made through the GUI, in the cellular model update tab (see figure 5.1). It is in this tab that the PriorityClassPanel is embedded and where EpiLog defines the parameters needed to restrict the Priority Updater to only accept one group per rank, essentially removing the notion of group, and to only accept ranks with single successors updating schemes associated with them (see table 4.1). Other Java classes like the Epithelium class also had to be altered to support the changes in bioLQM.

### 5.2 Tracking simulation phenotype data

Currently, EpiLog simulation tool focuses on the observation of patterns, outputting graphical representations of the epithelium defined. It also identifies cyclic attractors and stable states of the grid. This output, while extremely meaningful, is limited in its capacity to be used by the user to perform quantitative analysis of the defined model’s behaviour.

Subsequently, we created a new tool, that allows the modeller to acquire a view of the cell states distribution across the simulation time-steps, which not only furthers the study of cell patterns, it also gives the modeller an alternative view of the dynamics of the system being modelled focused at the level of the

## 5. EPILOG EXTENSIONS

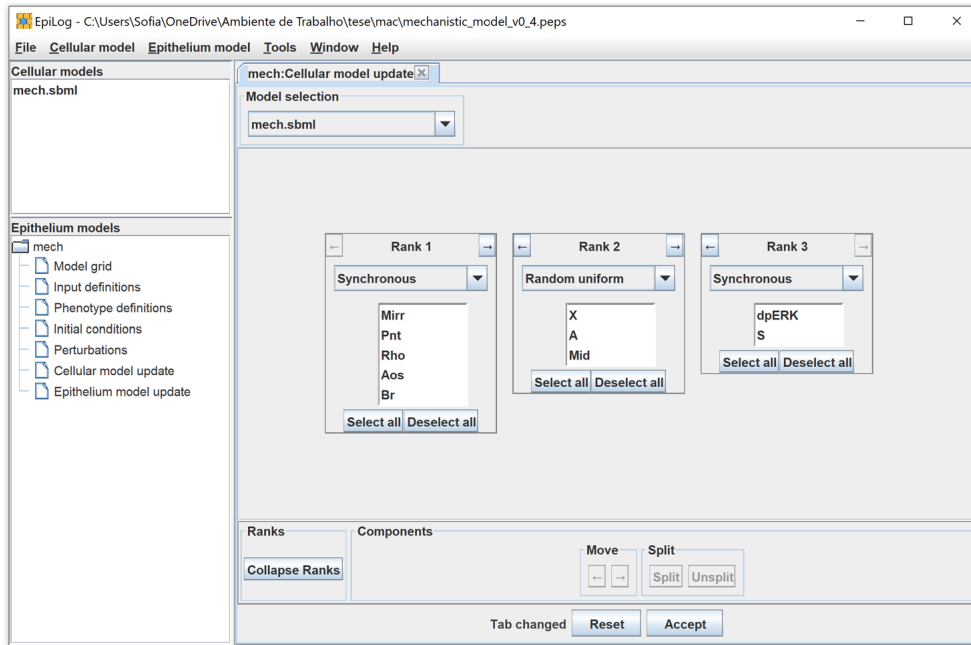


Figure 5.1: View of the Cellular model update tab in EpiLog. Multiple groups ranked classes are not allowed, as such the components that allowed group manipulation do not appear. Below the Priority class panel, EpiLog that notifies the user in the bottom bar if the priority classes have been changed and allows the user to reset them to the last saved version or to save the current definition. In this case, if the “Collapse Ranks” button is used, the ranks will be collapsed into a single group.

different cell states.

### 5.2.1 Proposed changes

The epithelium grid is a  $m \times n$  matrix of cells, each with its own logical model. At every time step the simulation updates the grid, allowing the states of each cellular model to be registered.

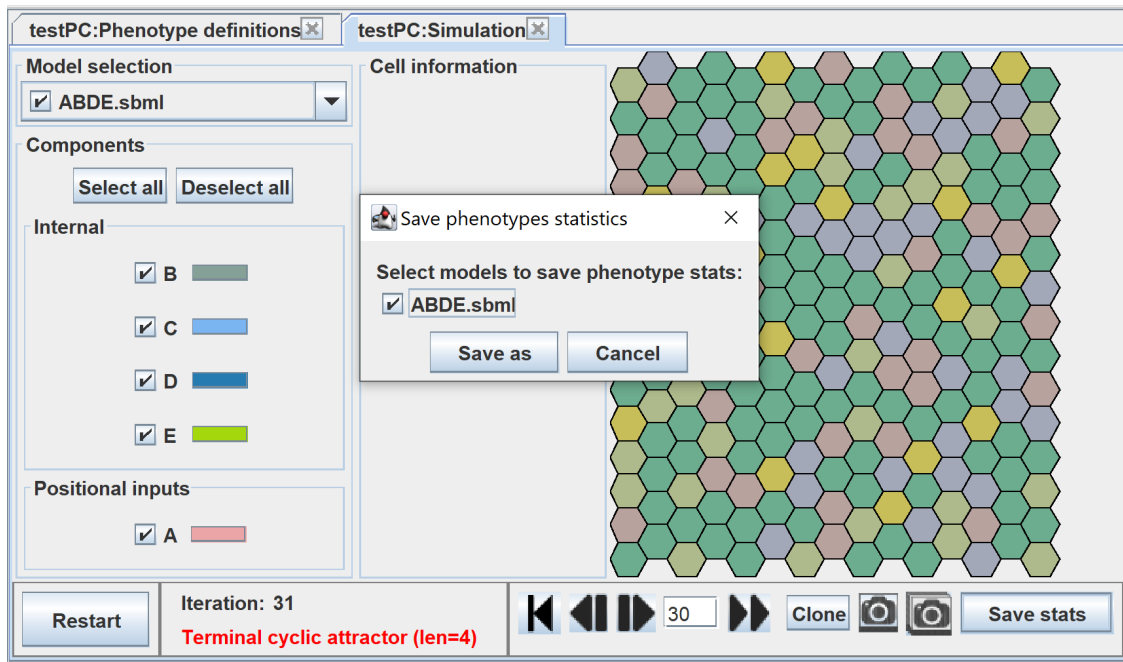
The proposed extension is to give the user a tool that outputs data about the cells states across the simulation time-steps. The states for which the data is to be saved is defined by the user in a new EpiLog tab and retrieved in the simulation tab at any point of the simulation.

While it would be ideal to track all states throughout all time-steps, doing so would be computationally expensive, especially for large grid cells, and would slow down the user experience. One solution is to give the user the possibility of specifying which states wishes to track. Not only this optimises the tool, it also makes sense, as not all cell states have the same biological significance for the user. This way, the user is able to specify cell states that may represent gene expression signatures, which are patterns of expression characteristic of certain cell types that are specific of certain biological processes.

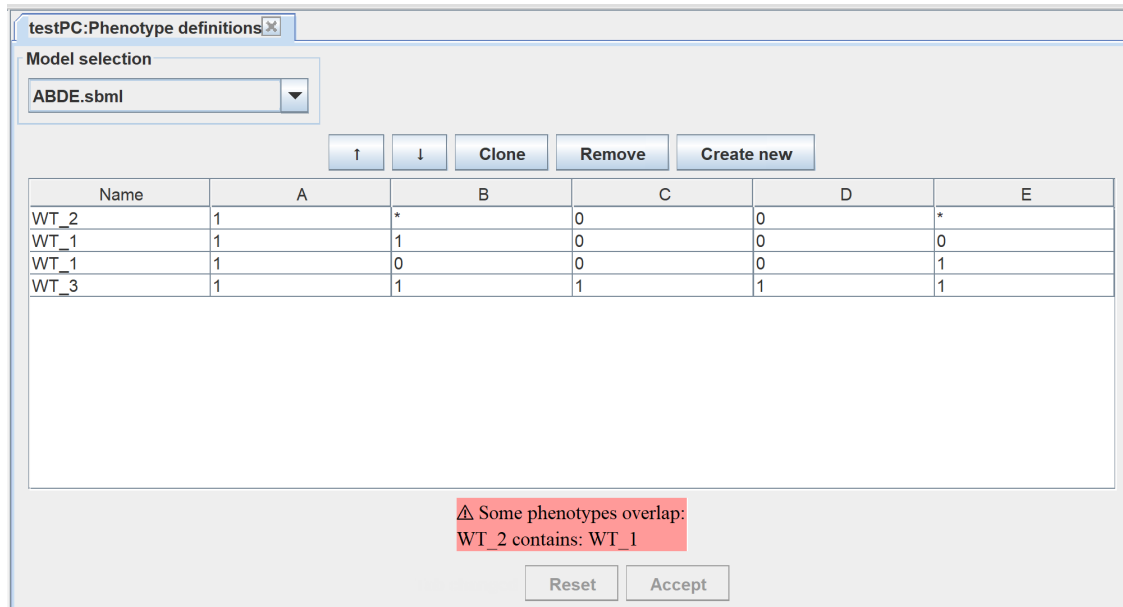
With this in mind, a new tab/Epithelium parameter was added to EpiLog to support the definition of states to track, which from now on will be referred as phenotypes. A **phenotype** is defined by (a set of) pattern(s), each formed by a succession of cellular model components values and/or wildcards, that is to say a (set of) model state(s).

The set of states defined by a phenotype might overlap with the set of states of another phenotype. In another words, a cellular state might belong to multiple phenotypes and be counted as such in the results. In that situation, a warning is issued by EpiLog.

## 5.2 Tracking simulation phenotype data



(a)



(b)

Figure 5.2: (a) View of the Simulation tab in EpiLog. The “Save stats” button was added to the tab, bottom right. The button when clicked shows the “Save phenotypes statistics” dialog at centre, which the user can use to select which models the data should be saved for. The data is referent to the phenotypes defined in the “Phenotype tab” previous to the simulation. (b) View of the Phenotypes definitions tab in EpiLog. A large table is at the centre of the tab where the user can define the phenotype’s patterns. A phenotype is the union of patterns defined with the same name. Above the panel are buttons, that from left to right, allow the user to reorder, clone, remove and create new patterns. When defining a pattern, the interface does not accept component values outside of the component declared range of values, automatically changing the user value to the closest valid one. In this example, phenotype “WT\_1” is formed by the union of two patterns, both named “WT\_1”. The interface issues a warning when some phenotypes overlap: here “WT\_2” contains “WT\_1”.

## 5. EPILOG EXTENSIONS

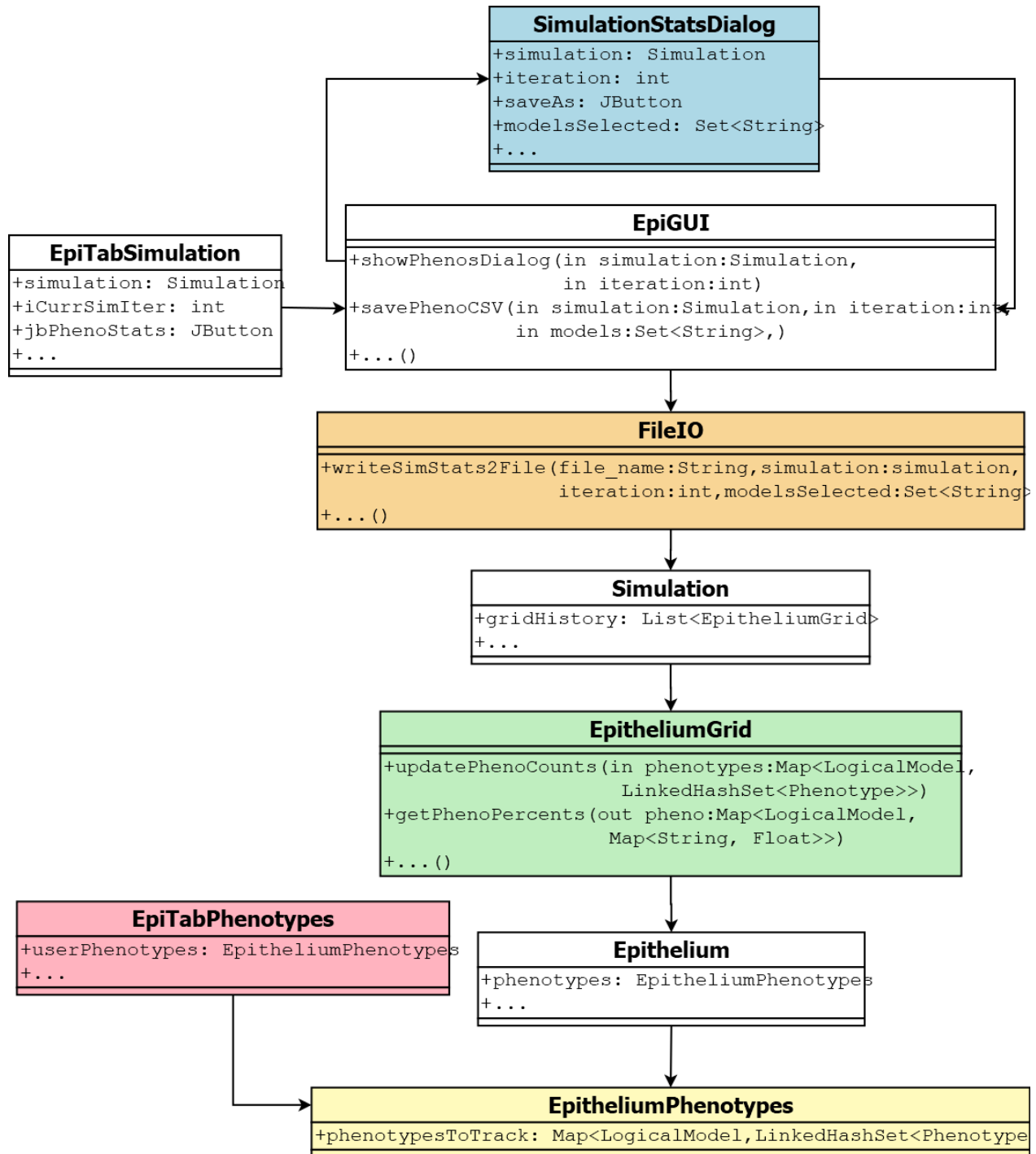


Figure 5.3: Diagram showing the architectural path between clicking on the “Save stats” button on the Simulation tab (EpiTabSimulation), and the export of the file on the FileIO class (in orange). When the button is clicked, the EpiGUI class creates and shows the dialog (SimulationStatsDialog - in blue). The user then is able to select which models he wants to save the data for and this information is given to the EpiGUI class which calls the FileIO class. The FileIO class receives a simulation object, which contains all the information needed to calculate the phenotypes percentages. The simulation object contains a history of all the epithelium grids (EpitheliumGrid - in green), which compute the phenotypes percentages. The phenotypes are associated with each Epithelium, which the EpitheliumGrid has access to. The phenotypes are implemented in the EpitheliumPhenotypes class (in yellow) and are defined through the Phenotype tab (EpiTabPhenotypes - in pink).



## 5.3 Textual mode to define the parameters of an Epithelium

### 5.2.2 Interface approach and phenotype architecture

The outputted data consists of a time-series of phenotype percentages, *i.e.*, number of cells in the epithelium with that phenotype over the total number of cells. It can be obtained by clicking in the “Save stats” button on the simulation tab (see figure 5.2a), which will launch a dialog prompting the user to select in which directory the CSV file is to be saved to. Furthermore, when multiple logical models are used in the Epithelium grid, the user can also choose which logical models it wants to save the data for.

This action can be done at any time step of the simulation, and will retrieve the phenotype statistics from the beginning of the simulation up to the simulation step when the user clicked on “save stats”. This way the statistics are not computed during each time step of the simulation not slowing it down.

As mentioned above, the definition of the phenotype is done through a tab, the Phenotype definitions tab (see figure 5.2b). This tab is accessed through the left tree panel, the “Epithelium models” panel, the same way the other tabs are opened.

At its centre it has a editable table that shows the components from selected logical models, where the user can introduce values for the components. If these values are out of range for the component, then the interface will convert it to the closest component valid value. Note that a phenotype is defined by the union of the patterns associated with the same name. The interface will also warn the user when a phenotype overlaps with another.

The architecture of the program suffered a few alterations and some additions in order to be able to accommodate the phenotypes addition, figure 5.3 shows all the steps between defining a phenotype with the GUI, and the output of a file with the phenotype data.

Essentially, three new classes were added to EpiLog. The **EpitheliumPhenotype** (which is an attribute of the Epithelium class), where the phenotypes are saved, the **EpiTabPhenotypes** that supports the definition of phenotypes through the GUI, and the **SimulationStatsDialog** where the phenotypes are retrieved from. Besides the new classes, the FileIO Java class was modified to support the writing of the .csv file, the EpitheliumGrid to compute the phenotype statistics and other classes suffered minor changes.

## 5.3 Textual mode to define the parameters of an Epithelium

EpiLog is a graphical tool for the definition and simulation of grids of cells named Epithelium, this means that any modeller can easily use this tool. On the other hand, when the cellular logical models have many components, defining the Epithelium through the graphical interface might become cumbersome.

For this reason, the possibility for the user to textually define the Epithelium parameters was introduced.

**Interface approach and architecture of text mode** EpiLog has a left side panel labelled “Epithelium models”, visible in the figure 5.4, that displays the user-defined Epithelia. Under each Epithelium, a tree structure displays the Epithelium’s parameters. By double clicking on any of these tree nodes, the user opens a tab. Instead, a right-click brings up a “Edit as text format” popup, which when clicked opens a dialog box where the user can define the Epithelium parameter textually.

This mode does not support the definition of the “Model grid”, “Initial conditions” and the “Perturbations”. Because the definition of these parameters is done for each cell of the Epithelium grid, making the textual mode not useful since it would require the textual definition of each cell.

## 5. EPILOG EXTENSIONS

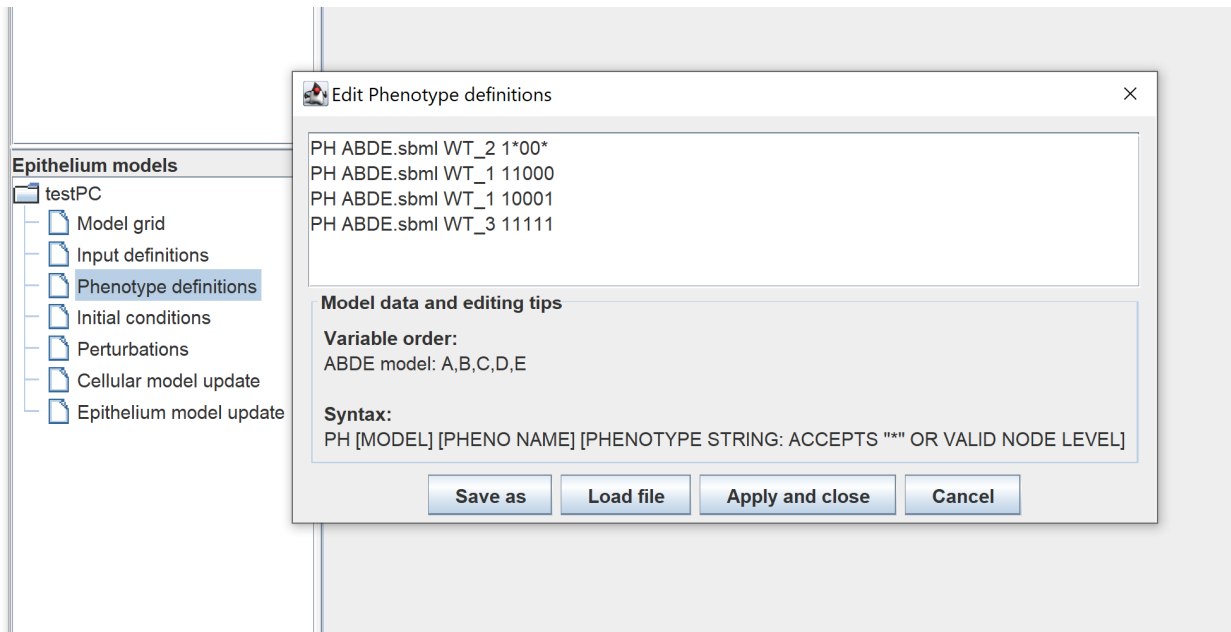


Figure 5.4: View of the “Edit Phenotype definitions dialog”. The dialog shows up when right clicking on the “Phenotype definitions” (highlighted in blue) on the “Epithelium models” panel. The “Cellular model update”, “Input definitions” and the “Epithelium definitions” are also supported by the “Edit by text” mode.

This dialog (see figure 5.4), is composed of a text-box that presents the current definition of the epithelium parameter being edited. Under the text-box there is a panel called “Model data and editing tips” with all the information the user needs to be able to define the epithelium parameter, this includes the syntax used by the EpiLog parser for the specific epithelium parameter, and when relevant, the logical model names and their components. The user can write in the text-box, or alternatively, upload a text file with the textual definitions. These can also be exported as a file. The user can only apply the textual definitions to the Epithelium if they comply with the fixed syntax; otherwise, the text-box appears in red, and the user is unable to apply the changes.

This approach generalises the EpiLog input and output operations, *i.e.*, the reading and writing of a EpiLog project. The EpiLog project is saved as a peeps archive, which the new Epithelium definition mode relies on. This file syntax did not change but because of the textual editing mode, verification of the user syntax had to be added and each Epithelium parameter writing and reading became independent.

## Chapter 6

# Evaluation of new updating schemes with the Segment Polarity model

### 6.1 Background

This chapter discusses the impact of the new updating schemes added to EpiLog. To this intent, the segment polarity model proposed by (Sanchez et al., 2008) is considered as a case study. It relates to the segmentation of *Drosophila*, which is a well-studied phenomenon in development biology. This process involves inter-cellular interactions to maintain patterns of gene expression, which lead to the segmentation. The (Sanchez et al., 2008) model includes a logical intra-cellular model, and its extension to a six cell stripe.

**The segmentation of *Drosophila*** larva is preceded by the para-segmentation which first becomes visible after the formation of the blastoderm, during gastrulation (Johnston et al., 1992). A segment is formed by the anterior region of a para-segment plus the posterior region of the next para-segment. The para-segmentation is a process initiated by the maternal genes which activate the zygotic genes. The latter are divided into three categories, the first zygotic genes that are activated are the gap genes which in combination with the maternal genes activate the pair-rule genes which finally, activate the segment polarity genes (Martinez-Arias et al., 1985; Jaeger, 2011).

The segment polarity genes ultimately dictate if a cell expression pattern is dominated by the *engrailed* gene (*en*) or the *wingless* gene (*wg*). It is this differentiation that defines the boundaries of the para-segments, as the *En* cells will form the anterior region of each para-segment and the *Wg* cells will form the posterior region (Heuvel et al., 1993).

Initially, the expression of either *engrailed* or *wingless* depends on the intra-cellular pair-rule genes, but further consolidation of this signal becomes dependent on cell-cell interactions between contacting cells (Sandler et al., 2016; Heuvel et al., 1993). This interaction is regulated by two different pathways, the Hedgehog (*hh*) and the Wingless pathways (Vincent et al., 2008; Ingham et al., 1991).

The *Hh* pathway is launched by a cell expressing *en*. This gene encodes for a transcription factor of the *hh* gene. The *hh* gene product is secreted by the cell and binds to the Patched (*ptc*) receptor of the neighbouring cells. This transduction starts a signalling cascade which leads to the conversion of the *Cubitus interruptus* (*ci*) protein to a transcription factor of the *wg* gene.

## 6. EVALUATION OF NEW UPDATING SCHEMES WITH THE SEGMENT POLARITY MODEL

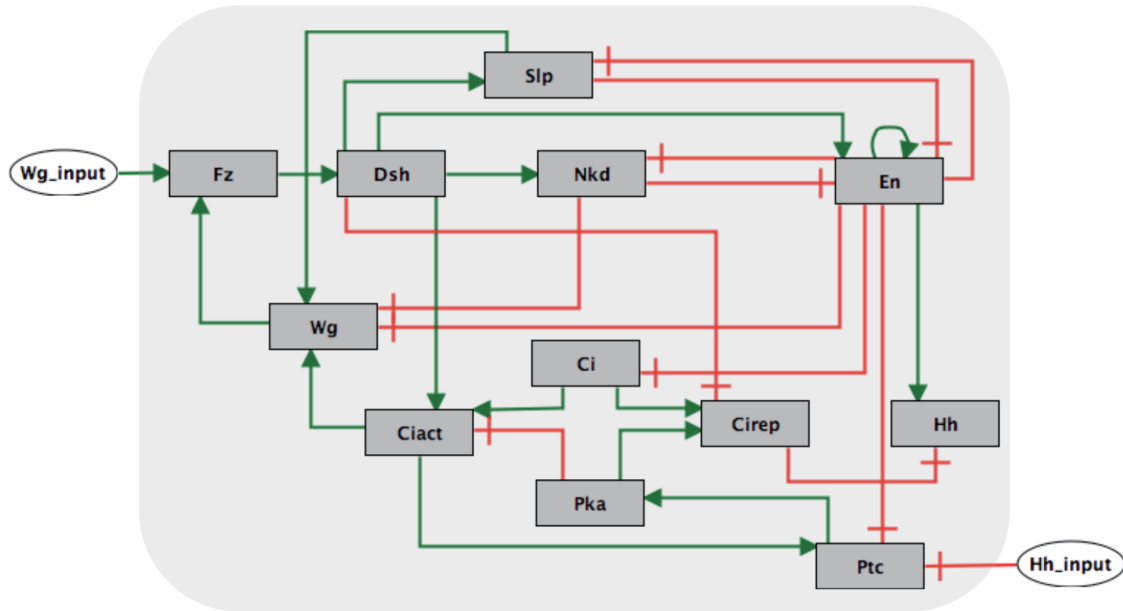


Figure 6.1: Intra-cellular network of the segment-polarity module as published by Sanchez et al., 2008.

The Wg pathway begins with the secretion of wg, which binds to the Frizzled (fz) receptor of the neighbouring cells (en cells). A signalling cascade is started by wg transduction and leads to the activation of en.

**In 2008, Sanchez, Chaouiya and Thieffry** published a model considering the above-mentioned interactions using a logical framework, focusing on the segment polarity module influence on the consolidation of en/wg pattern (Sanchez et al., 2008). The cellular logical model is represented in figure 6.1.

Aside from the above mentioned components, the model considers two additional versions of the ci protein. This protein has a transcription activation domain as well as a transcription inhibitor domain. It exists in the Ci-155 form with its activation domain intact and in the Ci-75 form with this domain clived. The latter (ci[rep]) version inhibits and the former (ci[act]) activates the hh target genes. The ci[act] form accumulates in cells adjacent to those that secrete hh. Another component is the kinase A protein (pka), which inhibits hh signaling by phosphorylating ci[act] (Aza-Blanc et al., 1997; Wang et al., 1999).






Finally, the model takes into account other components of the Wg pathway, such as the Dishevelled (dsh) protein, which is required for the wg signaling cascade to initiate (Yanagawa et al., 1995). This cascade, in addition to en transcription, leads to the transcription of the sloppy paired (slp) and naked (nkd) genes, both of which play important roles in the differentiation of En/Wg expressing cells. Slp is an activator of wg and a repressor of en, whereas nkd is induced by wg expression and represses wg (Grossniklaus et al., 1992).

The authors analysed their model with GINSIM<sup>1</sup> and found that five stable states (see table 6.1) were reached when fixing the wg and hh inputs, *i.e.*, each wg/hh input combination lead to a stable state. One of the five stable states, the Wg cell is characterised by being the only stable state with wg expression, while the En cell is the only stable state with en or hh activity.

<sup>1</sup><http://www.ginsim.org>

## 6.1 Background

Table 6.1: All stable states of the single cell segment polarity model, achieved by fixing the wg and hh inputs. All wg, hh input combinations considered. Note only the Wg cell has wg activity while only the En cell has en and hh activity. The colour column indicates the colour code used by EpiLog, which is used throughout the remainder of this chapter.

Wg	Fz	Dsh	Slp	Nkd	En	Hh	Ci	Ciact	Cirep	Pka	Ptc	Color	
0	0	0	0	1	0	0	1	0	1	2	1		Trivial (T)
0	0	0	0	1	0	0	1	1	0	0	0		CiCiact (C)
2	1	1	1	2	0	0	1	2	0	0	0		Wg (W)
0	1	1	0	0	1	1	0	0	0	0	0		En (E)
0	1	1	1	2	0	0	1	1	0	2	2		Nkd (N)

To simulate a whole para-segment, the authors extended the one-cell model to six cells, defined in a single logical model. The one-cell model was multiplied six times and connected via the wg and hh components, to the fz (wg receptor) and ptc (hh receptor) of the next cell, respectively.

The third cell of this stripe of six cells corresponds to the cell anterior to the para-segmental border while the fourth cell corresponds to the cell posterior to the border. Both these cells have an initial and maintained expression of en and wg, while the cells adjacent, the second and fifth initially express both en and wg but lose expression afterwards. The first and last cell do not express neither en nor wg (see figure 6.2). The initial expression of either en or wg is defined by the pair-rule module.

The reachable stable states were obtained from these realistic initial conditions; while this is now achievable with GINsim alone, the large scale of the model (6\*12 components) did not enable it at the time. As a result, the authors did so by transforming the logical model in a Petri net model (Murata, 1989) that allowed the use of a reachability tool, INA (Integrated Net Analyzer), to compute a state transition graph (see section 2.1.2). The STG was determined using asynchronous priority classes, which divided the components based on the speed of their transitions. The first (fast) class includes protein modification events and the second (slow) class *de novo* gene expression processes. The fast class being Ci[act], Ci[rep], Fz, Dsh and Pka and the slow class being constituted by Ci, Wg, Nkd, En, Slp, Hh and Ptc.

From a total of 36 possible stable states, only two states are reachable when considering the realistic initial conditions, these are the wild-type (since the obtained pattern matches the experimental observations of what is seen *in vivo*) and the trivial states (see figure 6.3).

The wild-type is defined by the following pattern of single cell stable states: "Trivial, Nkd, Wg, En, Ci-Ciact, Trivial." It is distinguished by the presence of wg, slp, nkd, ci, and ciact activity in the third cell (Wg cell), just anterior to the para-segmental boundary. This cell lacks ptc because it is linked to the secreted hh. The fourth wild-type cell (En cell), posterior to the border, exhibits en and hh activity, in addition to fz and dsh, which are activated by wg diffusion by their neighbouring cell. The other cells show no en or hh activity.

## 6. EVALUATION OF NEW UPDATING SCHEMES WITH THE SEGMENT POLARITY MODEL

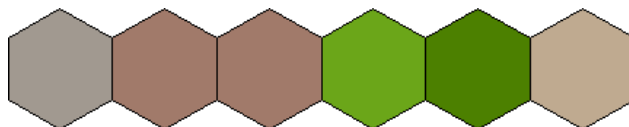
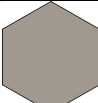
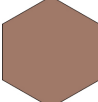
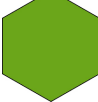
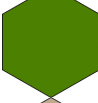
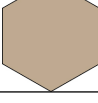


Figure 6.2: Figure taken from EpiLog. Initial wild-type gene expression, dictated by the pair-rule module. The two cells just anterior of the para-segmental border (2<sup>nd</sup> and 3<sup>rd</sup> cells) are characterised by the wg expression while the two cells just posterior of the border (4<sup>th</sup> and 5<sup>th</sup> cells) by en expression.

Table 6.2: Listing of the initial conditions cell gene expression.

Wg	Fz	Dsh	Slp	Nkd	En	Hh	Ci	Ciact	Cirep	Pka	Ptc	Color
0	0	0	0	1	0	0	1	0	1	2	1	
2	1	0	1	1	0	0	1	0	0	2	1	
0	1	1	0	1	1	1	0	0	0	0	0	
0	0	0	0	1	1	1	0	0	0	0	0	
0	0	0	0	1	0	0	1	0	0	2	1	

In the second stable state, trivial, all cells have the same gene expression, defined as a "Trivial" cell. This cell is devoid of en and wg with nkd mildly expressed and ci, ci[rep], pka and ptc highly expressed. This situation corresponds to what happens when the signalling by the pair-rule genes is lost before the inter-cellular interactions consolidate the En and Wg pattern.

## 6.2 Segment polarity multi-cellular model in EpiLog

### 6.2.1 Introduction

To study a stripe of six cells, that constitutes a para-segment, the previous authors had to consider a single logical model composed of the components of all six cells. Although the study was successful, GINsim is not suited to study multi-cellularity, instead, the use of EpiLog may be a better alternative for studying the same model. EpiLog considers multiple cells, each with a copy of a logical model, this modularity makes scaling a single cell model to multi-cellularity easier. As a result, by uploading the single cell segment polarity model to EpiLog, it is now possible to simulate the stripe of six cells with more ease, as well as larger grids of cells. The model was implemented in EpiLog and is available in <http://epilog-tool.org/node/189>.

The implementation in EpiLog requires the definition of some parameters: the initial conditions, integration functions and the updating schemes. The definition of the initial conditions is straight forward as it is the same specified by Sanchez et al., 2008, which assume the initial expression of the pair-rule genes.

## 6.2 Segment polarity multi-cellular model in EpiLog

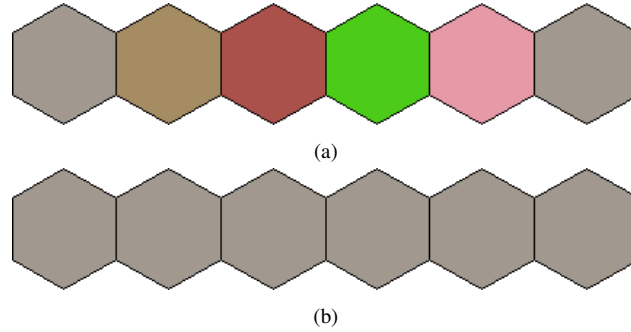


Figure 6.3: The two stable states reachable from realistic initial conditions under the priorities random uniform updating scheme. Simulated with EpiLog and are in accordance with the (Sanchez et al., 2008) results: (a) Wild-type state. (b) Trivial state.

Table 6.3: Phenotypes obtained from simulating a six cell stripe. Wild-type (WT) and Trivial phenotypes are represented in figure 6.3; ticked cell indicates that the corresponding phenotype was recovered.

	Priorities synchronous	Priorities random uniform	Random non uniform
WT		✓	✓
Trivial	✓	✓	✓
Others	0	0	2

The integration functions for the two input components, Hh-external and Wg-external are,  $\{Hh, \min=1\}$  and  $\{Wg;2, \min=1\}$  respectively, see section 3.2.1.1 for integration functions. Hh-external is activated when the level of Hh of the immediate neighbouring cells is at least one and the latter, Wg-external is activated when the level of Wg of the immediate neighbouring cells is at least two.

### 6.2.2 Single para-segment: stripe of six cells

Using the implemented segment polarity model, a stripe of six cells was defined in EpiLog. This enabled a comparison of the author's results with those obtained by EpiLog and even more significant, compare the outcome of the different updating schemes and access the impact of the new ones.

The three updating schemes considered are all based on the two priority classes defined in the original publication (Sanchez et al., 2008), where the two classes are: 1<sup>st</sup> Ci[act], Ci[rep], Fz, Dsh and Pka and 2<sup>nd</sup> Ci, Wg, Nkd, En, Slp, Hh and Ptc. The updating schemes are:

1. **Priorities synchronous:** Both classes are synchronously updated;
2. **Priorities random uniform:** Both classes are updated by the random uniform updating scheme;
3. **Random non uniform:** All components are in a single class with random non uniform updating. The components that were in the previous first class have an updating rate of 20 while the previous second class components have an updating rate of 1.

#### 6.2.2.1 Wild-type: no mutations

Table 6.3 shows that in the absence of perturbations, a simulation under a priority synchronous updating scheme can only lead to the trivial stable state. While with priority random uniform updating both the trivial and wild-type stable states are reachable (see figure 6.3).

## **6. EVALUATION OF NEW UPDATING SCHEMES WITH THE SEGMENT POLARITY MODEL**

The Random non uniform updating scheme not only reaches the wild-type and trivial cases, but also two additional stable states. This is due to the fact that, unlike priorities (whether synchronous or random uniform), it does not restrict the state transition graph trajectory, but rather makes some stable states less or more likely to be reached.

The success of the two new updating schemes in retrieving the original paper phenotypes, in contrast to the synchronous class updating scheme, demonstrates that the addition of the two updating schemes allowed the exploration of new trajectories that may be more biologically relevant, hence proving its worth.

### **6.2.2.2 Mutants**

In the original study, in addition to simulating the wild-type, the authors further tested the model's capacity of replicating the experimental observations, by examining the model's output with genetic perturbations, either loss of (partial or total) function or gain of function (ectopic).

As the priorities synchronous updating scheme was not able to retrieve the expected stable pattern for the wild-type, when simulating the mutants, this updating scheme was discarded. Only the priorities random uniform and random non uniform updating schemes were evaluated.

The priority random uniform updating scheme is able to retrieve all stable states deemed relevant by the authors for the different perturbations. The random non uniform fails in capturing one of the patterns obtained for the *Wg* ectopic mutant. Although this pattern was never reached with the random non uniform updating scheme, it is shown to be a reachable state with the priorities random uniform updating scheme. This occurs due to sampling, the number of simulations runs, with the defined rates, were not enough to obtain the pattern. Table 6.4 summarises these findings.

### **6.2.3 Multiple para-segments: grid of 12 by 12 cells**

#### **6.2.3.1 Wild-type**

As EpiLog can simulate a high number of cells, the six-stripe cell grid was expanded to a grid of 12 by 12 cells. This scenario corresponds to the modelling of a portion of the embryo trunk with several para-segments, in theory two per row (24 total).

The initial conditions and integration functions used in this grid simulation are the same as those used in the six cell stripe simulation. In the latter case, both the priorities random uniform updating and the random non uniform updating schemes were capable of reproducing the author's paper results, with or without mutations. With that, the two updating schemes were used to simulate the grid.

The patterns obtained with the priorities random uniform and the random non uniform updating schemes can be seen in the figure 6.4a and 6.4b respectively. Both these updating schemes fail in capturing the wild-type pattern, that should correspond to a repetition of the six cell stripe wild-type pattern seen before. This occurs because of the greater grid size; whereas the number of possible configurations for six cell stripes was  $5^6$ , the number for a 12 by 12 grid is  $5^{12 \times 12}$ . Although the initial conditions, integration functions, and updating schemes greatly restrict this number, the large number of possible trajectories makes it unlikely that the correct phenotype will be reached.

For the above reason, an alternative updating scheme was proposed:



## 6.2 Segment polarity multi-cellular model in EpiLog

- **Priorities random non uniform:** The components are divided in the two classes just as before (see section 6.2.2). Each component has a given rate, which is explicit in the table 6.5. Some components in the first class are split, the rate of the component increasing in value differs from the rate of the component decreasing in value.

The two priority classes were coupled with the random non uniform updating scheme, with the reasoning that as the number of possible trajectories increases, more restrictions are required for the simulation to reach the desired stable state. The rates were determined by succeeding iterations of simulations of observation of which STG transitions were derailing the target stable state, and favouring the transitions that helped achieved it. As a result, some components had to be split to associate different rates to the increase or decrease of its value.

The final stable state using this updating scheme is shown in figure 6.4c, and equates to the repetition of the pattern seen in figure 6.3a, which corresponds to the wild-type pattern. This pattern is not achieved 100 percent of the times running a simulation, similar patterns where one or two para-segments have cells that break the six cell stripe wild-type pattern occur punctually (see figure 6.5).

All simulations so far used a rectangular topology, *i.e.*, the cells in the simulation grid edge are disconnected from the other grid edges. However, the consideration of a vertical wrap grid, where the cells of the vertical edges of the grid cells are connected (a cylinder), is more resembling of an embryo trunk. Under this condition, the simulation result remains the same and the wild-type pattern is reached.

Both the priority random uniform and random non uniform updating schemes were sufficient to recover the wild-type pattern in the six stripe simulations. In contrast, in the grid simulations, due to the large number of possible stable states, the combination of the priorities with the random non uniform scheme proved to be necessary to reach the the right phenotype. While the priorities limited the STG, the classes updater, random non uniform, made the obtained phenotype more probable. This, in addition to emphasising the usefulness of the priorities, highlights the added value of the random non uniform updating scheme, particularly in achieving patterns that although not probable to reach, due to larger search spaces, are biologically more relevant.

### 6.2.3.2 Mutants

The same mutations tested in the six cell stripe case were also considered in the grid of cells scenario. The use of the priorities random non uniform allowed the retrieval of expected all phenotypes. However, under no circumstances, the simulation reaches more than one stable state due to the restrictiveness of the updating scheme. Therefore, the mutants "Ptc knock-out", ""Wg ectopic" and "Nkd ectopic" that according to the 6.3 had more than one stable state, had the following unique stable states: "E,W,W,E,C,C", "E,W,W,E,E,E" and "T,N,W,E,C,T" respectively.

As it occurred in the wild-type, the pattern observed corresponds to two para-segments where the pattern observed in the six cell stripe phenotype is repeated vertically. Here, a vertical grid wrap was also considered which again conserved the expected phenotypes.

## 6. EVALUATION OF NEW UPDATING SCHEMES WITH THE SEGMENT POLARITY MODEL

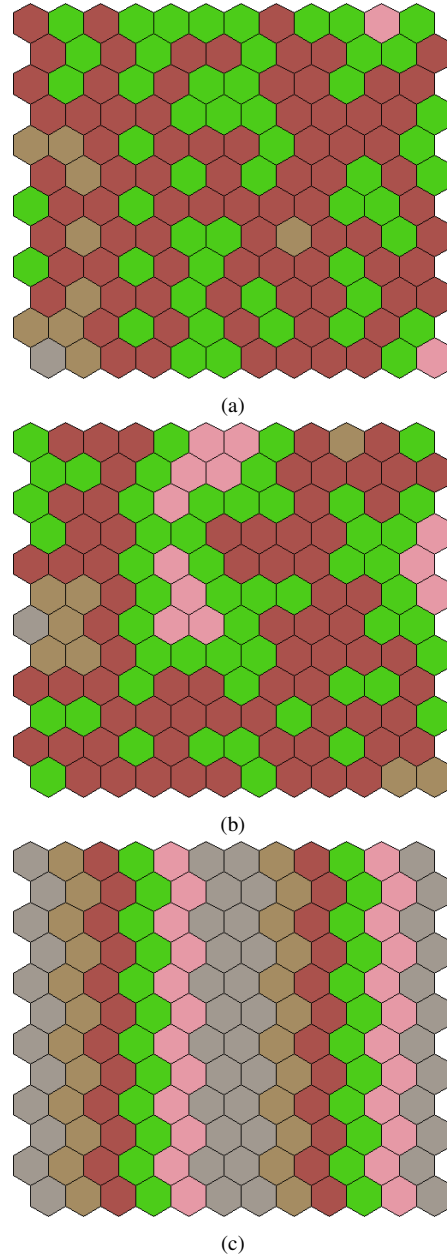


Figure 6.4: Some instances of stable states obtained in the absence of perturbations, for multiple updating schemes. (a) Priorities random uniform (b) Random non uniform with the same rates as for the six cell stripe (c) Priorities random non uniform, the rates are specified in 6.5.

## 6.2 Segment polarity multi-cellular model in EpiLog

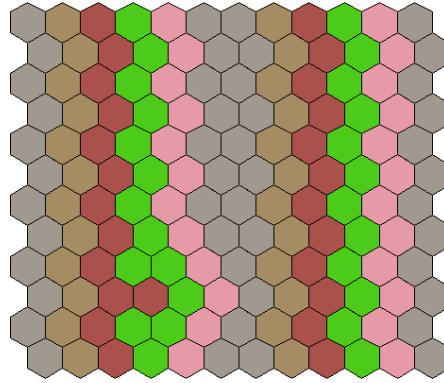


Figure 6.5: One of the stable state obtained with priorities random non uniform updating scheme. Representative of states similar to wild-type but with some irregularities.

Table 6.4: Stable states for the different perturbations, considering both the priority random uniform and the random non uniform updating schemes. All listed stable states were obtained by Sanchez et al., 2008. The last column points to the uniqueness or reachability of the obtained stable states in the original publication.

Perturbations	Stable States	Updating Schemes		Comments
		Priorities random uniform	Random non uniform	
Wg KO En KO Hh KO Ci KO	All Trivial	✓	✓	Unique stable state
Ptc KO	W,E,E,E,C,C	✓	✓	All 2 reachable states
	E,W,W,E,C,C	✓	✓	
	Others	0	2	
Ptc, Wg KO	All Ci-Ciact	✓	✓	Unique stable state
Ptc KO, Wg [0,1]	C,W,W,C,C,C	✓	✓	3 other reachable states according to original paper
En Ect	All En	✓	✓	Unique stable state
Wg Ect	N,N,W,E,E,W	✓	✓	All 4 reachable states according to original paper
	N,N,W,E,E,E	✓		
	E,W,W,E,E,E	✓	✓	
	E,W,W,E,E,W	✓	✓	
	Others	0	2	
Nkd Ect	All Trivial	✓	✓	Closest reachable state from initial conditions
	T,N,W,E,C,T	✓	✓	Stable state not expected according to the original paper
	Others	0	2	
Wg Ect, Slp KO	All En	✓	✓	Unique stable state

## 6. EVALUATION OF NEW UPDATING SCHEMES WITH THE SEGMENT POLARITY MODEL

Table 6.5: Priority classes were used in the simulation of the 12 by 12 grid of cells. Each class is updated using the random non uniform updating scheme, and each component has an associated rate. It is worth noting that multiple components of priority class one are split, which means that the rate of the component increasing in value differs from the rate of the component decreasing in value.

Priority class 1		Priority class 2	
Component	Rate	Component	Rate
Dsh[+]	50	Ci	30
Pka[-]	50	Ptc	30
Dsh[-]	10	En	10
Fz[-]	10	Wg	1
Ciact[-]	10	Slp	1
Ciact[+]	1	Nkd	1
Cirep	1	Hh	1
Fz[+]	1		
Pka[+]	1		

## Chapter 7

# Conclusions and further work

This thesis sought to improve the software tool EpiLog (and bioLQM) in order to provide a better modelling experience to the end user. The main focus was the introduction of asynchronicity in the cellular model updating mode, including within priority classes, in an effort to generate more biologically appropriate simulation of EpiLog models. The relevance of this extension was verified in the EpiLog environment using the segment polarity module as a case study.

The introduction of asynchronicity in the priority classes amounted to singling out random asynchronous trajectories. It proved to be crucial to the retrieval of the expected phenotypes of the previously published segment polarity model. Indeed neither for the six-cell stripe nor for the 12 by 12 grid of cells, the simulations using the synchronous or the priorities synchronous updating schemes was able to retrieve the predicted phenotypes. The simulations of a whole para-segment, achieved the expected phenotypes when using the random non uniform or the priorities random uniform updating schemes. Whereas the priorities limit the number of trajectories in the STG (in comparison to the asynchronous STG), the random non uniform does not limit the number of trajectories but rather increases the likelihood of certain trajectories. With the expansion to a 12 by 12 grid of cells, the number of possible stable states increases significantly, and thus, the two updating schemes, the priorities random uniform and the random non uniform, were not able to achieve the expected phenotypes. Consequently, the priorities random non uniform updating scheme was used and was able to retrieve the expected phenotype as it restricted the number of trajectories to be explored while favouring relevant ones. The new updating schemes thus proved to be especially useful in reaching the expected stable state pattern in a large search space, and demonstrated the importance of breaking the synchronicity underlying cellular automata dynamics and adding biologically relevant information (rates associated to the component updates) to achieve biologically meaningful patterns.

EpiLog has been shown to be capable of reproducing the previous authors' segment polarity phenotypes, not only for a para-segment but also for a larger epithelium (12 by 12 grid of cells). The previous authors obtained the phenotypes by representing a para-segment in a single logical model, whereas EpiLog considers several single cell logical models that are connected by integration inputs. This approach of multi-cellular modelling has the advantage of being easily scalable to bigger epithelia, making it more suitable for studying multi-cellular dynamics.

The present implementation of the priority classes provides the user with many combinations of the priority classes with the synchronous, random uniform and random non uniform updating schemes. As a

## 7. CONCLUSIONS AND FURTHER WORK

consequence, it expands the number of trajectories a system can explore while giving the user more control over the trajectories it explored.

This extension of the priority classes grants a more flexible cellular updating approach not only in EpiLog but also in all software tools dependent on bioLQM. The current priority updater supports the single successor updating schemes used by EpiLog and listed above, in addition to the asynchronous and complete updating schemes, which yield multiple successors.

Besides the alteration of the priority classes, two new features were added to endow EpiLog with more tools to define and analyse a simulation.

The new “phenotype tracking” tool outputs a time-series of the distribution of user defined phenotypes. A phenotype captures the genetic expression signatures that are characteristic of specific cells, which may be of biological interest for the user. This time-series can be outputted at any step of the simulation and gives the user an alternative view of the simulation dynamics, focused at the level of the phenotype instead of the whole epithelium.

Finally, the new "Edit by text" mode is directed at more experienced users, allowing them to partially bypass the graphical interface, permitting a speedier modelling experience.

The three described extensions were the result of an update of the existing graphical user interface or the development of a new one, using the Java Swing toolkit.

Following up on the described extensions, there are still ways to improve EpiLog. A natural extension of the phenotype definition would be to enable the visualisation of the defined phenotypes in the simulation grid of the simulation tab. This could be done by defining a colour in the phenotype definition tab, and being able to toggle between colour by gene expression and colour by phenotype in the simulation tab. Additionally, it would also be convenient to be able to click on a cell and see the name of the matched phenotype if any.

When dealing with non-deterministic updating techniques, one limitation of EpiLog, as demonstrated in the segment polarity model, is that the number of stable states is uncertain. An improvement would be to give the user the ability to run a fixed number of simulations in one go, either using the GUI or through the command line. The output would consist of a count of how many times an attractor is reached, as well as the image of the related epithelium. This feature, while not giving an exact measurement of how many attractors there are, would give an idea of how reachable (probable), a state or cycle is. Additionally, it would make the simulation process more automated.

Overall, multi-cellular modelling is an active research topic in which various methodologies have been proposed, each with its own set of benefits and drawbacks. EpiLog appears to be the only framework that explicitly considers the communication between cells while supporting complex cellular models. With this thesis, the EpiLog tool continues to be improved in order to provide a more biologically complete and user-friendly modelling experience.

# Bibliography

- Abou-Jaoudé, Wassim et al. (May 2016). “Logical Modeling and Dynamical Analysis of Cellular Networks”. In: *Frontiers in Genetics* 7. DOI: 10.3389/fgene.2016.00094. URL: <https://doi.org/10.3389/fgene.2016.00094>.
- Alon, U. et al. (Jan. 1999). “Robustness in bacterial chemotaxis”. en. In: *Nature* 397.6715, pp. 168–171. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/16483. URL: <http://www.nature.com/articles/16483> (visited on 10/28/2021).
- Aza-Blanc, Pedro et al. (June 1997). “Proteolysis That Is Inhibited by Hedgehog Targets Cubitus interruptus Protein to the Nucleus and Converts It to a Repressor”. en. In: *Cell* 89.7, pp. 1043–1053. ISSN: 00928674. DOI: 10.1016/S0092-8674(00)80292-5. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0092867400802925> (visited on 08/30/2021).
- Bouré, Olivier, Nazim Fatès, and Vincent Chevrier (Dec. 2012). “Probing robustness of cellular automata through variations of asynchronous updating”. en. In: *Natural Computing* 11.4, pp. 553–564. ISSN: 1567-7818, 1572-9796. DOI: 10.1007/s11047-012-9340-y. URL: <http://link.springer.com/10.1007/s11047-012-9340-y> (visited on 05/31/2021).
- Chaouiya, Claudine, Duncan Bérenguier, et al. (Dec. 2013a). “SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools”. In: *BMC Systems Biology* 7.1, p. 135. ISSN: 1752-0509. DOI: 10.1186/1752-0509-7-135. URL: <https://doi.org/10.1186/1752-0509-7-135> (visited on 05/29/2021).
- Chaouiya, Claudine, Duncan Bérenguier, et al. (2013b). “SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools”. In: *BMC Systems Biology* 7.1, p. 135. DOI: 10.1186/1752-0509-7-135. URL: <https://doi.org/10.1186/1752-0509-7-135>.
- D’Argenio, Valeria (2018). “The high-throughput analyses era: are we ready for the data struggle?” In: *High-throughput* 7.1, p. 8.
- Dassow, George von et al. (July 2000). “The segment polarity network is a robust developmental module”. en. In: *Nature* 406.6792, pp. 188–192. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/35018085. URL: <http://www.nature.com/articles/35018085> (visited on 10/28/2021).
- Deutsch, Andreas and Sabine Dormann (2005). *Cellular Automaton Modeling of Biological Pattern Formation*. Birkhäuser Boston. DOI: 10.1007/b138451. URL: <https://doi.org/10.1007/b138451>.
- Di Cara, Alessandro et al. (2007). “Dynamic simulation of regulatory networks using SQUAD”. en. In: *BMC Bioinformatics* 8.1, p. 462. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-462. URL: <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-462> (visited on 05/30/2021).

## BIBLIOGRAPHY

- Drasdo, D., A. Buttenschön, and P. Van Liedekerke (2018). “Agent-Based Lattice Models of Multicellular Systems”. In: *Numerical Methods and Advanced Simulation in Biomechanics and Biological Processes*. Elsevier, pp. 223–238. DOI: 10.1016/b978-0-12-811718-7.00012-5. URL: <https://doi.org/10.1016%2Fb978-0-12-811718-7.00012-5>.
- Drasdo, Dirk and Stefan Höhme (July 2005). “A single-cell-based model of tumor growth in vitro: monolayers and spheroids”. In: *Physical Biology* 2.3, pp. 133–147. DOI: 10.1088/1478-3975/2/3/001. URL: <https://doi.org/10.1088%2F1478-3975%2F2%2F3%2F001>.
- Fatès, Nazim (Aug. 2014). “A guided tour of asynchronous cellular automata”. In: *arXiv:1406.0792 [nlin]*. arXiv: 1406.0792. URL: <http://arxiv.org/abs/1406.0792> (visited on 05/31/2021).
- Faure, A. et al. (July 2006). “Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle”. In: *Bioinformatics* 22.14, e124–e131. DOI: 10.1093/bioinformatics/btl210. URL: <https://doi.org/10.1093%2Fbioinformatics%2Fbtl210>.
- Fauré, Adrien et al. (Mar. 2014). “A Discrete Model of Drosophila Eggshell Patterning Reveals Cell-Autonomous and Juxtacrine Effects”. en. In: *PLoS Computational Biology* 10.3. Ed. by Stanislav Shvartsman, e1003527. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003527. URL: <https://dx.plos.org/10.1371/journal.pcbi.1003527> (visited on 06/02/2021).
- Gibson, Matthew C. et al. (Aug. 2006). “The emergence of geometric order in proliferating metazoan epithelia”. In: *Nature* 442.7106, pp. 1038–1041. DOI: 10.1038/nature05014. URL: <https://doi.org/10.1038%2Fnature05014>.
- Glass, Leon and Stuart A. Kauffman (Apr. 1973). “The logical analysis of continuous, non-linear biochemical control networks”. en. In: *Journal of Theoretical Biology* 39.1, pp. 103–129. ISSN: 00225193. DOI: 10.1016/0022-5193(73)90208-7. URL: <https://linkinghub.elsevier.com/retrieve/pii/0022519373902087> (visited on 10/31/2021).
- Graner, François and James A. Glazier (Sept. 1992). “Simulation of biological cell sorting using a two-dimensional extended Potts model”. In: *Physical Review Letters* 69.13, pp. 2013–2016. DOI: 10.1103/physrevlett.69.2013. URL: <https://doi.org/10.1103%2Fphysrevlett.69.2013>.
- Grossniklaus, U, R K Pearson, and W J Gehring (June 1992). “The Drosophila sloppy paired locus encodes two proteins involved in segmentation that show homology to mammalian transcription factors.” en. In: *Genes & Development* 6.6, pp. 1030–1051. ISSN: 0890-9369. DOI: 10.1101/gad.6.6.1030. URL: <http://www.genesdev.org/cgi/doi/10.1101/gad.6.6.1030> (visited on 10/23/2021).
- Helikar, Tomáš et al. (2012). “The Cell Collective: Toward an open and collaborative approach to systems biology”. en. In: *BMC Systems Biology* 6.1, p. 96. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-96. URL: <http://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-6-96> (visited on 05/30/2021).
- Heuvel, M. van den et al. (1993). “Cell patterning in the Drosophila segment: engrailed and wingless antigen distributions in segment polarity mutant embryos”. eng. In: *Development (Cambridge, England). Supplement*, pp. 105–114.
- Hucka, M. et al. (Mar. 2003). “The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models”. en. In: *Bioinformatics* 19.4, pp. 524–531. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btg015. URL: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btg015> (visited on 05/29/2021).



## BIBLIOGRAPHY

- Ingham, P. W., A. M. Taylor, and Y. Nakano (Sept. 1991). “Role of the *Drosophila* patched gene in positional signalling”. en. In: *Nature* 353.6340, pp. 184–187. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/353184a0. URL: <http://www.nature.com/articles/353184a0> (visited on 09/02/2021).
- Jaeger, Johannes (Jan. 2011). “The gap gene network”. en. In: *Cellular and Molecular Life Sciences* 68.2, pp. 243–274. ISSN: 1420-682X, 1420-9071. DOI: 10.1007/s00018-010-0536-y. URL: <http://link.springer.com/10.1007/s00018-010-0536-y> (visited on 09/02/2021).
- Johnston, Daniel St and Christiane Nüsslein-Volhard (Jan. 1992). “The origin of pattern and polarity in the *Drosophila* embryo”. en. In: *Cell* 68.2, pp. 201–219. ISSN: 00928674. DOI: 10.1016/0092-8674(92)90466-P. URL: <https://linkinghub.elsevier.com/retrieve/pii/009286749290466P> (visited on 08/30/2021).
- Jong, Hidde de (Jan. 2002). “Modeling and Simulation of Genetic Regulatory Systems: A Literature Review”. In: *Journal of Computational Biology* 9.1, pp. 67–103. DOI: 10.1089/10665270252833208. URL: <https://doi.org/10.1089%2F10665270252833208>.
- Kauffman, S.A. (1969). “Metabolic stability and epigenesis in randomly constructed genetic nets”. In: *Journal of Theoretical Biology* 22.3, pp. 437–467. ISSN: 0022-5193. DOI: [https://doi.org/10.1016/0022-5193\(69\)90015-0](https://doi.org/10.1016/0022-5193(69)90015-0). URL: <https://www.sciencedirect.com/science/article/pii/0022519369900150>.
- Le Novère, Nicolas (Mar. 2015). “Quantitative and logic modelling of molecular and gene networks”. en. In: *Nature Reviews Genetics* 16.3, pp. 146–158. ISSN: 1471-0056, 1471-0064. DOI: 10.1038/nrg3885. URL: <http://www.nature.com/articles/nrg3885> (visited on 10/31/2021).
- Letort, Gaele et al. (Apr. 2019). “PhysiBoSS: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling”. eng. In: *Bioinformatics (Oxford, England)* 35.7, pp. 1188–1196. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bty766.
- Martinez-Arias, Alfonso and Peter A. Lawrence (Feb. 1985). “Parasegments and compartments in the *Drosophila* embryo”. en. In: *Nature* 313.6004, pp. 639–642. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/313639a0. URL: <http://www.nature.com/articles/313639a0> (visited on 09/02/2021).
- Meineke, F. A., C. S. Potten, and M. Loeffler (Aug. 2001). “Cell migration and organization in the intestinal crypt using a lattice-free model”. In: *Cell Proliferation* 34.4, pp. 253–266. DOI: 10.1046/j.0960-7722.2001.00216.x. URL: <https://doi.org/10.1046%2Fj.0960-7722.2001.00216.x>.
- Mirams, Gary R. et al. (Mar. 2013). “Chaste: An Open Source C++ Library for Computational Physiology and Biology”. en. In: *PLoS Computational Biology* 9.3. Ed. by Andreas Prlic, e1002970. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002970. URL: <https://dx.plos.org/10.1371/journal.pcbi.1002970> (visited on 05/27/2021).
- Murata, T. (Apr. 1989). “Petri nets: Properties, analysis and applications”. In: *Proceedings of the IEEE* 77.4, pp. 541–580. ISSN: 00189219. DOI: 10.1109/5.24143. URL: <http://ieeexplore.ieee.org/document/24143/> (visited on 09/03/2021).
- Müssel, Christoph, Martin Hopfensitz, and Hans A. Kestler (May 2010). “BoolNet—an R package for generation, reconstruction and analysis of Boolean networks”. en. In: *Bioinformatics* 26.10, pp. 1378–1380. ISSN: 1460-2059, 1367-4803. DOI: 10.1093/bioinformatics/btq124. URL: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btq124> (visited on 05/30/2021).

## BIBLIOGRAPHY

- Naldi, Aurélien (Nov. 2018). “BioLQM: A Java Toolkit for the Manipulation and Conversion of Logical Qualitative Models of Biological Networks”. In: *Frontiers in Physiology* 9. DOI: 10.3389/fphys.2018.01605. URL: <https://doi.org/10.3389/fphys.2018.01605>.
- Naldi, Aurélien, Céline Hernandez, Wassim Abou-Jaoudé, et al. (June 2018). “Logical Modeling and Analysis of Cellular Regulatory Networks With GINsim 3.0”. In: *Frontiers in Physiology* 9. DOI: 10.3389/fphys.2018.00646. URL: <https://doi.org/10.3389/fphys.2018.00646>.
- Naldi, Aurélien, Céline Hernandez, Nicolas Levy, et al. (June 2018). “The CoLoMoTo Interactive Notebook: Accessible and Reproducible Computational Analyses for Qualitative Biological Networks”. In: *Frontiers in Physiology* 9. DOI: 10.3389/fphys.2018.00680. URL: <https://doi.org/10.3389/fphys.2018.00680>.
- Naldi, Aurélien, Denis Thieffry, and Claudine Chaouiya (2007). “Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks”. In: *Computational Methods in Systems Biology*. Springer Berlin Heidelberg, pp. 233–247. DOI: 10.1007/978-3-540-75140-3\_16. URL: [https://doi.org/10.1007/978-3-540-75140-3\\_16](https://doi.org/10.1007/978-3-540-75140-3_16).
- Osborne, James M. et al. (Feb. 2017). “Comparing individual-based approaches to modelling the self-organization of multicellular tissues”. In: *PLOS Computational Biology* 13.2. Ed. by Qing Nie, e1005387. DOI: 10.1371/journal.pcbi.1005387. URL: <https://doi.org/10.1371/journal.pcbi.1005387>.
- Palsson, Bernhard (Nov. 2000). “The challenges of in silico biology”. en. In: *Nature Biotechnology* 18.11, pp. 1147–1150. ISSN: 1087-0156, 1546-1696. DOI: 10.1038/81125. URL: [http://www.nature.com/articles/nbt1100\\_1147](http://www.nature.com/articles/nbt1100_1147) (visited on 11/01/2021).
- Rubinacci, Simone et al. (2015). “CoGNaC: A Chaste Plugin for the Multiscale Simulation of Gene Regulatory Networks Driving the Spatial Dynamics of Tissues and Cancer”. en. In: *Cancer Informatics* 14.Suppl 4. Publisher: SAGE Publications, p. 53. DOI: 10.4137/CIN.S19965. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4559197/> (visited on 05/27/2021).
- Salazar-Ciudad, Isaac, Jukka Jernvall, and Stuart A. Newman (May 2003). “Mechanisms of pattern formation in development and evolution”. en. In: *Development* 130.10, pp. 2027–2037. ISSN: 1477-9129, 0950-1991. DOI: 10.1242/dev.00425. URL: <https://journals.biologists.com/dev/article/130/10/2027/52032/Mechanisms-of-pattern-formation-in-development-and> (visited on 11/02/2021).
- Sanchez, Lucas, Claudine Chaouiya, and Denis Thieffry (2008). “Segmenting the fly embryo: logical analysis of the role of the Segment Polarity cross-regulatory module”. In: *The International Journal of Developmental Biology* 52.8, pp. 1059–1075. DOI: 10.1387/ijdb.0724391s. URL: <https://doi.org/10.1387/ijdb.0724391s>.
- Sandler, Jeremy E. and Angelike Stathopoulos (July 2016). “Stepwise Progression of Embryonic Patterning”. en. In: *Trends in Genetics* 32.7, pp. 432–443. ISSN: 01689525. DOI: 10.1016/j.tig.2016.04.004. URL: <https://linkinghub.elsevier.com/retrieve/pii/S016895251630018X> (visited on 08/29/2021).
- Schiff, Joel L. (2007). *Cellular Automata: A Discrete View of the World*.
- Smolen, P (Feb. 2000). “Modeling Transcriptional Control in Gene Networks—Methods, Recent Results, and Future Directions”. In: *Bulletin of Mathematical Biology* 62.2, pp. 247–292. DOI: 10.1006/bulm.1999.0155. URL: <https://doi.org/10.1006/bulm.1999.0155>.

## BIBLIOGRAPHY

- Stoll, Gautier, Barthélémy Caron, et al. (July 2017). “MaBoSS 2.0: an environment for stochastic Boolean modeling”. en. In: *Bioinformatics* 33.14. Ed. by Jonathan Wren, pp. 2226–2228. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btx123. URL: <https://academic.oup.com/bioinformatics/article/33/14/2226/3059141> (visited on 05/30/2021).
- Stoll, Gautier, Eric Viara, et al. (2012). “Continuous time boolean modeling for biological signaling: application of Gillespie algorithm”. en. In: *BMC Systems Biology* 6.1, p. 116. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-116. URL: <http://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-6-116> (visited on 05/30/2021).
- Thomas, René (1973). “Boolean formalization of genetic control circuits”. In: *Journal of Theoretical Biology* 42.3, pp. 563–585. ISSN: 0022-5193. DOI: [https://doi.org/10.1016/0022-5193\(73\)90247-6](https://doi.org/10.1016/0022-5193(73)90247-6). URL: <https://www.sciencedirect.com/science/article/pii/0022519373902476>.
- Thomas, René and Richard d’Ari (1990). *Biological feedback*. CRC press.
- Varela, Pedro L. et al. (Mar. 2019). “EpiLog: A software for the logical modelling of epithelial dynamics”. en. In: *FI000Research* 7, p. 1145. ISSN: 2046-1402. DOI: 10.12688/f1000research.15613.2. URL: <https://f1000research.com/articles/7-1145/v2> (visited on 06/13/2021).
- Vincent, Stephane, Norbert Perrimon, and Jeffrey D. Axelrod (Aug. 2008). “Hedgehog and Wingless stabilize but do not induce cell fate during *Drosophila* dorsal embryonic epidermal patterning”. en. In: *Development* 135.16, pp. 2767–2775. ISSN: 1477-9129, 0950-1991. DOI: 10.1242/dev.017814. URL: <https://journals.biologists.com/dev/article/135/16/2767/43637/Hedgehog-and-Wingless-stabilize-but-do-not-induce> (visited on 09/02/2021).
- Wang, G., B. Wang, and J. Jiang (Nov. 1999). “Protein kinase A antagonizes Hedgehog signaling by regulating both the activator and repressor forms of Cubitus interruptus”. en. In: *Genes & Development* 13.21, pp. 2828–2837. ISSN: 0890-9369. DOI: 10.1101/gad.13.21.2828. URL: <http://www.genesdev.org/cgi/doi/10.1101/gad.13.21.2828> (visited on 09/02/2021).
- Wolfram, Stephen (July 1983). “Statistical mechanics of cellular automata”. In: *Reviews of Modern Physics* 55.3. Publisher: American Physical Society, pp. 601–644. DOI: 10.1103/RevModPhys.55.601. URL: <https://link.aps.org/doi/10.1103/RevModPhys.55.601> (visited on 05/30/2021).
- (Jan. 1984). “Universality and complexity in cellular automata”. en. In: *Physica D: Nonlinear Phenomena* 10.1, pp. 1–35. ISSN: 0167-2789. DOI: 10.1016/0167-2789(84)90245-8. URL: <https://www.sciencedirect.com/science/article/pii/0167278984902458> (visited on 05/30/2021).
- Yanagawa, S et al. (May 1995). “The dishevelled protein is modified by wingless signaling in *Drosophila*.” en. In: *Genes & Development* 9.9, pp. 1087–1097. ISSN: 0890-9369. DOI: 10.1101/gad.9.9.1087. URL: <http://www.genesdev.org/cgi/doi/10.1101/gad.9.9.1087> (visited on 09/03/2021).



# Appendices



# Appendix A

## Graphical user interface manual

### A.1 Priority Panel - bioLQM

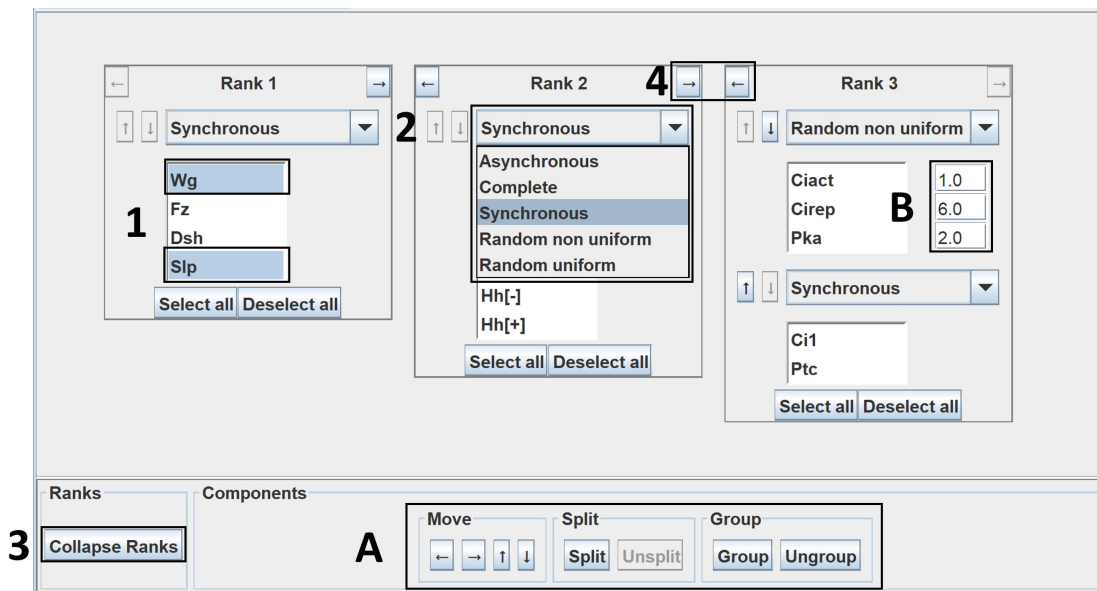


Figure A.1: Priority class panel of bioLQM

**Use of Priority Panel** No limitations are made to the updating scheme, both multiple groups per ranked class and multiple successors updaters are available. The figure A.1 highlights the steps needed to define the priority classes:

1. Select variables and one of the options of the Components panel (A):

- Move section: Move the selected variables to a different group or rank, with the arrow buttons.
- Split section: Split and unsplit the selected variables, in X[+] and X[-].
- Group section: Group the variables, joins the selected variables in a single group. Ungroup the variables, puts each selected variable in its own group.

## A. GRAPHICAL USER INTERFACE MANUAL

2. Select the group updating scheme from the JComboBox. (B): If the "Random non uniform" updating scheme is selected, define each component rates.

Other actions that deal with ranks are also possible:

3. Collapse Ranks, join all groups in a single rank, keeping the groups structure.
4. Change rank of each ranked class with the top arrows in each ranked class.

## A.2 EpiLog

### A.2.1 Cellular model update: Priority Panel

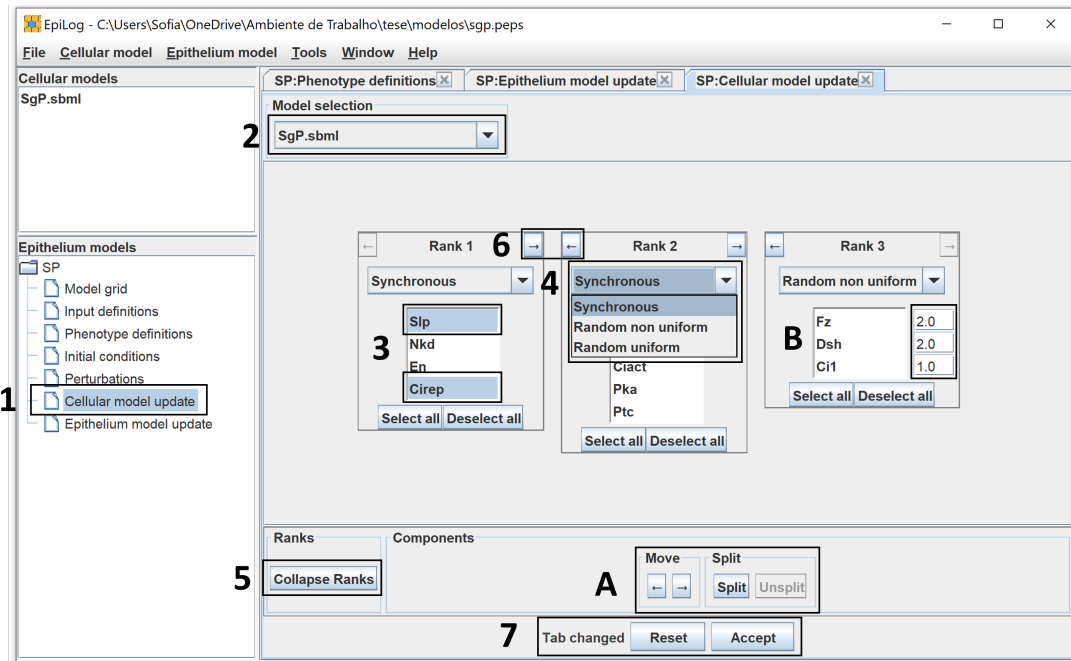


Figure A.2: EpiLog tab "Cellular model update", which uses the Priority class panel of bioLQM, and steps of utilisation.

**Use of Priority Panel within EpiLog.** EpiLog restricts the use of the priority panel to only accept one group per ranked class and single successors updaters. To define the cellular mode update in EpiLog follow the steps highlighted in figure A.2:

1. Right click on "Cellular model update", on the left panel.
2. Select the cellular model you want to define the updating mode to.
3. Select variables and one of the options of the Components panel (A):
  - Move section: Move the selected variables to a different group or rank, with the arrow buttons.
  - Split section: Split and unsplit the selected variables, in X[+] and X[-].



4. Select the group updating scheme from the JComboBox. (B): If the "Random non uniform" updating scheme is selected, define each component rates.

Other actions that deal with ranks are also possible:

5. Collapse Ranks, join all ranks in a single one.
6. Change rank of each ranked class with the top arrows in each ranked class.
7. At last, EpiLog allows the user to reset the tab to its initial state or accept the changes made.

### A.2.2 Phenotype definition

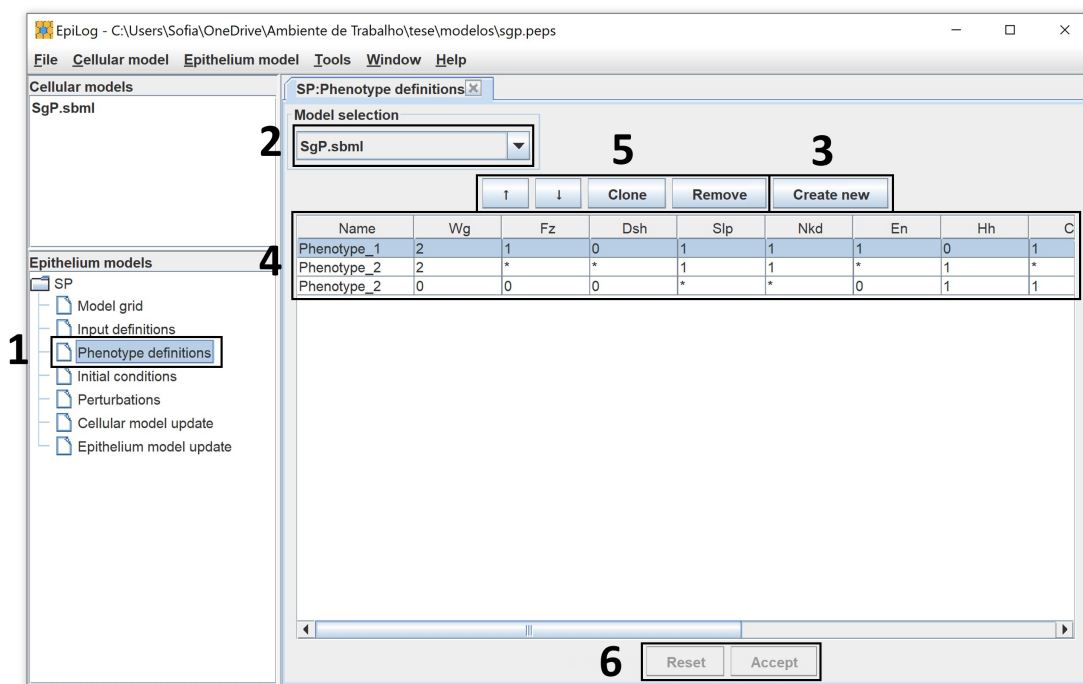


Figure A.3: EpiLog tab "Phenotype definitions".

**Use of Phenotype definitions tab.** The phenotypes are defined by the user with the following steps, highlighted in the figure A.3:

1. Right click on "Phenotype definitions", on the left panel.
2. Select the cellular model it wants to define the phenotypes to.
3. Click "Create new" to add a new state. Note a phenotype is a set of models states that share a name.
4. Use the table to define the state name and each component value. The component value has to be valid, i.e., within the component value's range, if not it will automatically change to the closest valid value. The default component value is "\*", a wildcard that represents any valid value.

## A. GRAPHICAL USER INTERFACE MANUAL

5. Use the buttons to reorder the states, clone or remove them.
6. At last, EpiLog allows the user to reset the tab to its initial state or accept the changes made.

### A.2.3 Edit by text

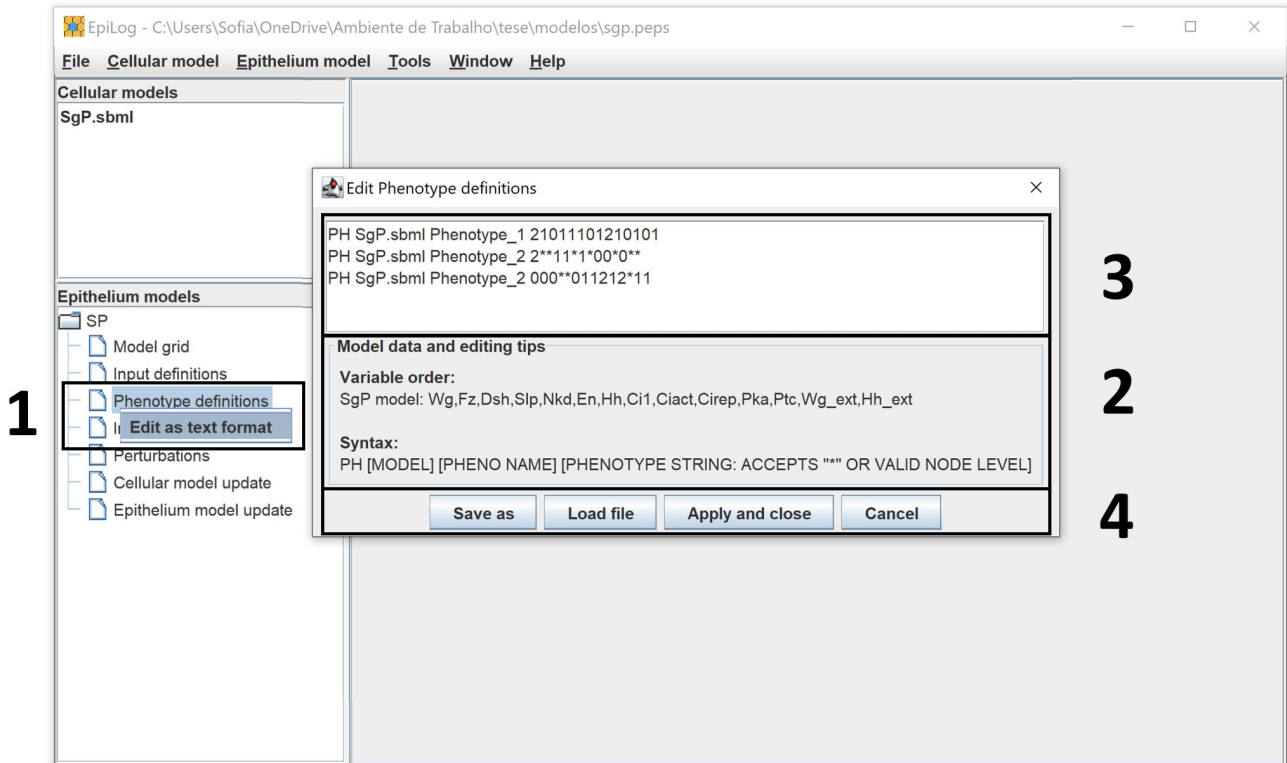


Figure A.4: EpiLog view of "Edit phenotype definitions".

**Use of "Edit by text" mode of defining the Epithelium** This mode is accessed and utilised with the following steps, highlighted in the figure A.4:

1. Left click on the desired tab, in this case "Phenotype definitions", followed by a right click on the pop-up below it.
2. Read the "Model data and editing tips" panel, to understand the syntax of the definitions. If it is not in accordance to it, it will not be accepted.
3. Write the desired Epithelium definitions on the textbox according to the syntax shown in the panel (2), one definition per row.
4. Finally, click "Apply and close" to complete the definition. Alternatively, the written definitions can be saved to a file, or the definitions can be uploaded to the textbox. To close and discard the changes made click "Cancel".

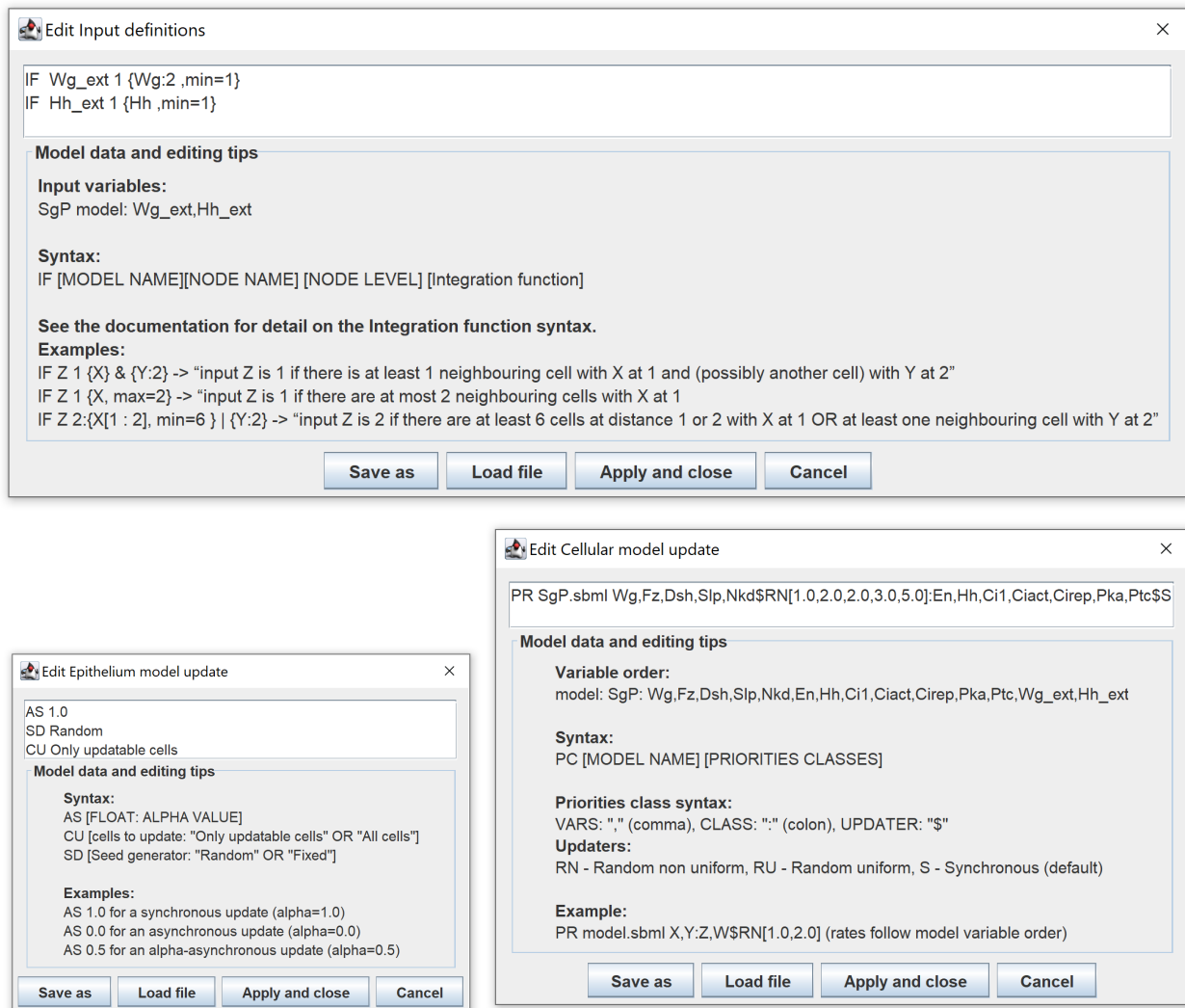


Figure A.5: Three of the four dialogs that support the "Edit by text" mode. On top is "Input definitions", bottom left is "Epithelium model update" and bottom right is "Cellular model update".



## Appendix B

### JUnit tests

JUnit is a framework to write and run unit tests, i.e., evaluate if units of code have the expected output. This framework has been used in bioLQM to test several of its functionalities, attesting that the changes described in this thesis do not affect the previous version.

The tests described bellow serve also to ensure that future modifications do not impact the work done in this thesis, and also test if the modifications are working in the present.

#### B.1 Models

1. Consider a 5 component model, model #1, where all the components are isolated, with a constant regulatory rule:

$$A = 1; B = 1; C = 1; D = 1; E = 1$$

2. Consider a 5 component model, model #2, with the following transition rules:

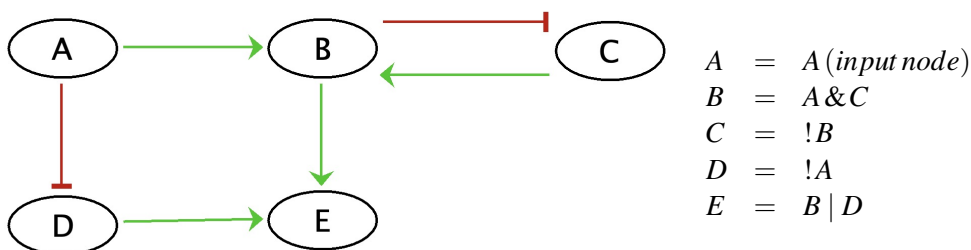


Figure B.1: A) Logical model with 5 components B) Transition rules of each model component

#### B.2 JUnit tests

The modifications to bioLQM are centered on: (1) the PCRankGroupVars class, which defines a priority class and, (2) the Priority Updater class, which computes the priority classes successors. As a result, the

## B. JUNIT TESTS

JUnit tests primarily cover these two classes.

### B.2.1 Random uniform

1. testRandomUpdaterWrapper() at TestUpdaters

- **Model:** #1
- **Goals:**
  - (a) Test if the successor distribution is random.
- **Priority definition (one rank with single group - Random uniform updater)**
- **Input - initial state:** [0,0,0,0,0]
- **Potential Successors = Expected Successors:**
  - (a) A update - [1,0,0,0,0]; Prob = 1/5;
  - (b) B update - [0,1,0,0,0]; Prob = 1/5;
  - (c) C update - [0,0,1,0,0]; Prob = 1/5;
  - (d) D update - [0,0,0,1,0]; Prob = 1/5;
  - (e) E update - [0,0,0,0,1]; Prob = 1/5;
- **Expected output:** After 10.000 repetitions, each potential successor count is between  $[10.000 * Prob * 0.9, 10.000 * Prob * 1.1]$

### B.2.2 Random non uniform - Random with rates

1. testRandomUpdaterWithRates() at TestUpdaters

- **Model:** #2
- **Goals:**
  - (a) Test if the successor distribution is in accordance with the given components rates.
- **Priority definition (one rank with single group - Random non uniform updater)**
- **B:** 0.1; **C:** 0.1; **D:** 0.2; **E:** 0.2;
- **Input - initial state:** [0,0,0,0,0]
- **Successors if all variables were in one asynchronous class:**
  - (a) C update - [0,0,1,0,0];
  - (b) D update - [0,0,0,1,0];
- **Expected Successors:**
  - (a) C update - [0,0,1,0,0]; Prob = 0.1/0.3
  - (b) D update - [0,0,0,1,0]; Prob = 0.2/0.3
- **Expected output:** After 10.000 runs, each potential successor count is between  $[10.000 * Prob * 0.9, 10.000 * Prob * 1.1]$

2. testRandomUpdaterWithRatesAndSplit() at TestUpdaters

- **Model:** #2
- **Goals:**
  - Test if the successor distribution is in accordance with the given components rates.
  - Tests if the split variables accept different rates.
- **Priority definition (one rank with single group - Random non uniform updater)**
- **B:** 0.3; **C[+]:** 0.7; **C[:-]** 0.2; **D:** 0.1; **E:** 0.4;
- **Input - initial states:**
  - (a) [1,0,0,1,0]
  - (b) [1,1,1,0,0]
- **Successors and expected (random non uniform) distribution:**
  - Of initial state 1
    - (a) **C[+] update** - [1,0,1,1,0]; Prob = 0.2/0.7;
    - (b) **D update** - [1,0,0,0,0]; Prob = 0.1/0.7;
    - (c) **E update** - [1,0,0,1,1]; Prob = 0.4/0.7;
  - Of initial state 2
    - (a) **C[-] update** - [1,1,0,0,0]; Prob = 0.7/1.1
    - (b) **E update** - [1,1,1,0,1]; Prob = 0.4/1.1
  - **Expected output:** After 10.000 repetitions, each potential successor count is between  $[10.000 * Prob * 0.9, 10.000 * Prob * 1.1]$

### B.2.3 Priority Updater

1. testPriorityUpdaterRankChoice() at TestUpdaters

- **Goals:**
  - Tests if the priority updater chooses the successor from the higher rank when different ranks have one valid successor each.
- **Model:** #2
- **Priority definition (two ranks single group):**
  - **Rank 1, Group 1:** B, C. Updater: Synchronous
  - **Rank 2, Group 1:** D, E. Updater: Synchronous
- **Initial state:** [1,1,1,0,0]
- **Successors if all variables were in one asynchronous class:**
  - (a) C update - [1,1,0,0,0]
  - (b) E update - [1,1,1,0,1]
- **Expected successors:** Successor (a). Rank 1, Group 1: C update - [1,1,0,0,0]

## B. JUNIT TESTS

### 2. testPriorityUpdaterGroupSucc() at TestUpdaters

- **Goals:**
  - (a) Test if the number of successors in one rank coincides with the number of groups, in that rank, that have successors.
- **Model: #1**
- **Priority definition (one rank with two groups):**
  - **Rank 1, Group 1:** A, B, C. Updater: Synchronous;
  - **Rank 1, Group 2:** D, E. Updater: Synchronous
- **Initial state:** [0,0,0,0,0]
- **Successors if all variables were in one asynchronous class**
  - (a) A update - [1,0,0,0,0]; B update - [0,1,0,0,0], C update - [0,0,1,0,0], D update - [0,0,0,1,0], E update - [0,0,0,0,1]
- **Expected successors:**
  - (a) Rank 1, Group 1: A, B and C update - [1,1,1,0,0]
  - (b) Rank 1, Group 2: D,E update - [0,0,0,1,1]
- **Expected output:** Two successors are expected, (a) and (b). One for each group.

### B.2.4 Simulation: Priority Updater

#### 1. testPriorityUpdaterSimulation() at TestSimulations

- **Model: #2**
- **Goals:**
  - (a) Tests if the priority updater captures the cyclic attractor.
- **Priority definition (two rank with single group)**
- **Rank 1, Group 1:** B, C. Updater: Random uniform
- **Rank 2, Group 1:** D, E. Updater: Random uniform
- **Input - initial state:** [1,0,1,1,0]
- **Successors (of initial state):**
  - (a) Rank 1, Group 1: B update - [1,1,1,1,0]
  - (b) Rank 2, Group 1: D update - [1,0,1,0,0]
  - (c) Rank 2, Group 1: E update - [1,0,1,1,1]
- **Expected output:** After 10.000 time steps of the iteration, the only successors visited are: [1,0,1,1,0] → [1,1,1,1,0] → [1,1,0,1,0] → [1,0,0,1,0] (→ [1,0,1,1,0] again). At each time-step, there is at least one available successor from rank 1, either B or C, at the same time rank 2 also has two successors, D and E. Rank 2 successors will never be chosen as there is always a rank 1 successor available.