Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Formal Modelling for Population Dynamics

Suryana Setiawan

Supervisor
Andrea Maggiolo-Schettini

Supervisor
Antonio Cerone

Abstract

The spirit of sustainable development has inspired our research work. Ecologically sustainable development needs preventative strategies and measures against environmental degradation. In our work we focus on constructing a formalism that enables modellers to model the population dynamics within an ecosystem and to analyse them. Furthermore, preventative strategies can be put into the model so that their effectiveness for ecosystems can be measured.

An ecosystem consists of many interacting components. These components have many behaviours which are not easy to put together in a model. Work on such modelling started a long time ago, and even more has been done recently. These approaches have been taken from ordinary differential equations to stochastic processes. There are also some existing formalisms that have already been used for this modelling. In ecosystems there are several important aspects that need to be incorporated into the model, especially: stochasticity, spatiality and parallelism. One formalism has strengths in a certain aspect but weaknesses in others. Being motivated by this situation our work is to construct a formalism that could accommodate these aspects. Besides this, the formalism is intended to facilitate the modellers, who are generally biologists, to define the behaviours in the model in a more intuitive way. This has led our work to adopt features from existing formalisms: Cellular Automata and P Systems. Then, after adding new features, our work results in a new formalism called Grid Systems.

Grid Systems have the spatiality of Cellular Automata but also provide a way to define behaviours differently in each cell (also called membrane) according to the reaction rules of P Systems. Therefore, Grid Systems have a richer spatiality compared to CA and the parallelism and stochaticity of P Systems. Besides these, we incorporate stochastic reaction duration for the reaction rules so that Grid Systems have stochasticity in rule selection and stochasticity in reaction termination. This enables us to define scheduled external events which are important aspects in modelling ecosystems.

In addition to these, we extend Grid Systems with a new feature called 'links'. A link is an object that can carry pointers. The pointer of a link can be used in the rule to transfer objects to another membrane. Because a link is also an object, its existence as well as its pointer are dynamic. By using the links, the membranes of Grid Systems can be structured as a tree to imitate the membrane structure of P Systems, or even more as a graph for a more general computation. The property of the links enables the structure to be dynamic, in a similar way to the dissolving membrane in the P Systems.

The features of Grid Systems are defined in terms of syntax and semantics. The syntax describes how the model should be expressed by the modeller. The semantics describes what will happen to the model when the model evolves. From the semantics a software tool can be developed for analysing the model.

In our research work we have developed the models in two case studies. In the first case study, we focus on the interacting events and external events that affect the population dynamics of mosquitoes. We observe how the impacts of events are propagated in space and time. In the second case study, we focus on the spatiality movement created by the seasonal migration of wildebeests. We observe that the pathways in the migration can be modelled well using links.

The models of both case studies are analysed by using our simulation tool. From both case studies we conclude that our formalism can be used as a modelling framework especially for population dynamics, and in general for analysing the models of ecosystems.

Dedication

To my Lord Jesus Christ,
my source of inspiration.
Thank you for the yearning that you put in me,
to participate in your work in making the world better.
May this be my humbly offering to You.

To my beloved, Fince Anakotta,
and to my beautiful angels Ivy, Liana and Michael.
Thank you for all of your love, pray, support,
encouragement and dedication.
This is a tribute to the four of you.

Acknowledgments

- I would like to thank my supervisors: Professor Andrea Maggiolo-Schetini, Professor Roberto Barbuti, Dr. Antonio Cerone, and Dr. Paolo Milazzo, who have patiently guided me through the phases of my work.

- I would like to thank the members of my committee for their extreme patience in the face of numerous obstacles.

- I would like to thank Professor Pierepaolo Degano and the staff at the Dipartimento di Informatica, Università di Pisa, who have helped me in many matters in this programme.

- I would like to thank my fellow doctoral students: Sara Fernandes, Ruzhen Dong, Ahmad Ibrahim and Sheema Shamin, and to thank the people of IIST/UNU, Macau period 2011-2013: Tomas, Elsa, Wendy, Alice, and Michelle, for their friendship, support, and encouragement.

- I would like to thank Emily C. Fullwood who has patiently proofread and corrected the English of my thesis.

- Last, but certainly not least, I would like to thank the Faculty of Computer Science, University of Indonesia, who has allocated my scholarships from the I-MHERE Project (IBRD Loan No. 4789-IND & IDA Credit No. 4077-IND, Ministry of Education and Culture, Republic of Indonesia) and the United Nations University, International Institute for Software Technology, Macau, who has managed the Joint Programme between IIST/UNU and Università di Pisa.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In 1987 the United Nations (UN), through the World Commission on Environment and Development, expressed its concerns regarding the degradation caused by recent developments all over the world. The concerns were reported in [1]. It also defines: "Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs." This means that each development by the current generation should preserve the resources that may also be needed by the next generations for their own development, and in their turn, they should preserve them in a similar way for the following generations, and so forth. An important component of Sustainable Development, sometimes called Ecologically Sustainable Development, is the use of preventative strategies and measures against environmental degradation [2]. These strategies could be in the form of regulations such as eco-labelling policies, in the form of recovering actions of the degraded resources, or in the form of preserving actions such as those for endangered species populations. The measures are defined in order to indicate effectiveness of the strategies. Our question is how can we ensure that the strategies can prevent the environment from degradation over global and long-term implementation?

Ecology is well-known as one field of study in Biology. Our environment is composed of ecosystems and each ecosystem may consist of many interacting components either living (biotic) or non-living (a-biotic). Biologists study how the components individually behave and influence one another and identify their related parameters. Among them we can cite parameters depending on environmental characteristics (light, humidity, presence or absence of water, ...), parameters depending on periodic or non-periodic events (temperature, level of water bodies, ...), and parameters depending on heterogeneity due to geographical differences over wide area ecosystems (vegetation, elevation, ...). Regarding strategies to be applied to an ecosystem, they can be seen as external components interacting with the internal ones in the ecosystem resulting in a much more complex ecosystem. In answering our previous question, biologists need a method to make predictions about ecosystems, especially when they consist of so many components.

A limited number of species living in the ecosystem can be brought to the laboratory and their behaviours can be observed under given parameters. This method may work for small scale ecosystems. However, some species may behave differently when they are observed individually compared to when they are observed coexisting with other components. In most cases the observation should be conducted in the real habitats (*in vivo* observation), however, such a way unfortunately takes years and covers a wide area. Thus, this often implies that evaluation will be infeasible and very costly.

Biologists then 'put' the ecosystems into mathematical models and analyse the models to predict 'the future' of the actual ecosystems. The classical approaches of modelling are based on differential equations. For example, in 1798 Malthusian Models were proposed for modelling population growth, and in 1926 the predator and prey models of Lotka-Volterra Systems were proposed for modelling fish catching in the Adriatic [41]. Of course, in this era mathematical analysis was usually performed by solving the equations by hand. After computers were available, the methods like Runge-Kutta were used to perform approximation analysis of more complex models. The manuscripts [41, 42] written by Murray listed the models of biological phenomena using some variants of differential equations including Ordinary Differential Equations (ODE). The weakness of this approach is in its difficulty to find precise mathematical models to represent behaviours. Besides this, since the number of components is proportional to the number of variables/equations the model tends to require a lot of computing resources.

There are some natural phenomena that could not be modelled deterministically in terms of differential equations due to their micro behaviours or unknown external factors. For such cases, random variables are introduced into the equations and they turn out to be stochastic processes. A "Birth-and-Death" process is often used for modelling a single-species population dynamic. Markov Chains, a class of stochastic processes, have been used in modelling more general population dynamics.

However, there are some other aspects that need to be considered as part of the models: spatiality and parallelism. Spatiality, in terms of spatial heterogeneity of the components, is often found in wide area ecosystems. Geographical differences imply different parameters of components' behaviours. In Murray's manuscripts [41, 42], over two dimensional space, the position is represented by two parameters (u, v) along with already existing time parameter t. Cellular Automata (CA) was the first formalism in Computer Science that enabled the modelling of spatial movement. In CA the behaviours are defined as evolution functions based on neighbouring cells. Due to this feature some works using CA were reported for modelling vegetation spread over an area [4]. However, due to its definition, a cell in a CA could only represent the state of one species at a time.

Parallelism is an important aspect in population dynamics since nature itself performs the processes instantly and simultaneously. So, every component in the ecosystems should behave and interact in a parallel manner, too. Recently, an approach called individual based modelling (IBM) has become increasingly popular. It can be constructed by Process Calculus in Computer Science. In this approach each component is modelled analogously to a process unit that coexists with other processes and behaves in a parallel manner. BlendX is one modelling language based on this approach [18, 31]. The behaviours in population dynamics can be more simply expressed by using LIME, an interfacing language for BlendX. However, the behaviours are defined to each individual instance of component, a model consisting a big number of components will require heavy computation resources (space and time). Therefore, this approach seems to be more suitable for a model with a smaller number of components. P Systems are the other formalism based on parallelism of behaviours. The behaviours are defined as reaction rules. The components inside the ecosystem can be simply expressed as the objects of P Systems and as the reactions representing the interactions/behaviours among components.

From a biologist's point of view, modelling work eventually requires skills which go beyond their actual field. Differential equation based modelling requires either mathematical analytical ability or simulation programming skills. Individual based modelling requires such a tool-related programming skill. These requirements create a methodological difference regarding their usability except when they are supported by someone with such skills. The commercial analytical tools are available based on certain approaches. For example, Vensim is a tool developed for models based on differential equations and stochastic approaches which was originally intended for modelling in system dynamics. By using such a tool a modeller can put each component into the model as a stock and its quantity will change according to its given growth function. One component can be related to another by creating a 'stock and flow' relationship according to a flowing quantity from the former component to the latter one. Stochasticity is introduced by adding random variables into the growth functions as well as into the quantity.

There is a big gap between the approaches of life scientists and computer scientists. Cerone and Scotti in [16] have described this gap by identifying six aspects:

1. the model description language;
2. the means of model analysis;
3. the motivations for the modeling process;
4. the source and reliability of data;
5. the level of abstraction of the model;
6. the use of the models.

Life scientists prefer to use modeling languages that support a realistic and intuitive description of the systems of interest. They frequently perform statistical inference of the observed data produced from repeated simulations of the model. Formalisms proposed by computer scientists are instead low-level and aim to be used as the input of analysis tools commonly used in computer science, such as model-checking.

Life scientists aim to contribute to their own domain. Thus they develop models that focus on biological aspects and make use of real data to solve concrete problems of their domain. However, such models are often limited in terms of analysis capabilities. Computer scientists are instead more interested in developing notations and tools and see the application domain mainly as a source of case studies to demonstrate the power of their tools. However, such case studies are often oversimplified and based on *ad hoc* rather than realistic data. As a consequence, the outcome of the analysis process can seldom be applied to the real problem.

Life scientists often develop models that include biological details that are often unessential, but that are related with aspects of the modeled systems that they are used to deal with. This makes the models often very intuitive and close to reality, but also difficult to analyze. Computer scientists, instead, tend to develop more abstract models that are focused on the key aspects of the described systems by avoiding unnecessary details.

Life scientists typically use models for better understanding the functioning of the ecosystems and to test possible control interventions, in order to predict and evaluate the impact of such interventions. Computer scientists, instead, often make use of formal models just to replicate the ecosystem evolution without much attention to the evaluation of human intervention impact. To make it worse, since computer scientists typically do not use real data, the ability of replicating reality is mostly just potential and is seldom documented in the literature.

Ideally, there should be a formalism that biologists can use to express the behaviour of the components in the ecosystem in accordance with their intuition. By expressing the model in such a formalism, biologists can communicate the model independently between themselves just like computer programmers communicate their algorithms. Based on the formalism's syntax and semantics various tools could be provided by the software industry.

To conclude this part, based on the nature of ecosystems ecological modelling requires an approach that integrates the ability to define parallelism of the individual components such as in Individual Based Modelling, defining stochastic behaviours of each component such as in stochastic approaches, and the ability to define spatial dynamics of the population such as in Cellular Automata. In order to allow biologists to be more independent in their modelling work there should be a formalism that complies with their intuition.

Regarding our initial issue in ecologically sustainable development, the availability of such a formalism, as well as the availability of its analytical tool, as described above, can answer our question about how to predict the long term impact of the strategies and measures which are applied to an ecosystem. Moreover, when they are available the alternative preventative strategies can also be compared and chosen properly based on their prediction before they are actually implemented.

1.2 Contributions

In this thesis a new formalism is constructed in order to answer the above question. This formalism is named *Grid Systems* and it has been inspired by concepts of Membrane Computing (P Systems) [45, 49].

To accommodate spatiality dynamics a Grid System consists of an envelope compartment (the outer membrane) containing a grid of adjacent inner compartments (inner membranes). Each inner membrane is characterized by its position in the grid, and, as in P Systems, its own rules and objects. The format of the evolution rule in P Systems is adopted to make expressing the behaviours of the components easier. The format consists of four elements: reactants, products, promoters and inhibitors. More complex behaviour can be expressed as several of such evolution rules.

To accommodate parallelism, rules are applied in a maximal parallel manner and can be promoted or inhibited by objects in the system. The outer membrane has rules too, which represent events common to the entire Grid System. Differently from standard P Systems, the application of Grid System rules has duration, which can be different for each rule. This duration also enables each rule to behave stochastically.

Grid Systems are intended to model the heterogeneity of environments. Each membrane in the grid may represent a distinct part of the environment with specific parameters and behavioural rules. Rules can move objects across membranes, thus describing the migration of resources or individuals. Regarding the movement, an extended definition of the Grid Systems is defined which enables the pathways to be modelled in a more elegant way. External events such as environment events, strategies/policies given to the ecosystem can be expressed as the set of evolution rules.

To deal with a large population size and a large number of different components, the semantics of Grid Systems maintains the state of the systems in terms of the aggregation at its cell level. Through this aggregation the detail of the system can be adjusted by setting the cell size/coverage. Therefore, it has the ability to deal with the case of a more real scale population size, but it can evolve the state of the system in more detail than just at its total population level. Because of its maximal parallelism and its locality of the reactions, the semantics of Grid Systems is designed so that it can be implemented to benefit from parallel computing platforms. When the computing system has reached a much higher capacity, the implementation will provide the computing ability for the actual population size.

1.3 Related Works

Formal methods in Computer Science have played an important role such that the models of processes can be formulated mathematically and their behaviour can be studied and analysed accurately. From the initial formulation, a model can be improved in a consistent manner to accommodate further requirements.

Some emerging works in formal methods have been inspired by biological systems by exploiting analogies between concurrent processes and living entities. In Biology each living entity has its own behaviour and it may be involved in interactions with other entities. The application of formal methods to biological systems is reasonable since the computation models tend to be more complex and more dynamic as they are approaching the degree of complexity of biological systems. Namely, running processes in smart phones can be analogously seen as a population of mobile species that are sharing the same communication channels as their resources. Also, cloud computing considers distributed computing resources in the Internet as a single resource interacting dynamically with the user's mobile devices.

On the other hand, these formalisms are powerful tools for biologists to model biological phenomena so that they can be considered as more comprehensive systems rather than partial subsystems. Since biological phenomena are complex systems, biologists have practised reductionist paradigm by subdividing a system into subsystems. When any subsystem is still complex it is further subdivided until biologists can perform an intensive and effective observation on it [52]. This kind of analytical practice may produce an accurate observation of the partial subsystem but not the behaviour of the whole system itself. When the subsystems coexist as a complex system they may behave differently. Since formalisms for concurrent processes have been developed to model such complex systems the use of formalisms in this problem may potentially result in richer observations and conclusions. Systems biology has emerged as a new discipline in putting them together rather than taking them apart (holism paradigm) [43]. The advance of computing technology has reached a tremendous level of performance. The behaviour of the model of such cooperating subsystems in a complex system may potentially be observed through simulations of them.

The π-calculus [39], a process algebra originally proposed for the description of concurrent mobile systems, has also been considered for modelling biological processes in systems biology. Like other process calculi defined in the context of the theory of concurrency, the π-calculus is based on concurrent processes that can communicate through message passing over channels. The peculiarity of the π-calculus is the possibility of communicating channel names in messages thus allowing dynamic re-configuration of the communication network. A process in the π-calculus is defined as a sequence of input and output actions performed over channels. Processes can then be composed by means of a non-deterministic choice operator that allows alternative operations to be described, and by means of a parallel composition operator that allows concurrency to be dealt with.

Since the processes are considered as parallel processes π-calculus can be used to model mechanisms inside a biological system. An example is RTK-MAPK signal transduction pathway [50]. In this example molecules are considered as processes and modelled as of π-calculus terms. Some other formalisms emerged later as extensions of π-calculus considering the aspects that are not covered by π-calculus, namely quantitative properties of the systems. Stochastic π-calculus was proposed in incorporate a probabilistic behaviour in the systems [48].

In π-calculus, spatiality can be expressed implicitly using linkages to the processes which logically represent the location. The extensions of π-calculus were proposed considering the representation of spatiality [30, 13]. Cardelli and Gardner proposed a geometric process algebra based on affine geometry. In this process algebra a concurrent evolution of geometric structures in 3D space can be described. In [47] Philippou and Toro propose PALPS, a process algebra for individual based modelling. This formalism enables one to incorporate spatial information to each species in the model. Galpin [21] introduced spatiality in PEPA, a stochastic process algebra that was previously intended for modelling and performance evaluation [26]. In PEPA spatiality is defined by enriching processes with labels representing locations, whose connections are described as an hypergraph. Locations are used in the operational semantics of PEPA to enable the synchronisation of components.

In his thesis Pardini proposed three forms of formalism considering spatiality [45]. The first formalism is the extension with compartments of MIM calculus (MIMc) [7] while MIMc was a calculus inspired by Molecule Interaction Maps (MIMs) [34]. The MIMc provides high level operators to express the interactions between the processes. Spatiality is represented in terms of compartments.The second and the third formalisms of Pardini's thesis provide a more explicit notion of space. Spatial Calculus of Looping Sequence (Spatial CLS) is an extension of Calculus of Looping Sequence (CLS). CLS is a formalism that was inspired by biological cellular systems [38]. The processes are an analogy of biological cells that are composed of a set of rules of construction. The rules are: sequencing, parallel composition, looping and containment. Looping and containment are together defining nested structures of membranes as in cells. The possible interactions are expressed in terms of rewriting rules. Pardini has extended the descriptive capability of CLS with the notion of spatiality such that each component as well as containment may have their own positional information. With positional terms in each component the interactions may describe spatial dynamics among the components.

The third formalism of Pardini's thesis is a spatial extension of P Systems and it shows modelling of spatial movement of schooling fishes [45]. The model is based on the *boid* flock model that determines the movement direction of each individual fish by a weighted-averaging computation from other fishes' directions inside its viewing space. Such heavy computations were originally intended for computer graphics animation rather than for biological validity [51].

1.4 Structure of the Thesis

The chapters are organized as follows: The thesis is structured as follows.

- In Chapter 2, some relevant notions are presented that will be assumed in the rest of the work.
- In Chapter 3, our formalism, called Grid Systems, which is the main ingredient of the thesis, is presented. This presentation will be preceded by the underlying principles that were taken from the nature of population dynamics so that the features of our formalism could then be defined based on them. To avoid jumping into a complicated definition of Grid Systems a feature of Grid Systems for modelling movement is presented as an extended version of Grid Systems. Therefore, these will be presented respectively in two different sections: the basic version and the extended version.
- In Chapter 4, the expressiveness of Grid Systems regarding other formalisms is discussed. The other formalisms are Turing Machines, Cellular Automata, and P Systems. The motivation behind this discussion is to learn about Grid Systems' computability as well as Grid Systems' back compatibility with Cellular Automata and P Systems.
- In Chapter 5, some considerations for implementing an analytical tool for the model of Grid Systems are presented.
- In Chapter 6, the first case study is presented. The objective is to show how Grid Systems can model the spatial dynamics of a species within its interaction with both internal and external events. It shows the rainfall events and temperature fluctuations as scheduled external events.
- In Chapter 7, the second case study is presented. The objective is to show how Grid Systems can model more extensive spatial dynamics of a species driven by external conditions. The model was based on the phenomena of seasonal migration driven by the grazing availability in Serengeti National Park, Tanzania.
- Finally some conclusions and further discussion are presented in Chapter 8.

1.5 Published Material

Part of the material presented in this book has appeared in some publications, in particular:

- The definition (syntax and semantics) of Grid Systems and the first case study has appeared in [6].
- The definition of the Links as the extension of Grid Systems and the second case study has appeared in [53].

All the published material is presented in this book in a revised and extended form.

Chapter 2

Background

In this chapter some notions related to the work of this thesis are presented. We will start by presenting existing formalisms that have been the main source of inspiration in constructing our formalism: Cellular Automata and P Systems. Then, we will discuss the notions of stochasticity and spatiality since these are considered to be important aspects that need to be provided by our formalism.

2.1 Cellular Automata

A cellular automaton describes the evolutionary change of a spatial shape that evolves over a discrete time line. The changes are performed according to a set of evolution rules that are defined based on the states of neighbouring cells. Andy Adamatzky has formalized CA as follows [3].

Definition 2.1. Cellular automata are dynamic models that are discrete in time, space and state. A simple automaton A is defined by a 4-tuple: $A = (L, Q, \delta, f)$, i.e., a lattice L, a finite state space Q, a neighbourhood template δ and a local transition function f. At time t each cell will be in any state $q_t \in Q$, and in the next time it will change its state following the transition function f which is based on the states of its neighbouring cells inside neighbourhood template δ.

The lattice structure of the cellular automaton may vary from one dimension to n dimensions. The transition function f is usually defined as a set of rules mapped from the neighbourhood patterns to Q, whereas each neighbourhood pattern is formed by the states of neighbour cells scoped in δ. The rules can be deterministic or stochastic. The deterministic rules map each pattern to exactly one state. In the stochastic case, from each pattern there are several possible next states. Furthermore, these states are given a probability regarding whether or not they will happen.

Wolfram has studied the behaviour of 1-D CA for some different transition rules. He concentrated on dual state CA, having [-1,0,+1] neighbourhood. Therefore, the CA requires a set of eight transition rules $C'_i = f(C_{i-1}, C_i, C_{i+1}), C_i \in \{0, 1\}$. There are 256 possible unique sets for defining the rules and Wolfram numbered each set by the binary values given by the rules. One of them is the famous Rule-110. Cooke proves that Rule-110 is a universal machine [17] through showing its ability to simulate Universal Turing Machine indirectly.

The other interesting one is Rule-30 and Figure 2.1 shows the evolutionary result for Rule-30. Rule-30 has transition rules as follow:

$$\begin{aligned} f(0,0,0) &= 0 & f(1,0,0) &= 1 \\ f(0,0,1) &= 1 & f(1,0,1) &= 0 \\ f(0,1,0) &= 1 & f(1,1,0) &= 0 \\ f(0,1,1) &= 1 & f(1,1,1) &= 0 \end{aligned}$$

The cells were initially set to 0 (drawn as a white cell) except one cell in the middle (shown at topmost line). The subsequent lower lines are the result of its step-by-step iterations.

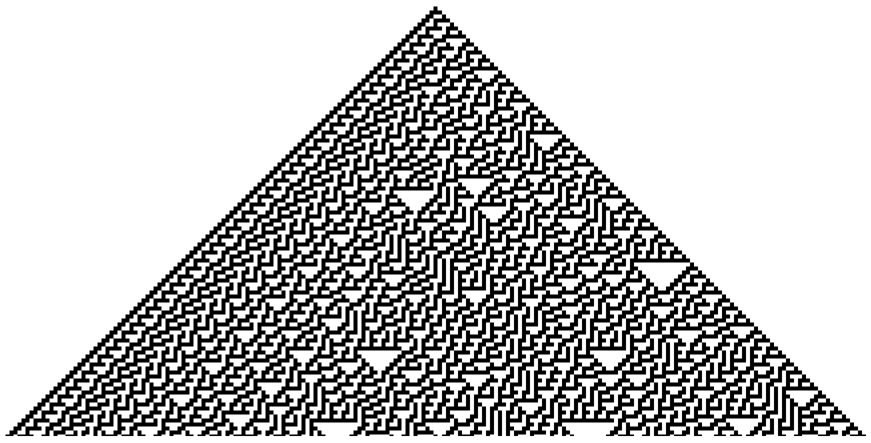


Figure 2.1: CA evolution by Wolfram Rule-30

In the 2-dimensional cases, most popular neighbouring shapes are the von Neumann neighbourhood (also called first-degree neighbourhood) and the Moore neighbourhood (also called second-degree neighbourhood). The former one consists of 4 adjacent cells and the central cell (south-north-west-east) and the latter one consists of 8 adjacent cells and the central cell, i.e., a 3x3 sub-grid.

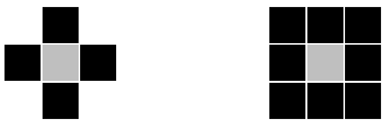


Figure 2.2: Von Neumann's neighbourhood cells (left) and Moore's neighbourhood cells (right) to the central cell (grey colour)

Instead of square grids, the hexagonal plot is also interesting, i.e., each cell has 6 adjacent circular-located neighbours. When using this plot the neighbours have the same distance to the centre which is more natural than the square one. In the implementation this plot can be realized as an ordinary square grid having the odd numbered columns logically shifted to half-cell position lower to the even-numbered columns.

Being inspired by John von Neumann's self-replication machine, John Horton Conway, a British mathematician, in 1970 invented a solitaire game named the Game of Life (GOL). The game uses an orthogonal grid of squared cells. Each cell can be in one of two states: live or die. The neighbour of a cell C is the set of surrounding cells in a 3x3 sub-grid

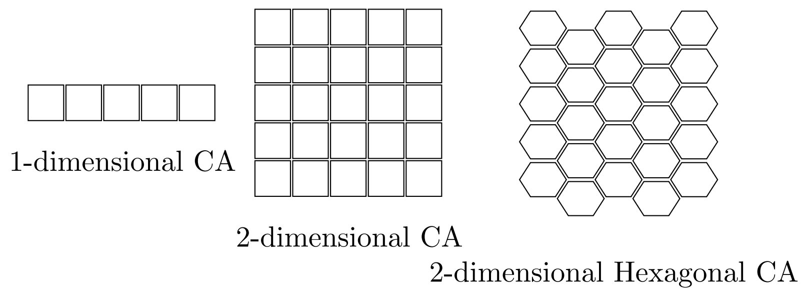


Figure 2.3: Example of lattice structures for CA

(Moore's neighbourhood). A configuration is states of overall cells at one point of time. A configuration will evolve to be another configuration after a discrete unit of time following these rules:

- Any live cell with fewer than two live neighbours dies, as if caused by under-population.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any live cell with more than three live neighbours dies, as if by overcrowding.
- Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Figure 2.4 shows shapes, called "Lightweight Spaceship (LWSS)", generated one after another (from left to right) by applying GOL's rules subsequently. The shapes are regenerated after four steps but at two-cell distance to the right creating an illusion of a moving object. Since it was published by M. Gardner in 1970 [22] it became famous as a field of mathematical research as well as ecological modelling.

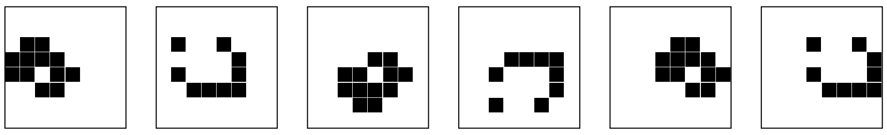


Figure 2.4: Lightweight Spaceship (LWSS) patterns subsequently created by Conway's GOL

CA were used for modelling ecosystems [54, 40, 32, 4]. In his manuscript Tilman discussed CA as one of three approaches for dealing with the spatial aspects of species interactions and population biology [54]. He suggested that CA are best approached via computer simulation. Moreno et.al. have studied population dynamics based on CA in [40]. They used CA for modelling patch occupancy of the species so that the CA could show the percentage of occupancy when the species reaches its stable equilibrium, and also the pattern of occupation. The dynamics of the population, i.e., the evolution rules were derived from classification data produced from the remote sensing methodology of satellite imagery.

The generalisation of CA which is called Complex Automata or CxA, was introduced by Hoekstra *et al.* [28, 27] for modelling multi-scale systems. CxA are composed of some

CA according to some given relationship. This structure enables modelling at several different levels of aggregation and mechanisms and at each level the model is defined as CA. Hoekstra *et al.* have shown, through modelling works, namely coral growth and snow deposition, how coral growth modelling has three levels: (1) growth of coral; (2) flow of water around coral; and (3) advection - diffusion of nutrients. In [5] Cacciagrani *et al.* used CxA for multi-scale trabecular bone remodelling. CxA in their model is composed of two levels: micro execution flow and macro execution flow. Each level is represented by one CA at different rates and with different details and they are linked by micro-macro composition patterns.

2.2 P Systems

Georghe Păun has proposed P Systems, a biologically-inspired model of parallel computing [49]. With a P system, the computation is modelled as the objects that are contained in a series of membranes. The membranes are structured hierarchically such that a membrane may consist of some child membranes or none, and a membrane is within one parent membrane or none. There should be at least one membrane and it is at the root of the hierarchy. The behaviour of the model described by a P System is defined by the set of evolution rules that can be applied to the objects, called reactants, producing the other objects, called products.

He proposed two versions of P Systems: Transition P Systems and P Systems based on Rewriting. The main difference of both lies in the definition of the objects in the evolution rules. In the former the objects are in the forms of multisets. Intuitively a multiset is a collection of objects denoted by their species and the occurrence of each species in the collection. In the latter the objects are in the form of a string $v \in V^*$, V is the set of species. They are semantically different in that in the former the objects are regarded without structural relationship among them and in the latter the objects are regarded as each object is placed inside the string. Since it is based on rewriting, the latter could have more general structures like graphs rather than such linear structures. However, in the following presentation the P Systems are referred to as the definition of Transition P Systems that is based on multisets. We start by defining the multiset formally.

Definition 2.2. Let V be an arbitrary set. A multiset over V is defined as mapping $M : V \to \mathbb{N}$; $M(a)$, for $a \in V$, is the multiplicity of a in the multiset M. The facts in M should be expressed as pairs $(a, M(a))$ for each $a \in V$ and (preferably) for $M(a) > 0$. And, all facts in M should be expressed as the sequence of those pairs.

The pairs $(a, M(a))$ may also be written as $a^{M(a)}$. Therefore, they will be used interchangeably in this document. However, for its consistency each multiset should be expressed uniformly using the same style. Furthermore, we need to define operations over multisets as follows.

Definition 2.3. Over the multisets we define that:

- Multiset M is **empty** if and only if $\{a \in V | M(a) > 0\} = \emptyset$. An empty multiset is denoted by symbol λ.
- Let M_1 and M_2 be multisets over V. We say that M_1 is **included in** M_2, or simply $M_1 \subseteq M_2$, if and only if $M_1(a) \leq M_2(a)$, for all $a \in V$.

- The **union of** $M1$ and M_2 is the multiset $M_1 \cup M_2 = M_3$ such that $M_3(a) = M_1(a) + M_2(a)$, for all $a \in V$.
- The **difference** $M_1 \setminus M_2 = M_3$ is defined only when $M_2 \leq M_1$ and $M_3(a) = M_1(a) - M_2(a)$, for all $a \in V$.

Then, we define P Systems according to its references [49].

Definition 2.4. A P System is a tuple $\Pi = (V, \mu, w_1, ..., w_n, (R_1, \rho_1), ..., (R_n, \rho_n), i_0)$ where:

- V is a finite alphabet whose elements are called objects;
- $\mu \subset \mathbb{N} \times \mathbb{N}$ describes the tree structure of membranes, where $(i, j) \in \mu$ denotes that the membrane labelled by j is contained directly in the membrane labelled by i;
- w_i, with $1 \leq i \leq n$, are strings from V^* representing the multiset over V associated initially with membranes $1, 2, ..., n$ of μ;
- (R_i, ρ_i), with $1 \leq i \leq n$, R_i are finite sets of evolution rules (defined in Definition 2.5) over V associated with membranes $1, 2, ..., n$ of μ; ρ_i is a partial order relation over R_i, $1 \leq i \leq n$, specifying a priority relation among rules of R_i;
- $i_0, \in \{1, .., n\}$, specifies the output membrane of Π.

The set R_i defines the rules that are associated to a membrane labelled i. Each rule in R_i is applicable only to the reactants of that membrane. When a rule is applied, its reactants will be removed and the products will be produced which may be either in the same membrane or in the outer membrane or in a child membrane. This situation is analogous to chemicals that interact and pass through cell membranes.

The dynamics are expressed in terms of evolution rules that change the reactants in the same membrane as the rules are defined.

Definition 2.5. An evolution rule is a pair (u, v), also in the form $u \to v$, where u is a multiset over V and $v = v'$ or $v = v'\delta$ where v' is a multiset over

$$(V \times \{here, out\}) \cup (V \times \{in_j | 1 \leq j \leq n\})$$

Special symbol δ indicates that the reaction should dissolve its membrane and move everything inside the membrane to its parent membrane. The produced (s, p), $s \in V$ and $p \in \{out\} \cup \{in_j | 1 \leq j \leq n\}$, can also be written in form s_p and $(s, here)$ as s.

Some examples of evolution rules are as follows.

- The rule $a \to (b, here)$, or $a \to b$, will consume one a and produce one b in the same membrane.
- The rule $a \to (b, in_k)$, or $a \to b_{in_k}$, will produce one b in a child membrane k.
- The rule $a \to (b, out)$, or $a \to b_{out}$ will produce one b in its parent membrane (child membranes will belong to the parent membrane).

- The rule $a \rightarrow b\delta$ is similar to $a \rightarrow b$ but it will be followed by dissolving the membrane and moving everything (objects and child membranes) it contains to its parent membrane.

The set of rules can be deterministic or non-deterministic. Rules are non-deterministic if for the same reactants there is more than one possible applicable rule instead of one, as in the deterministic case. When the reactants are available any rule can be chosen such that there are more than one possible trajectories of applying the rules until there is no rule to apply. Rules are applied with maximal parallelism, namely at each step rules are applied in parallel so that there are no objects that are reactants of the same rule and that are left unchanged. Some variants of P Systems include a notion of priority among rules. The notation $r_1 > r_2$ means that r_1 has a higher priority than r_2 and hence r_2 is applicable only if r_1 is not.

Figure 2.5 shows an example of a P System. It has three membranes defined one inside another: membrane-1 is the outer one; membrane-2 is the inside membrane-1 and contains membrane-3. Rules r_1, r_2 and r_3 belong to membrane-3 while rules r_4, r_5, r_6 and r_7 belong to mebrane-2 and rule r_8 belongs to membrane-3. Non-determinism is defined since r_1 and r_2 have the same reactants whereas r_6 and r_7 have the same reactants but they are deterministic since the priority of r_6 is higher than r_7. Rule r_8 will consume one e and produce one e outside its membrane (i.e., this is the only way to move the object).

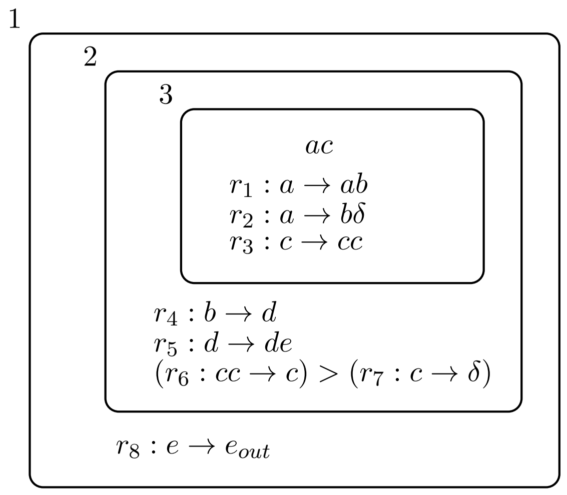

1
2
3
ac
$r_1 : a \rightarrow ab$
$r_2 : a \rightarrow b\delta$
$r_3 : c \rightarrow cc$
$r_4 : b \rightarrow d$
$r_5 : d \rightarrow de$
$(r_6 : cc \rightarrow c) > (r_7 : c \rightarrow \delta)$
$r_8 : e \rightarrow e_{out}$

Figure 2.5: An example of P System

One possible trajectory when applying the rules is as follows. Initially there are only objects a and c in membrane-3. For object a there are two possible rules whereas there is only one rule for c. Let us apply r_1 and r_3 first such that there are one a, one b and two c's (denoted $abcc$ or abc^2). Applying r_1 and r_3 in parallel is based on the principle of maximal parallelism. Then, by applying r_2 and r_3, abc^2 becomes b^2c^4 and membrane-3 is then dissolved so that the objects belong to membrane-2. At this stage there is only one possible combination of rules to be applied: r_4 (twice) and r_6 (twice) since r_7 has a lower priority than r_6. Applying them will produce d^2c^2. After the next step the objects will become d^2e^2c and then finally d^2 in membrane 1 and e^4 outside membrane 1.

Inhibitor objects and promoter objects for the reactions in P Systems were also introduced in [11]. The rules containing promoters can be applied only when the promoters also exist as well as the reactants. The promoters act as catalysers for the reaction since they are not absorbed by that reaction. The inhibitors are the ones whose existence could disable the reaction.

2.3 Notions of Stochasticity

Natural phenomena might be composed of a small number of well-known low level processes. For such phenomena the processes might be modelled as some differential equations. Then, the modellers could analyse the equations mathematically or run them in a simulator to reveal the behaviour of the phenomena. Many successful works were reported using this approach.

For example, given one simple reversible chemical reaction:

$$2A \underset{c_2}{\overset{c_1}{\rightleftharpoons}} B$$

The rate of production of species B is:

$$\frac{dB^+}{dt} = c_1[A]^2$$

The rate of destruction of species B is:

$$\frac{dB^-}{dt} = c_2[B]$$

Its equilibrium is just simply found from the equation of both rates $\frac{dB^+}{dt} = \frac{dB^-}{dt}$, i.e., $c_1[A]^2 = c_2[B]$.

2.3.1 Markov Chains

However, in general situations the number of processes might not be very small. In terms of computation the model becomes intractable. Moreover, if some of those processes are still unknown, it becomes impossible to put them into a single model. Mathematicians provide their approach in modelling these complex phenomena in terms of stochastic processes. A stochastic process is defined as a set of random variables $\{X(t)|t \in T\}$. The notion of T can be either of a time or of a space dimension. The notion of $X(t)$ is also called the state of the process and its range can be either discrete or continuous. Also, when the range is discrete, it can be finitely or infinitely countable. Markov processes are the models for representing the dynamics of the state transition problems whose random variables for the next state rely only on the ones from the current state. The probability of state changes is expressed as:

$$P\big[X(t_i) = x_i|X(t_{i-1}) = x_{i-1}, X(t_{i-2}) = x_{i-2}, ..., X(t_0) = x_0\big] = \\ P\big[X(t_i) = x_i|X(t_{i-1}) = x_{i-1}\big]$$

This assumption reduces the problem complexity by forgetting the history of the process except the most recent one. This is known as Markov (memoryless) property. The famous class of Markov processes is Markov Chains which define the state space to be discrete. Moreover, DTMC's are Markov Chains with discrete time/space while CTMC's have a continuous time/space. By discrete states the models can be represented as graphs where the vertices represent the states and the edges as the state transition relations.

The DTMC's usually use the notion of index instead of time/space, i.e., $\{X_t | t \in \mathbb{N}\}$. Finite state DTMCs are called "homogeneous in time" when each probability has a fixed value, i.e., the notion of time is no longer needed. So, $p_{ij} = P\big[X_j | X_i\big]$, for $i, j \in T$, creating a probability transition matrix. Due to its simplicity, the properties of finite DTMC's can be further explored.

The Birth-and-Death (B&D) Processes are CTMCs that are often used for modelling simple (one species) population growths. The B&D processes limit the state changes to only the states of the right next values; i.e., $X(t+1)$ can be only either $X(t)+1$ ("birth") or $X(t)-1$ ("death"), where $X(t+1) \geq 1$. A B&D process can be depicted as a graph which has linear chained vertices, as shown in Figure 2.6. The edges represent transitions whose probabilities are denoted by $\lambda_n(t)$ for birth transitions and $\mu_n(t)$ for death transitions (of course, in most real population problem it is likely that $\lambda_0(t) = 0$).

$$\begin{tikzcd}
0 \arrow[r, bend left, "\lambda_0(t)"] & 1 \arrow[l, bend left, "\mu_1(t)"] \arrow[r, bend left, "\lambda_1(t)"] & \cdots & n-1 \arrow[l, bend left, "\mu_{n-1}(t)"] \arrow[r, bend left, "\lambda_{n-1}(t)"] & n \arrow[l, bend left, "\mu_n(t)"] \arrow[r, bend left, "\lambda_n(t)"] & n+1 \arrow[l, bend left, "\mu_{n+1}(t)"] \arrow[r, bend left, "\lambda_{n+1}(t)"] & \cdots
\end{tikzcd}$$

Figure 2.6: Birth and death process for modelling population growth

Random Walks can be seen as CTMCs whose states are the vectors of some values $X(t) = (x_1(t), x_2(t), .., x_d(t)), t \in T$. For instance, the states of a 2-dimensional random walk are the points at the integral positions in a plane. Random Walks limit the transition to only the change of one value from $0, \pm 1, \pm 2, ...$ at a time. From this point of view, B&D processes can be seen as 1-dimensional Random Walks of ± 1 changes. Figure 2.7 shows an example of a pathway made by a Random Walk.

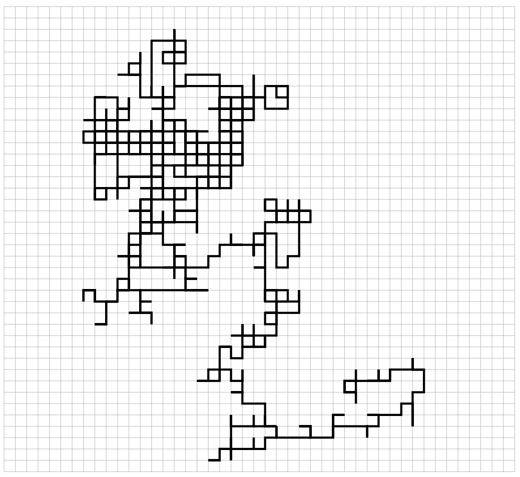


Figure 2.7: An example of a 2-dimensional random walk pathway

The problem in modelling real-world cases using Markov Chains is the explosion of the state space. Maria Mateescu, in her thesis [37], reported two approaches to deal with this problem: the sliding window method, for Continuous Time Markov Chains, and the general computational framework of propagation models, for Discrete Time Markov Chains. Based on the framework of propagation models, Călin C. Guet *et al.* [24] proposed an extension of CTMC with a notion of delay for considering non-instantaneous events in the modelling of genetic regulatory circuits. In their report they observed that the simulations of models with delays produced more realistic outcomes than when the models were based on pure CTMCs.

2.3.2 Gillespie's SSA

When a model contains more complex behaviour and more interacting components the model cannot be analysed efficiently anymore. The simulation of the model is used more frequently than a mathematical analysis. In [23] Daniel T. Gillespie proposed a method for simulating models of chemical reactions. His algorithm is formulated by seeing the reactions as the transition mechanism to walk around the state space. The state space is n-dimensional state space where n is the number of species involved in the reaction both as reactants and as products. At a small interval time $(t, t+\tau)$ only one single reaction can take place. From every one state there might be some M possible reactions to be applied. The likelihood that the reaction R_μ at next interval time τ to be taken is based on reaction probability density functions:

$$P[\tau, \mu] = h_\mu c_\mu \exp\left[-\sum_{v=1}^{M} h_v c_v \tau\right]$$

The quantity $h_\mu c_\mu$ is called the propensity of the reaction R_μ. Factor h_μ is the number of distinct reactant combinations for R_μ (according to stoichiometry coefficients of the reactant) which can be found in the current state of the system at time t. And, factor c_μ is the reaction's rate r_μ.

Considering the previous example of chemical reactions, given that at the current time the number of species A is 10 and the number of species B is 4, also, $c_1 = 100$, and $c_2 = 75$.

$$\begin{aligned} a_{2A\to B} &= \frac{10!}{2!(10-2)!}100 = 4500 \\ a_{B\to 2A} &= \frac{4!}{1!3!}75 = 300 \end{aligned}$$

The nature of this method is to simulate the reactions in a sequential manner. Therefore, the original algorithm runs simulation in cycles until there is no more reaction to be performed, or when it reaches a given time. During initialization, $t = 0$ and every initial propensity $a_i = h_i c_i$ of reaction R_i is computed. At every time t one cycle/iteration will be performed consisting of actions (according to the Direct Method):

1) computing random variable τ (duration of the next reaction), and determining μ (selected reaction) weighted by the propensities;

2) advancing time t by τ, and applying selected reaction change the system's state accordingly; and

3) re-computing propensities $a_i = h_i c_i$ of each reaction R_i.

The interval time τ is exponentially distributed over all reactions, i.e., its rate is $a = \sum_{i=1}^{M} a_i$, which is the total propensity. The variants of the algorithm are due to the optimization of the basic operations above in providing a better computing performance. Some variants propose more efficient data structures for step-2. Some other variants propose *tau-leaping* approaches that reduce the computation of step-3.

Stochasticity was introduced into P Systems such that non-deterministic rules may be associated to a real number representing the rule's probability. Some simple approaches have just proposed fixed valued probabilities by assuming the models are Markov chains in 'homogeneous in time' probabilities. Its example is the model that was reported by Cardona *et al.* in [14].

A dynamic probabilistic approach was proposed by Pescini *et al.* in [46] that is called Dynamic Probabilistic P Systems (DPP). In their approach the probabilities associated to the rules vary during the evolution of the system according to the propensity function of Gillespie's SSA. Afterwards, in case of non-determinism, the probabilities of competing reactions of the instance of time are computed by normalizing those propensities and a reaction is selected based on those probabilities in a parallel manner. In DPP an evolution rule r is redefined including the parameter k as a reaction rate factor in propensity function, i.e., the rule is written in the form:

$$u \xrightarrow{k} v, \text{ for } u \text{ and } v \text{ as defined in Definition 2.5.}$$

In [15] Cazzaniga *et al.* reported their review on a strategy which employs Gillespie's SSA (see Section 2.3) to conduct the reaction in each membrane. It was also compared to the implementation of the DDP to the same case and using the same definition for the rules; appending the reaction rate parameter to the rule. The SSA strategy can solve non-determinism due to the reactions competing with each other to get the resource stochastically, which means that the reaction will be selected probabilistically based on their propensities. However, since the algorithm performs each reaction sequentially the feature of parallelism of P Systems is diminished at the membrane level but the reactions of different membranes remain in a parallel manner.

In [8] Barrio *et al.* introduced delays as an approach for understanding processes, such as transcription and translation in chemical kinetics, that do not occur instantaneously. In their models there are non-delayed reactions and delayed reactions. A non-delayed reaction is like the ones treated in the original SSA. A delayed reaction is such that once selected it will be applied after a given time interval τ.

They extended the SSA to deal also with delayed reactions. The extended algorithm, called Delayed SSA (DSSA), is as follows. When, at time=t, step size Θ has been determined, and there are previously selected delayed reactions that are expected to be applied in interval $(t, t + \theta)$, the earliest one, say reaction k at time $t + \tau$, is applied by updating time $t = t + \tau$ and system state $X(t + \tau) = X(t) - v_k$. If no previously selected delayed reaction is expected to be applied in the interval $(t, t + \theta)$, the next reaction is selected as usual. The selected reaction can be either a non-delayed reaction, which will be applied at $t + \theta$, or a delayed reaction, which will be scheduled at $t + \theta + \tau$ (and be applied as in the above procedure).

In [8] they also described delay models to explain oscillations in numbers of *hes1* mRNA and Hes1 protein in mouse. Furthermore, they observed that their approach produces models that are more accurate than continuous deterministic models (Delayed Differential Equations) of the same case.

2.4 Notions of Spatiality

The Population dynamics of species living in a wider range area may be influenced by spatial differences in their habitats. Incorporating spatiality related behaviours into the model is a must in such conditions.

In [41] a simple model for the spatial spread of an epidemic based on differential equations is elaborated. Let X be the space variable and t be time variable. The population is differentiated into two sub-populations: infective $I(X,t)$ and susceptible $S(X,t)$ sub-populations. Spatial dispersal of I and S is modelled by the following equations:

$$\begin{aligned} \frac{\partial S}{\partial t} &= -rIS + D\nabla^2 S, \\ \frac{\partial I}{\partial t} &= rIS - aI + D\nabla^2 I, \end{aligned}$$

where a, I, and D are positive constants. r is a measure of the transmission efficiency of the disease from an infective one to a susceptible one, $1/a$ is life expectancy of an infective member of the population, and D is diffusion coefficient.

Cellular Automata as already discussed in Section 2.1 have the ability to show such spatial dynamics. However, each cell can only hold the state of a single species as the single vegetation was modelled in [40]. Moreover, the rules homogeneously define on overall grid. In [4] Bazter *et al.* proposed a variant of Cellular Automata that is derived as a spatio-temporal Markov Chain (STMC). The behaviour of the model is based on its temporal and spatial orders. It was used to model the population dynamics of three plant species on a lawn.

Spatiality was incorporated into process calculus formalisms by labelling the structures or compartments as the vertices in a graph without a geometric notion. The processes/rules associate the labels to their entities indicating which structure/compartment they belong to. If the number of places is too big, this explicit labelling may result in a complicated model. CLS is a formalism based on term rewriting which allows a direct representation of membranes in the syntax of calculus [38]. CLS defines the structures in the system by a set of rewriting rules constructed according to CLS grammar. By these rewriting rules, a structure is composed either of a simple sequence of symbols (the smallest objects in the system), or a parallel composition of substructures, or in the form of '*branes* inside a substructure. A *brane* is a compartment containing either a sequence of objects or a parallel composition of other *branes*. Through rewritings the system evolves from one structure to the next structure. In [45] Pardini proposed a spatial extension of CLS, named Spatial CLS. In this extension spatial properties (either in 2-D or in 3-D) are attached to the sequences or to the branes. These spatial properties may influence the effectiveness of the rules. This type of spatiality is very useful in describing formations of geometric and structural entities, namely DNA molecules.

Also, in [45] Giovanni Pardini incorporated spatiality into P Systems, and then he called them Spatial P Systems (SP Systems). Intuitively, in SP Systems the membranes are defined in forms of geometric shapes and each object is located on a non-negative integral coordinate inside its membrane. All membranes, as well as the objects, use one common non-negative integral coordinate system. The shape of a membrane is defined by its boundary given as a circular list of its coordinates. The membranes cannot overlap one another, although a child membrane must be fully inside (there is no shared boundary position) its parent membrane. The objects carry coordinates as their attribute. Some objects located in one coordinate can be referred to as a multiset attributed by the coordinate. The objects located inside a membrane can be referred to as a string of several multisets and each one is attributed by its coordinate. Figure 2.8 shows an example of SP Systems. Membrane-1 has two child membranes: membrane-2 and membrane-3. Let the coordinate system have position (0,0) at the bottom-left position of membrane-1. Membrane-2 has three object a's, they are at positions (2,2), (3,2) and (3,1) respectively. Membrane-3 has one object a and one object c, they are together at position (5,3).

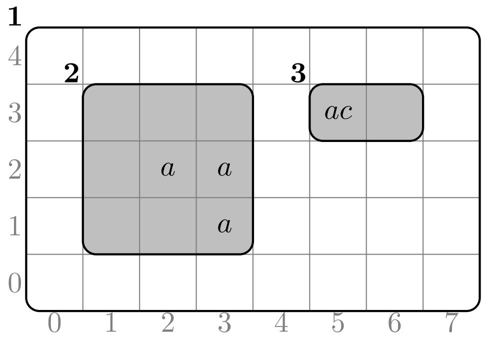


Figure 2.8: An example of Spatial P System

As implied above, the objects in the rules are also attributed by coordinates. The coordinates in a rule that is associated to a membrane are expressed relatively to one position inside that membrane. As in the P Systems, each membrane is associated to some rules but each rule's applicability is related to positions inside the membrane, as follows. A rule is applicable to a position inside its associated membrane when the reactants at the specified positions are available, and the promoters at their specified positions are available, and the specified positions for the products are valid. A position for produced objects is valid when it is inside the targeted membrane. As in P Systems the targeted membrane can be either the same membrane, or parent's membrane, or child's membrane.

The complexity is increased by distinguishing new additional concepts of object occupancy in its position. A mutually exclusive object occupies a position exclusively while an ordinary object may share it with other objects. This concept will limit further applicability of the rules when the target position is already occupied by a mutually exclusive object.

Refer to Figure 2.8, a rule $r = a \to (b)_{(-1,0)}(c)_{out_{(1,0)}}$ that associated to Membrane-2, can only be applied as follows. Rule r is applicable to an object a in position p, and it would produce an object b in position $p + (-1, 0)$, and an object c being sent out to its parent membrane along the direction $(0, 1)$. According to the situation depicted by the figure, r is only applicable to a in $(3, 1)$ and a in $(3, 2)$ but not to a in $(2, 2)$. The reactions will remove both a's, and put b in $(2, 1)$ and $(2, 2)$ and put c in $(4, 1)$ and $(4, 2)$. The rule is not applicable to a in $(2, 2)$ since there is no satisfying location to produce c, that is, the location should be outside of Membrane-2 but in position $(3, 2)$.

In [45], Pardini has shown that the evolution of "ring species" based on small changes between geographically contiguous population can be modelled by using SP Systems. He has also shown that the schooling behaviour of fishes according to *boids* model, a 3-D flock motion from Reynolds [51], can be modelled also by using SP Systems. The fish are the mutually exclusive objects that can be exclusively placed in one position. Each fish is in its state determined by position q and heading direction d. The model rules are described by rules of the form:

$$(d)_{(0,0)}[(\pi_1)_{q_1}...(\pi_\gamma)_{q_\gamma}] \rightarrow (d')_q$$

where $d, d' \in \{1, ..., 8\}$, and $q \in \mathbb{Z}^2$. When the rule is applied the state will change to the next state according to some forces caused by the states of the surrounding fishes: repulsive separation force, attractive cohesion force, and alignment force, geometrically. Thus, the model should be supported by geometric functions for computing the forces and inferring them into the fish's next state.

In [12] Cacciagrano *et al.* alse used Spatial P Systems for multi-scale bone remodelling. Bone remodelling is the continuous replacement of old bone by new tissue. The spatiality in their works was due to spatial processes in remodelling. For example, coalescence process, the formation of osteoclasts cells (a kind of bone cells) is possible only when a sufficient number of pre-osteoclasts are available and when they are quite close to each other. They also worked previously using CxA in [5] and they mentioned that compared to CxA, SP Systems are able to handle an infinite number of different states, namely the number of objects that can be located in each cell. They also found that defining update rules through SP Systems seems to be more expressive than the single deterministic update rule for CxA.

However, the rules in SP Systems are associated to the membranes so that spatial heterogeneity among different locations inside one single membrane should be expressed in terms of regions, by defining child membranes having specific rules. For highly heterogeneous situations the hierarchical structure of its SP System could be very complicated.

Chapter 3

Grid Systems

In this chapter the main contribution of this thesis, which is called Grid Systems, will be presented. For its readability, it will be preceded by an intuitive elaboration on how the main key idea and expected features are identified. Also, the definition of Grid Systems will be presented in two sections: its basic definition and its extended definition including "links". In each part some examples will be inserted mostly taken from our published papers.

3.1 Key Ideas

The need to construct a formalism that is suitable for the modelling of population dynamics has been mentioned in Chapter 1. This was specified to comply with modellers' intuition, where they are expectedly Biologists rather than Computer Scientists. Chapter1 summarized how formalism should make it possible to define the parallelism of components, as well as their stochastic behaviours, and their spatial dynamics in these ecosystems. In this section those notions will be discussed in more detail.

3.1.1 Partition of Space as a Grid of Cells

The space is the area where the model of the ecosystem will take place. The idea of the grid of cells in CA can represent this space well since the size represented by a cell can be set according to the modelling needs. This partitioning enables the model of CA to behave differently on each cell according to the states of that cell and the neighbouring cells. If more detail of spatiality is needed then this size of the cell could be made smaller with the cost of higher complexity in analysing the model. On the contrary, the size could be made bigger providing a less accurate spatial model. Therefore, a cell could represent the smallest space of the system with considerably homogenous behaviour without losing too much accuracy.

3.1.2 Representation of Components of the Ecosystem and Their Behaviour

Basically in ecological cases the objects are the components of the ecosystem including both biotic (living) components and a-biotic (non-living) components. In basic forms of CA the objects are of one kind and each cell is defined as a finite state automaton whose state represents the number of objects inside the cell. Therefore, the state space of each cell is a finite set of non-negative integers. The transition rules are defined uniformly for each cell and the rules are defined for each possible combination of the state of a cell and the states of its neighbouring cells. In our formalism we need a representation of multiple kinds of objects in the system. Respectively, each object type in a cell could have its own behaviour. This specification is satisfied by the idea from P Systems. Besides this, there might be some properties in the ecosystems that affect some other objects located in the cells, namely the air temperature in the area. This aspect requires a global membrane where the environmental objects, such as the temperature level can be placed. This place should be named environmental/global membrane which is analogous to the root membrane of P Systems. Through this representation each cell will be in a state which is represented by a multiset over the objects and the overall grid of cells can be seen as a table of the multisets plus a multiset of the global membrane.

3.1.3 Enabling Duration in Applying the Rule

The behaviour of objects may need to take place at a time interval. The time interval or the duration might be either fixed or randomized. While, both in basic forms of CA and in basic forms of P Systems the rules are intended to take place in one single step. For our formalism the behaviour should be able to be defined as having such duration.

3.1.4 Parallelism

In nature, objects behave simultaneously. They may inhibit or promote the others to behave. To some extent, P Systems enable us to model this behaviour. Our formalism should enable this as well and combined with the duration our formalism should also define its semantics carefully.

3.1.5 Stochasticity for Non-determinism

Some complex behaviour in nature may be in non-deterministic situations, namely there is more than one possibility of behaviour. One of them may be more likely to happen than others. Such non-determinism should be resolved in such a way considering the likelihood given in the model.

3.1.6 Further Spatial Dynamics

Recall that in CA the set of transition rules is defined for all cells uniformly. Its spatial dynamic behaviour results from the current state of the cell among the states of its neighbouring cells. This situation causes behaviour of the model which is not as dynamic as the natural world. To enable Grid Systems to have more spatially dynamic behaviour, our formalism should enable us to define applicability to the cell heterogeneously. As in P Systems objects from different cells could have different behaviours.

3.2 Grid Systems

A grid system divides the space into a two dimensional *grid of cells* adopting the idea of cells in Cellular Automata. Each cell, or sometimes called *membrane*, is addressed by an ordered pair of natural numbers: *row number* and *column number*. The term *membrane* comes from P Systems and the term *cell* comes from Cellular Automata. Formally the dimensions are infinite, both in row-wise and column-wise directions. However, in a real modelling work of an ecosystem they would normally be finite and inferred from the model itself. In a grid system the objects are represented by a set of unique symbols Σ and the state of a cell is represented by a multiset over Σ.

Similar to CA and P Systems, the behaviour of Grid Systems is described by reaction rules. Note that, for some contextual purposes they can be called *transition rules*, as for spatial movements, or *evolution rules*, as for transformation in P Systems. However, *reaction rules* will be the most used terminology in the rest of this thesis. The definition of the reaction rules in Grid Systems adopts of the rules of P Systems albeit with some differences. To enable Grid Systems to have more spatially dynamic behaviour, the applicability of the rules is defined by means of a set of associations of rules with membranes.

In Grid Systems a reaction will be performed over duration from being applied until being completed. Therefore, the state of a grid system at time t is also defined by the set of reactions that at time t are already applied but not yet completed apart from the objects that are in the cells. An initial configuration should be given so that the system can evolve starting from this initial configuration.

The following definition formally defines Grid Systems. The parts of the definition will be defined in its following subsections.

Definition 3.1. A Grid System $G = (\Sigma, R, A, C^{(0)})$ is defined as follows:

- G is the Grid System name;
- Σ is a finite set of symbols representing the alphabet of object types;
- R is a finite set of reaction rules (see Definition 3.6);
- A is the set of associations of the rules with the membranes, i.e.

  $$A = \{(\rho, \gamma) \mid \rho \in R,\ \gamma \in \{G_{i,j} \mid i, j \geq 0\} \cup \{G_E\}\};$$

  where

  - $G_{i,j}$, with $i, j \geq 0$, called local membrane, denotes the cell in position (i, j);
  - G_E is the global membrane surrounding the cells;
- $C^{(0)}$ is the initial configuration of the Grid System (see Definition 3.9).

Note that, sometimes the cells in Grid Systems are bounded to $N \times M$ by the nature of the model itself. For the actual implementation of its analytical tool a bounded version of Grid Systems, $G = (N, M, \Sigma, R, A, C^{(0)})$, can be used instead, by specifying N and M, where $N, M \in \mathbb{N}$, such that the number of membranes is $N \times M$ and the association set is:

$$A = \{(\rho, \gamma) \mid \rho \in R,\ \gamma \in \{G_{i,j} \mid 0 \leq i < N,\ 0 \leq j < M\} \cup \{G_E\}\}$$

3.2.1 The Grid of Membranes

Local membranes of a Grid System will be referred to by the grid name G and subscripted by the pair membrane's row number and column number. Each number will start from 0, 1, ... and $G_{0,0}$ refers to the topmost and left most membrane. The global membrane will be referred to by the Grid System name G, too, but it is subscripted by the letter 'E'. A graphical representation of a Grid System is illustrated in Figure 3.1. The global membrane describes the environment and is drawn as the outer square (drawn by the rounded square). Local membranes are represented by a grid of squares. The two associations $(r_k, G_{3,1})$ and $(r_k, G_{2,3})$ in Figure 3.1 are the examples of associations indicating that r_k is applicable in local membranes $G_{3,1}$ and $G_{2,3}$.

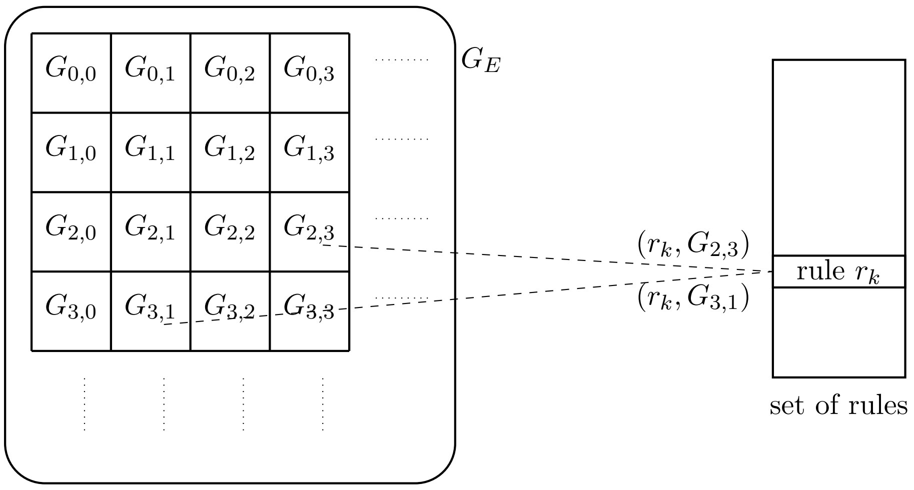


Figure 3.1: Grid of membranes and association of rules with membranes

3.2.2 Objects and Multiset over Objects

Objects are the important entities since the number of their occurrence will change from time to time reflecting the system's dynamics. The objects are categorized according to their behaviour in the system and in the model each category is represented by a unique name/symbol. Σ is the finite set of unique names representing all those categories.

The objects are not treated individually. Their existence is aggregated based on their locality inside the same membrane as the multisets over object types Σ. The definition of multisets refers to Definition 2.2.

Example 3.1 (A hypothetical case)**.** Let us define a hypothetical ecosystem containing an insect species as the main object category. The insects have three stages of metamorphosing: egg, immature (pupa, larva), and adult. For the moment we disregard its sex. The population dynamics of this species at each stage is controlled by the existence of predators in their habitat as well as by natural causes. Only adults can lay a certain number of eggs. The existence of water is needed for their eggs to hatch. Water will be represented by its discrete volume levels W ranging from 0 to 4 from the driest to the wettest. Water evaporates at a rate dependent on environmental temperature. The temperature, as for water, is represented by a number of object T's ranging from 0 to 5 from the coldest to the hottest. The volume of water increases by the rainfall and the rainfall is represented by object R. The temperature increases due to external event U and decreases due to external event D. To shorten the notations let us use their initial letters (in capital letters) for the object identities in Σ. Therefore, $\Sigma = \{E, I, A, P, W, R, T, U, D\}$. For instance, if there are 50 Eggs, 20 Immature mosquitoes, 8 Adult mosquitoes, and 2 Predators along with Water of level 3 in one place, those objects are represented by a multiset $E^{50}\ I^{20}\ A^{8}\ P^{2}\ W^{3}$. Note that, this example will be referred to in subsequent examples in related parts of the discussion.

The ecosystem covers a bounded area that is subdivided into 5×5 cells. Therefore, the membranes are: local membranes $G_{0,0}, ..., G_{4,4}$, and environmental membrane G_E.

This case will also be referred to by the subsequent examples that are related to their respective parts of the discussion. Note that the examples are not intended to be so biologically precise. They are rather illustrations of the idea being given at its related part.

3.2.3 Reaction Rules

The behaviour of the system is defined by reaction rules that rewrite a given multiset of objects into a new multiset of objects. The reaction ρ is expressed as a chemical reaction-like expression:

$$\rho :\ \alpha \rightarrow \beta$$

The notation expresses that multiset α represents the objects that are consumed, called *reactants* and multiset β represents the objects that are produced, called *products*. Note that ρ is a unique identifier to that rule to be referred to in the discussion. Also, in order to discuss behaviour sometimes ρ can be omitted.

Example 3.2. A reaction rule

$$r_0 :\ A\ \rightarrow\ A\ E^{10}$$

specifies that each existing adult insect will lay 10 eggs in the membrane where the insect is currently staying.

Some objects, called *promoters*, may be required for the reaction to occur although they are not consumed when the reaction occurs; other objects, called *inhibitors*, may instead inhibit the reaction. A reaction rule includes promoters ψ and inhibitors χ:

$$\rho :\ \alpha \rightarrow \beta\ [\psi \mid \chi]$$

Example 3.3. Reaction rule

$$E \rightarrow I\ [\ W \mid W^4]$$

specifies that an egg will hatch to be an immature mosquito when there is water and the number of W's is between 1 and 3 inclusively. However,

$$E \rightarrow I\ [\ W^4 \mid W]$$

is meaningless since it could never be satisfied for all possible number of W's. While,

$$E \rightarrow I\ [\ W^4 \mid \lambda]$$

means no inhibition at all.

When promoters are made from more than one type of object, **all of them** should exist with sufficient quantities to apply the rule. Also, when inhibitors are made from more than one type of object, **at least one of them** should exist in a sufficient quantity to inhibit the rule. Figure 3.2 shows a diagram of a rule's applicability bounded by promoters $A^{n_1}B^{m_1}$ and inhibitors $A^{n_2}B^{m_2}$ for $n_1 < n_2$ and $m_1 < m_2$. A north-west hatched area is due to the absence of promoters. North-east hatched area is due to the presence of inhibitors.

Moreover, varying behaviour based on different parameters can also be defined by varying the number of promoters and inhibitors. For instance, hatched area in Figure 3.2 is used for applicability of the rule but in its different parameter values.

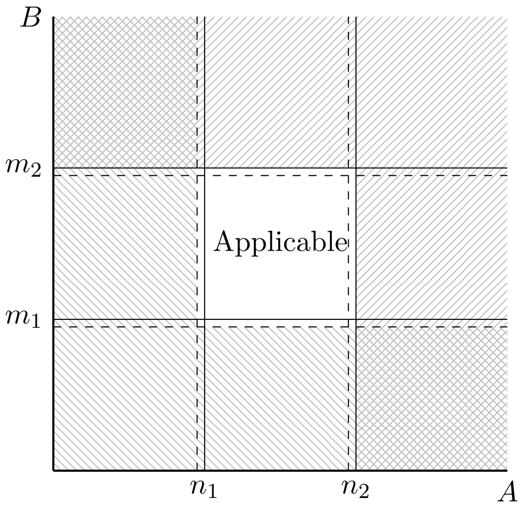


Figure 3.2: Applicability of a rule promoted/inhibited by $[A^{n_1}B^{m_1} \mid A^{n_2}B^{m_2}]$ for $n_1 < n_2$ and $m_1 < m_2$

3.2.3.1 Reaction Rate and Duration

In the previous forms the rules have neither rate nor duration information. In some cases more than one possible behaviour is defined for an object and some of them are more likely than others. In the context of chemical reactions some reactions may have a higher frequency than the others. For this situation we need one parameter to specify these different likelihoods. The frequency with which a reaction occurs is specified by a rate c as follows.

$$\rho : \ \alpha \xrightarrow{c} \beta \ [\psi \mid \chi].$$

These rates will be used to choose among several alternative reactions for a given multiset. Grid Systems make these choices in a stochastic way. This issue of stochasticity will be discussed later in this subsection. Note that, when the rate of a rule is not specified, it is assumed to be 1.0.

The other important use of the rate is in computing reaction duration. Reaction duration is the time interval between consumption of reactants and production of products. In this case the duration of the reaction is implicitly given by $1/c$. This supports the high-level modelling of natural processes whose duration is inversely proportional to the frequency of their termination. For example, in some cases the life expectancy of individuals of a population is inversely proportional to the natural death rate.

When modelling at a lower-level, for instance at an individual level, there may be short duration processes that overlap with longer duration processes and cause their early termination. For example, death due to predation concludes a very fast predation process that causes the early termination of the life of an individual. In this case the duration of its predation process is independent of its frequency. It is therefore necessary to explicitly specify duration d of such processes as follows.

$$\rho : \ \alpha \xrightarrow[d]{c} \beta \ [\psi \mid \chi].$$

Example 3.4. The rules in Example 3.2 are rewritten by specifying its rate along with some rules that create non-determinism.

$$\begin{array}{llcl} ra1: & A & \xrightarrow{0.2} & A\,E^{10} \\ ra2: & A & \xrightarrow[2.0]{0.01} & \lambda \\ ra3: & P\,A & \xrightarrow[1.0]{0.07} & P \end{array}$$

The example shows that an adult has three possible rules: laying eggs ($ra1$) that with the highest rate, natural death ($ra2$) that with the lowest rate, and the death by the predator ($ra3$). The duration of laying eggs is 5 time units, and the duration of natural death is 2 time units. The duration of predation is 2 time units which is the quickest impact on the population.

Note that according to the definition, each committed object (the object that has been taken in an on-going reaction) are staying in committed state until its reaction last. When such object is intended to be able to be affected by other event, more sophisticated behaviour should be defined at modelling level instead. Supposedly, some objects could be interfered during their reaction by other events, each respective rule should be decomposed into some smaller rules representing model's micro-behaviours as illustrated in Example 3.5.

Example 3.5. Referring Example 3.4, object A in *ra2* should behave others than just waiting for its death in 2 units of time. It could be redefined into an iterative rule $r2A$ and actual death rule $ra2B$ as follows.

$$\begin{array}{llcl} ra1: & A & \xrightarrow{0.2} & A\,E^{10} \\ ra2A: & A & \xrightarrow[0.2]{0.005} & A \\ ra2B: & A & \xrightarrow[0.2]{0.005} & \lambda \\ ra3: & P\,A & \xrightarrow[1.0]{0.07} & P \end{array}$$

Instead of being idled during two units of time, by those rules object A could be interfered by other events at every time point after interval length 0.2. Note that, all rule's rates/durations related to reactant A should be adjusted properly due to this change. In above example the rates/durations are given for an illustration only.

3.2.3.2 Maximal Parallelism

The reactions must be applied immediately and maximally whenever the required reactants are available and the conditions related to the promoters and the inhibitors are satisfied. The reactions must also be performed in a parallel (simultaneous) manner. As the reaction are taking place simultaneously, due to their varying durations at any point of time any number of reactions could take place.

Example 3.6. Rules in Example 3.4 will be applied in a possible multiset over them such that no more reactions can be conducted at that moment. One possible combination is: 5 $ra1$'s, 1 $ra2$'s, and 2 $ra3$'s, which consumes all A's and P's. It is also possible for 4 $ra1$'s and 4 $ra2$'s (without any $ra3$) with remaining 2 P's but no more A. However, applying 5 $ra1$'s and 2 $ra3$'s is not maximal yet since there is one remaining A (either $ra1$ or $ra2$ can still consume this).

The Stochasticity Part (see Section 3.2.3.4) will describe how such a combination will be selected.

3.2.3.3 Object States

Because of reaction duration there is a critical time interval between starting and completion of each reaction. For Grid Systems we postulate that each reactant still exists and for other reactions, it can act as a promoter or an inhibitor, but it cannot act as a reactant. When it is necessary to avoid acting as a promoter/inhibitor, this 'reaction' should be applied in stages through defining several reaction rules. Therefore, in Grid systems there are several states in which a reactant still remains until its reaction is accomplished; instead the state of the object changes from "available" to "committed" when the reaction starts.

Grid System evolves through reactions. During the reactions the system obeys three properties that are also typical in nature: parallelism, stochasticity and spatiality.

Example 3.7. Continuing Example 3.4, given that at time t_i there are eight adults and two predators (ie.,A^8 P^2) in one membrane at time $t = 0$. Assume that 2 $ra1$'s, 4 $ra2$'s and 2 $ra3$'s are selected to be applied to the objects. All objects are involved in the reactions. Since $t = 0$, their states are changed from available to committed. At $t = 1.0$ all $ra3$'s are accomplished and then at the same time their committed objects (2 A's and 2 P's) are removed and 2 new available P's are produced. Also, at $t = 2$, all $ra2$'s are accomplished and then their committed objects (two A's) are removed. After $t = 2$ the remaining committed objects (6 A's) remain until $t = 5$ when they are removed and the products (2 A's and 20 E's) are produced as available objects. Just right after 2 A's are produced, they are involved in the next reactions. Let us further assume that two $ra3$'s are applied, then at $t = 6$ only 2 P's remain in the systems (no more A). Figure 3.3 illustrates this situation.

3.2.3.4 Stochasticity

In Grid Systems different rules exist which consume various reactants. Several different rules may require the same reactants at the same time as in Example 3.6. This non-determinism will be resolved stochastically based on the propensity of each rule adopted from Gillespie's SSA [23] (see also Section 2.3.2).

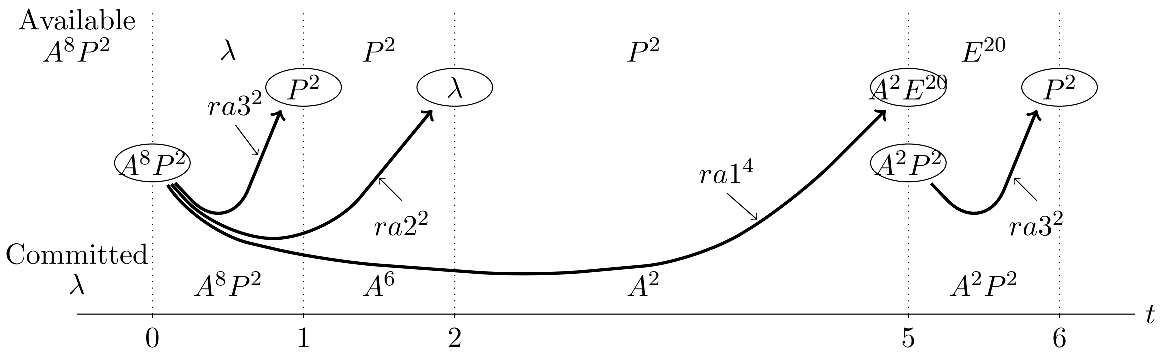


Figure 3.3: Illustration of Example 3.7

Weighted by their propensities a Monte Carlo process is performed to select a rule to be applied and the reactants are allocated. When the maximum has not been reached the propensities are recomputed based on the unallocated objects and the selection is repeated. Note that, this iterative process is performed before applying the selected reactions (their respective objects are marked as allocated, they are not changed to committed yet).

Example 3.8. Making reference to Example 3.7, their propensities initially are

$$\begin{array}{lll} a(ra1) & = 0.2 \times C(8,1) & = 0.2 \times 8 = 1.6 \\ a(ra2) & = 0.01 \times C(8,1) & = 0.01 \times 8 = 0.08 \\ a(ra3) & = 0.07 \times C(8,1) \times C(2,1) & = 0.07 \times 8 \times 2 = 1.12 \end{array}$$

Given that the randomized selection weighted by the propensities results in selecting $ra1$. Then, recomputing propensities will result in

$$a(ra1) = 1.4, \quad a(ra2) = 0.07, \quad \text{and} \quad a(ra3) = 0.98.$$

After reaching its maximality let us say that the resulted combination consists of 5 $ra1$'s, 1 $ra2$'s and 2 $ra4$'s. Then, at t_i the system will evolve according to those reactions (along with other reactions in other membranes).

Stochasticity is also manifested by varying the duration of the reactions. The duration is exponentially distributed with the mean $1/c$ when duration d is not specified. Otherwise, the duration is exponentially distributed with the mean d. To specify a rule which has this stochastic property, mark 'M' is placed after c for the former case,

$$\rho: \ \alpha \xrightarrow{c,M} \beta \ [\psi \mid \chi]$$

or after d for the latter case,

$$\rho: \ \alpha \xrightarrow[d,M]{c} \beta \ [\psi \mid \chi]$$

Example 3.9. Let us redefine the rules in Example 3.4 as follows.

$$\begin{array}{lll} ra1m: & A & \xrightarrow{0.2,M} \ A\ E^{10} \\ ra2m: & A & \xrightarrow[2.0,M]{0.01} \ \lambda \\ ra3m: & P\ A & \xrightarrow[1.0,M]{0.07} \ P \end{array}$$

The durations of the reactions according to these rules will be exponentially distributed and the means are 5 time units, 2 time units, and 1 time unit, respectively. Let us use the initial state of the system as in Example 3.7, i.e., A^8P^2, and the reactions at $t = 0$ are the same: 2 $ra1m$'s, 4 $ra2m$'s and 2 $ra3m$'s. The following Figure illustrates termination times of reactions that are exponentially distributed to their rates (computed by a simulation).

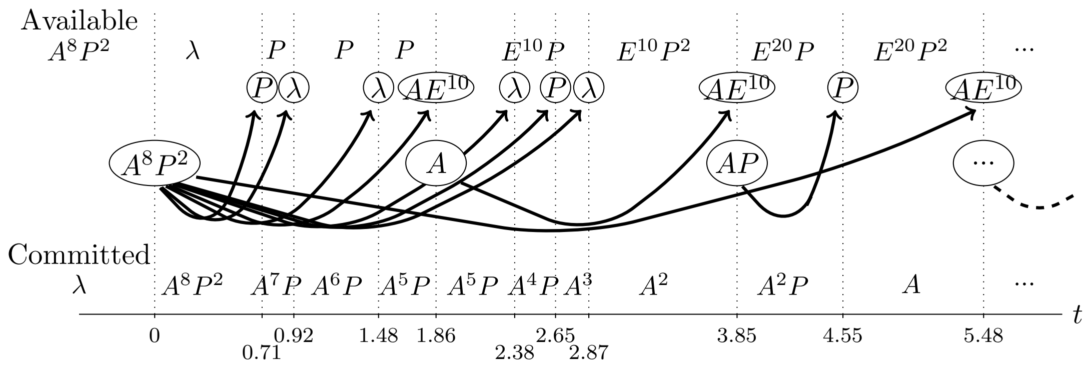


Figure 3.4: Illustration of Example 3.9

The system's state just remains on each interval from the time of an activation of reaction to the next earliest completion time which is shown as the segment between two consecutive vertical dotted lines. Along with all on-going reactions in other membranes, the system could ultimately explode its state space. Namely, when the number of objects is d, and the number of membranes is $N \times M$, the system creates a state space of $\mathbb{N}^{d \times N \times M}$. Furthermore, the existence of on-going reactions makes Grid Systems becoming non-markovian stochastic processes, and therefore, the model of Grid Systems could be mathematically intractable and a simulative analysis could be more suitable in general for observing its dynamics. Specifically, CTMCs are not enough for describing the dynamics of Grid Systems, since transitions with a fixed duration cannot be expressed in a CTMC. It would be necessary to use some variant/extension of CTMC such as, for example, "delayed CTMC" [24]. Also, Grid Systems are a quite high-level language. Hence, the translation into a form of CTMC is not trivial and it is left as a possible future work. However, either Markov Chains or Stochastic Petri Nets can be employed for analysing and refining the behaviours of the model at the individual level, namely, in estimating the rates of the rules consuming the same reactants, before putting them together as one ecosystem's model.

3.2.3.5 Spatiality

Grid Systems deal with many aspects of the spatial dynamics of the objects that are distributed into the membranes. Firstly, the spatial dynamics of objects is enabled by varying their behaviour by means of rule-membrane associations that are formally defined as the table/set of associations A in Definition 3.1. Associations regulate applicability of rules to membranes implying that two different membranes having different sets of associations will cause the objects inside them to behave differently although they are in equal multisets. The set A is a static table, therefore A is intended to represent a permanent aspect of spatiality, such as spatiality due to geographical differences. As an illustration, the difference of land elevations in a wide area ecosystem causes different vegetation types. In turn, the difference in vegetation causes different behaviour of the species that are being modelled.

Also P Systems enable associations between reactions and membranes. The difference between P Systems and Grid Systems is on how associations are expressed: in P Systems the rules are defined in each membrane, with possible duplication of certain rules.

Cellular Automata (CA), in the original definition, do not enable such dynamics since the applicability of rules is the same for all cells. However, Complex Automata (generalisation of CA) enable spatial dynamics through the definition of some levels of aggregation [28, 27].

Secondly, Grid Systems enable spatial dynamics between the membranes. In particular, the presence of promoters and inhibitors in the rules can create behaviours that depend on spatial properties. In Grid Systems the objects acting as promoters or inhibitors can be located in other membranes, and a rule can produce objects to be located in other membranes as well. In order to deal with this form of spatiality, the Grid Systems support two kinds of membrane referencing: absolute addressing and relative addressing. Absolute addressing specifies the referenced membrane's address as the referenced membrane's actual row and column numbers. Absolute addressing is similar to location labelling that is defined in many formalisms, namely Spatial PEPA [21], PALPS [47] and Spatial CLS [45].

Relative addressing specifies the referenced membrane's address as the relative column-row distance from the membrane which the rule is associated with. Such form of referencing enables further spatial dynamics through the concept of locality of membranes as the neighbourhood cells in Cellular Automata. This spatiality is similar, but not the same, to that of Pardini's Spatial P Systems [45]. In Spatial P Systems, Pardini defines a relative referencing among the coordinates inside the membrane while in Grid Systems the referencing is among the membranes.

Note that, in Spatial PEPA [21] such relationships can be defined as hypergraphs among the process components. Each hypergraph needs an explicit declaration while in Grid Systems the neighbourhood relationship is a direct implication of relative addressing. For more general relationships, an advanced referencing method for Grid Systems will be described as the links in Section 3.3.

In order to construct a more consistent (unambiguous) concept of rule applicability, the reactants of a reaction are restricted only to the objects of the membrane where the reaction is applied. The need to use other reactants from several different membranes can be realized at the modelling level by defining several instantaneous (0 time delay) rules to locate or to create objects into those membranes (of the major reactants).

In reactions, objects of different membranes are written with different notations. Objects from the current membrane are written as usual but objects from other membranes are marked by their membrane addresses. In other words, the same objects from different membranes are treated as different objects when they are written as a multiset.

The address of object a in a cell will be expressed as the subscript to a. An absolute address is written within squared brackets "[., .]" as in $a_{[3,4]}$, and a relative address within curved brackets "(., .)", as in $a_{(-1,1)}$. The following example shows the use of relative addressing. Based on our observation in working with some cases, the use of relative addressing is similar to the neighbourhood in CA. In most natural cases relative addressing is more suitable than the absolute one. Based on our exploration in general computation cases the absolute addressing will be more potentially useful in defining inter-membrane connectivity, namely for simulating the mechanisms of other formalisms. Both observations will be exhibited later in the next chapters.

Example 3.10. Let us refer to previous examples, spatiality of Grid Systems enables us to define adult insects which can move around the space. To keep the movement inside the area we need to define logically the outer cells (i.e., the cell whose row/column number is either 0 or 5) as the border by putting one dummy object Z into such a cell. For this example we have $\Sigma = \{E, I, A, P, W, R, T, U, D, Z\}$. The movement is defined to neighbouring cells by the following rules:

$$\begin{array}{llll} rm1: & A & \xrightarrow{2,M} & A_{(-1,0)} \quad [\ \lambda \mid Z_{(-1,0)}\ ] \\ rm2: & A & \xrightarrow{2,M} & A_{(1,0)} \quad [\ \lambda \mid Z_{(1,0)}\ ] \\ rm3: & A & \xrightarrow{2,M} & A_{(0,-1)} \quad [\ \lambda \mid Z_{(0,-1)}\ ] \\ rm4: & A & \xrightarrow{2,M} & A_{(0,1)} \quad [\ \lambda \mid Z_{(0,1)}\ ] \end{array}$$

Objects Z in its destination cells will inhibit any movement into their cells. Therefore when Z is placed in the boundary cell it functions as the blocker. Those rules provide a random-walk like movement of A in the space. Non-determinism caused by those rules, along with rules in Example 3.9 will be resolved as in Example 3.8.

Grid Systems also provide a method to address objects located in the "environment" which is the global membrane. Objects in this location can be expressed by subscripting respective objects with "$[E]$", as in $a_{[E]}$. Of course, as for the local membranes the rules can be associated to global membrane and, therefore, the addressing for the rule should not be used.

Example 3.11. Let us assume that object T exists only in global membrane. A rule that hatches an egg to be an immature mosquito can be defined as:

$$E \xrightarrow{2,M} I[\ T_E^2 \mid T_E^4\ ]$$

can be applied in a local membrane when the number of T's (Temperature level) in the environment membrane is between 2 and 3 inclusively.

Concepts of spatiality by those three addressing methods could be seen as extensions of alphabet Σ. The extensions are defined formally as follows.

Definition 3.2. Let Σ be the alphabet in a Grid System; each object $a \in \Sigma$ located in membrane $G_{i,j}$, where $i, j \geq 0$ can be addressed by a rule associated to any membrane as $a_{[i,j]}$. Moreover, $\Sigma_{[i,j]} = \{a_{[i,j]} | a \in \Sigma\}$ and $\Sigma_{ABS} = \bigcup_{i \geq 0, j \geq 0} \Sigma_{[i,j]}$. This type of object referencing is called absolute addressing to local membranes.

Definition 3.3. Let Σ be the alphabet in a Grid System; each object $a \in \Sigma$ located in membrane G_E can be addressed by a rule associated to any membrane as a_E. Moreover, $\Sigma_E = \{a_E | a \in \Sigma\}$. This type of object referencing is called absolute addressing to the environment membrane.

Definition 3.4. Let Σ be the alphabet in a Grid System; each object $a \in \Sigma$ located in membrane $G_{k,l}$ (where $k, l \geq 0$) can be addressed by a rule associated to membrane $G_{c,d}$ as $a_{(i,j)}$, where $i = k - c$ and $j = l - d$. Moreover, $\Sigma_{(i,j)} = \{a_{(i,j)} | a \in \Sigma\}$ and $\Sigma_{REL} = \bigcup_{i,j \in \mathbb{Z}} \Sigma_{(i,j)}$. This type of object referencing is called relative addressing.

We need to extend the notation in writing multisets by incorporating the spatial information of its objects. In Definition 2.2 a multiset in Σ is expressed either as a sequence of pairs (a, n)'s or a sequence of a^n's. By incorporating the spatial information we need to define its extension as follows.

Definition 3.5. The facts in a multiset over $(\Sigma_{ABS} \cup \Sigma_E \cup \Sigma_{REL})$ can be expressed either as pairs (a_p, n)'s or as triples (a, n, p)'s or as a_p^n's, where $n = M(a_p)$ is the multiplicity of object a in the membrane referred to by p.

Note that, in using either absolute addressing or relative addressing (Definition 3.2 and Definition 3.4), the actual address can syntactically refer to a negative column number or a row number. Semantically, the address is related to an 'empty-but-undefined membrane'. This means that every product which is put into this place will be lost and every object in this place referred to by any rule as a promoter/inhibitor will always be considered absent.

Hence, to generalize all reaction rules discussed intuitively above into a formal definition, transition rules are defined as follows.

Definition 3.6. Let Σ be the alphabet in a Grid System. A rule ρ is a relation of multiset α giving multiset β under conditions given by parameters: ψ, χ, c, d, X, and written as tuple:

$$\rho = (\alpha, \beta, \psi, \chi, c, d, X)$$

where

- ρ is the unique identifier of the rule;
- α is a non-empty multiset of reactants, α is a multiset over Σ and $\alpha \neq \lambda$;
- β is a multiset of products, β is a multiset over $(\Sigma \cup \Sigma_{ABS} \cup \Sigma_E \cup \Sigma_{REL})$;
- ψ is a multiset of promoters, ψ is a multiset over $(\Sigma \cup \Sigma_{ABS} \cup \Sigma_E \cup \Sigma_{REL})$;
- χ is a multiset of inhibitors, χ is a multiset over $(\Sigma \cup \Sigma_{ABS} \cup \Sigma_E \cup \Sigma_{REL})$;
- $c \in \mathbb{R}^+$; c is the rate with which the rule may be applied to perform a reaction;
- $d \in \mathbb{R}^+$; d is the duration of the reaction when it is applied;
- $X \in \{\text{'D'}, \text{'M'}\}$ is a marking to the rule; 'D' indicates that the duration of the reaction will take exactly d time units when it is applied; and 'M' indicates that the duration time is an exponentially distributed random variable that has mean value d time units.

The rule $\rho = (\alpha, \beta, \psi, \chi, c, d, X)$ can be expressed more intuitively using an arrow similar to the notation in P Systems according to several possible forms:

- Putting the parameters c, d, and X, around the arrow and ψ and χ bracketed in the right side, as:

$$\rho : \alpha \xrightarrow[d,X]{c} \beta \; [\psi \mid \chi]$$

- As the previous one, when $\psi = \chi = \lambda$ they could be omitted, as:

$$\rho : \alpha \xrightarrow[d,X]{c} \beta$$

- As the previous ones, when $X =$ 'D' this marking could be omitted, as:

$$\rho : \alpha \xrightarrow[d]{c} \beta \ [\psi \mid \chi]$$

- As the previous ones, when $d = 1/c$ parameter d could be omitted, as:

$$\rho : \alpha \xrightarrow{c,X} \beta \ [\psi \mid \chi]$$

- As the previous ones, when $c = d = 1$ and $X =$ 'D' they could be omitted, as:

$$\rho : \alpha \rightarrow \beta \ [\psi \mid \chi]$$

For further discussion, when $\psi = \chi = \lambda$, we say that the parameters have their default values. Also c, d and X have their default values when $X =$ 'D' and $c = d = 1$. Therefore, when all parameters of a rule have their default values we can write the rule simply as $\alpha \rightarrow \beta$. Note that, when $X =$ 'M', at least c should be written along with X as above.

Furthermore, we will need the following definition for further discussion. The functions are intended in order to acquire rule's components.

Definition 3.7. Given a rule $\rho = (\alpha, \beta, \psi, \chi, c, d, X)$, we define functions

$$\begin{aligned}
\boldsymbol{\alpha}(\rho) &= \alpha \\
\boldsymbol{\beta}(\rho) &= \beta \\
\boldsymbol{\psi}(\rho) &= \psi \\
\boldsymbol{\chi}(\rho) &= \chi \\
\text{rate}(\rho) &= c \\
\text{duration}(\rho) &= d \\
\text{mark}(\rho) &= X
\end{aligned}$$

3.2.4 Association Table

As defined in Definition 3.1 Association Table A defines spatial heterogeneity of cell behaviour around the grid:

$$A = \{(\rho, \gamma) \mid \rho \in R,\ \gamma \in \{G_{i,j} \mid i, j \geq 0\} \cup \{G_E\}\};$$

Each entry of the table defines an applicability relationship of a rule in a membrane explicitly. Therefore, the table defines many-to-many relationship between the membranes and the rules.

For further discussion we need to define a function in order to acquire the rules that are associated to a membrane.

Definition 3.8. assoc(m) is the rules that are associated with the membrane m, or

$$\text{assoc}(m) = \{r \mid (r, \gamma) \in A, \text{ and } \ m = \gamma\}$$

3.2.5 Configurations

A configuration $C^{(t)}$ is the state of the system at time point t. It records the system state in two parts: current objects and current ongoing reactions. The current objects are represented as the multisets of existing objects in each membrane. The ongoing reactions are the reactions that have been applied but due to their durations they have not been accomplished. For its semantic purpose, the list of ongoing reactions is sorted according to reactions' termination time.

Definition 3.9. A Configuration $C^{(t)}$ is a pair

$$(\{C^{(t,m)} \mid m \in \{G_{i,j} \mid i, j \geq 0\} \ \cup \ \{G_E\}\}, \ \Omega^{(t)})$$

where

- $C^{(t,m)}$ is the multiset over objects that exist inside membrane m at time t;
- $\Omega^{(t)} = \{(r_k, t_k, m_k) \mid k = 1, ..., n \ \text{ and } \ t \leq t_1 \leq ... \leq t_n\}$ is the set of ongoing reactions where each (r_k, t_k, m_k), $k = 1, ..., n$, denotes ongoing reactions that instantiate rule r_k in membrane m_k and will terminate at time t_k.

3.2.6 Operational Semantics

The evolution over time of a Grid System starts from an initial configuration $C^{(0)}$. The system evolves passing through a trajectory of configurations $C^{(0)}$, $C^{(t_1)}$, $C^{(t_2)}$, ..., where, $0 \leq t_1 \leq t_2....$

An evolution step at time t_k consists in a maximally parallel application of rules to available objects in all membranes of the system. In order to apply a rule, reactants have to be present among available objects. The application of rules with maximal parallelism means that several rules can be applied (also more than once) to different objects in the same evolution step until no further rule is applicable. Because rules have duration, reactants of the applied rules become committed objects of the configuration, and they remain committed until the completion of the rules. For each applied rule a termination time is computed and a new entry in the list of ongoing reactions $\Omega^{(t_k)}$ is created.

Once all applied rules of the current evolution step have been handled, the set of rules to be completed first, $\{(r_i, t, m_i)$ such that t is minimal in $\Omega^{(t_k)}\}$, is extracted from $\Omega^{(t_k)}$. Rules in this set are the next ongoing reaction to be completed in membrane m, and t is the time of the next evolution step, namely $t_{k+1} = t$. Completion of reactions consists in the removal of its reactants from the committed objects and in the addition of the products to the available objects. Note that if there exist in $\Omega^{(t_k)}$ several ongoing reactions associated with the same time t all of these will be extracted and handled. Subsequently, the procedure is repeated to perform the next evolution step. For further discussion, several definitions will be presented.

A function for computing reaction duration that will be used in the following algorithm is defined here. Its description has been elaborated previously in part "Reaction rate and duration" of Section 3.2.3. The way in which elapsed(r) is computed depends on the parameter d and X of the reaction rule:

Definition 3.10. For rule $\rho(\alpha;\psi,\chi,c,d,X) = \beta$, its function elapsed($\rho$) is the non-negative real number defined as follows.

$$\text{elapsed}(\rho) = \begin{cases} d & \text{for } X = \text{`D'} \\ -d\ \log Y & \text{for } X = \text{`M' and } Y \sim U[0,1](\text{ uniformly distrib. r.v.}) \end{cases}$$

Note that, according to the inverse method the function $-\log Y$ produces exponentially distributed random numbers whose mean value is 1 since the values of Y are uniformly distributed random numbers in unit interval.

Another function for identifying the membrane being addressed in the rule will be defined as well. It description has been elaborated previously in part "Spatiality" of Section 3.2.3.

Definition 3.11. The existence (actual membrane) of objects addressed by p in the context of membrane m in grid system G is defined as:

$$\pi(p,m) = \begin{cases} G_{i+k,j+l} & \text{if } m = G_{i,j}\text{, and } p \text{ is in form of "(k,l)"} \\ G_{k,l} & \text{if } p \text{ is in form of "[k,l]"} \\ G_E & \text{if } p \text{ is in form of "[E]"} \\ m & \textit{otherwise} \end{cases}$$

As described previously, each object which is just taken as a reactant in any reaction will have its state changed from "available" to "committed", and then it stays in a "committed" state before being removed when its reaction is accomplished. Formally, the states of objects are kept by the following sets relating to $C^{(t)}$. $Commit^{(t,m)}$ denotes objects of $C^{(t,m)}$ that are committed to any of the $(r_k,t_k,m_k) \in \Omega^{(t)}$ such that $m = m_k$; and $Avail^{(t,m)} = C^{(t,m)} \setminus Commit^{(t,m)}$.

The following is a logical function which is able to evaluate whether a rule is applicable to a membrane. Given a reaction rule r and a membrane m of a configuration $C^{(t)}$, let applicable(r,m) be a predicate that holds if and only if r is applicable to membrane m. Formally:

Definition 3.12.

$$\begin{aligned} \text{applicable}(r,m) &= \boldsymbol{\alpha}(r) \subseteq Avail^{(t,m)} \\ &\wedge \forall(a,n,p) \in \boldsymbol{\psi}(r).a^n \in (Avail^{(t,\pi(p,m))} \cup Commit^{(t,\pi(p,m))}) \\ &\wedge \forall(a,n,q) \in \boldsymbol{\chi}(r).a^n \notin (Avail^{(t,\pi(q,m))} \cup Commit^{(t,\pi(q,m))}) \end{aligned}$$

Note that both available and committed objects can provide promoters and inhibitors as included in the second and third terms.

3.2.6.1 Evolution Algorithm

Semantics of Grid Systems will be described through an algorithm called Evolution Algorithm of Grid Systems. The algorithm starts from a given initial configuration $C^{(0)}$ in which $\Omega^{(0)}$ is assumed to be an empty list. Given a configuration $C^{(t_k)}$ at time t_k, the evolution step giving $C^{(t_{k+1})}$ as result is obtained by performing the following steps (Note that operators $\cup$ and $\setminus$ denote union and subtraction respectively over multisets as defined in Definition 2.3 and assoc(m) is a function defined by Definition 3.8):

- **Step-1**: For each membrane m, compute a maximal multiset $Cand^{(t_k,m)}$ over assoc(m). Each reaction in $Cand^{(t_k,m)}$, which is called a candidate reaction, must be an instance of an applicable rule, namely $Cand^{(t_k,m)}$ is a multiset over $\{r \mid \text{applicable}(r,m)\}$. Maximality is defined as follows: let $\boldsymbol{\alpha}(Cand^{(t_k,m)}) = \bigcup_{r\in Cand^{(t_k,m)}} \boldsymbol{\alpha}(r)$, it must hold:

$$\forall r \in \text{assoc}(m).\Big(\text{applicable}(r,m) \implies \boldsymbol{\alpha}(r) \not\subseteq \Big(Avail^{(t_k,m)} \setminus \boldsymbol{\alpha}(Cand^{(t_k,m)})\Big)\Big)$$

  Namely, it must be impossible to apply any rule to the objects to which no candidate reaction is applied.

- **Step-2**: For each membrane m, perform each reaction $r \in Cand(m)$ by changing the states of its reactants from available to committed. Multiset $Reacted^{(t_k,m)}$ represents all those reactants. The operations are expressed as follows.

$$\begin{aligned} Reacted^{(t_k,m)} &= \bigcup_{r\in Cand^{(t_k,m)}} \boldsymbol{\alpha}(r)\\ \widehat{Avail}^{(t_k,m)} &= Avail^{(t_k,m)} \setminus Reacted^{(t_k,m)}\\ \widehat{Commit}^{(t_k,m)} &= Commit^{(t_k,m)} \cup Reacted^{(t_k,m)} \end{aligned}$$

- **Step-3**: Tuples $(r, t_k + \text{elapsed}(r), m)$'s of the reactions of each membrane are inserted into $\widehat{\Omega}^{(t_k)}$. The function elapsed(r) which computes duration of r refers to Definition 3.10.

$$\widehat{\Omega}^{(t_k)} = \Omega^{(t_k)} \cup \left\{ \bigcup_{m'\in G} \left\{ \bigcup_{r\in Cand^{(t_k,m')}} (r, t_k + \text{elapsed}(r), m') \right\}\right\}$$

- **Step-4**: Globally, determine t_{k+1} and multiset $First^{(t_{k+1})}$, the time and the earliest reactions in $\widehat{\Omega}^{(t_k)}$ that are to last.

$$\begin{aligned} t_{k+1} &= \min\Big\{t \mid (r_i,t,m_i) \in \widehat{\Omega}^{(t_k)}\Big\}\\ First^{(t_{k+1})} &= \{(r_i,t,m_i) \mid (r_i,t,m_i) \in \widehat{\Omega}^{(t_k)} \text{ and } \forall (r',t',m') \in \widehat{\Omega}^{(t_k)}.t \le t'\} \end{aligned}$$

- **Step-5**: For each membrane m multiset $F^{(t_k,m)}$ over R is constructed based on r_i's where each $(r_i,t,m) \in First^{(t_k)}$. Then, get all products $\boldsymbol{\beta}(F^{(t_k,m')})$ from membrane m' and add to m, and, remove its reactants from the committed objects. Also, remove the corresponding entry from $\widehat{\Omega}^{(t_k)}$. Let $\boldsymbol{\alpha}(F) = \bigcup_{r\in F} \boldsymbol{\alpha}(r)$ and $\boldsymbol{\beta}(F) = \bigcup_{r\in F} \boldsymbol{\beta}(r)$ Those operations are expressed as follows.

$$\begin{aligned} F^{(t_{k+1},m)} &= \{r_i|(r_i,t,m_i) \in First^{(t_{k+1})} \text{ and } m_i = m\}\\ Avail^{(t_{k+1},m)} &= \widehat{Avail}^{(t_k,m)} \cup \Big\{\bigcup_{m'\in G}\Big\{\bigcup_{(a,n,p)\in\boldsymbol{\beta}(F^{(t_{k+1},m')}),\pi(p,m')=m} a^n\Big\}\Big\}\\ Commit^{(t_{k+1},m)} &= \widehat{Commit}^{(t_k,m)} \setminus \boldsymbol{\alpha}(F^{(t_{k+1},m)})\\ \Omega^{(t_{k+1})} &= \widehat{\Omega}^{(t_k)} \setminus First^{(t_{k+1})} \end{aligned}$$

If $\Omega^{(t_{k+1})}$ is empty and for each membrane m no rule is applicable in $C^{(t_{k+1})}$ (namely $\forall m.Cand(m) = \emptyset$) then the evolution terminates with $C^{t_{k+1}}$ as the final configuration. Otherwise, the next iteration will be repeated from **Step-1** with the last t_{k+1} becoming the next t_k.

The diagram in Figure 3.5 shows the relations of the entities in the algorithm to aid in understanding the algorithm. Rectangles are global entities (updated by all membranes). Bold line ellipses are entities in membrane m and dashed line ellipses are entities from other membranes similar to the ones of membrane m. An arrow described the dependability relation between the entities: the entity at the arrowhead requires the entity at the other end in its formulation in the algorithm. The functions/operations of the formulations are not shown to simplify the picture. Set of Associations A and Set of Rules R are not shown to make it more readable, but they are referred to in several places.

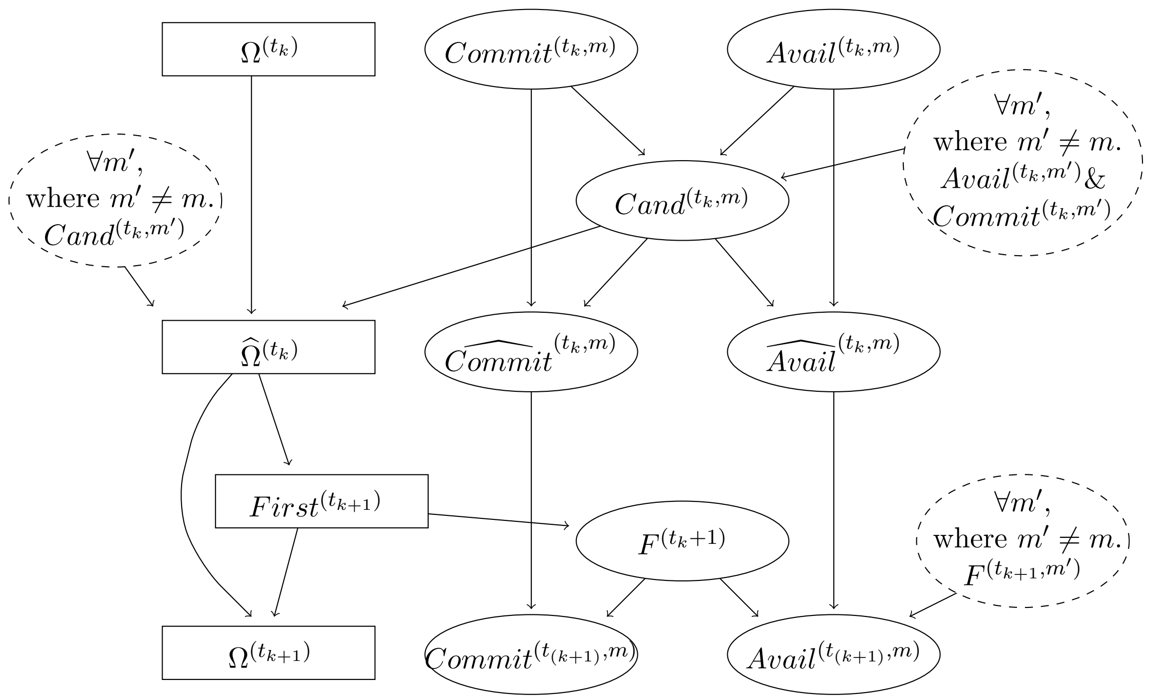

$\Omega^{(t_k)}$
$Commit^{(t_k,m)}$
$Avail^{(t_k,m)}$
$\forall m'$, where $m' \neq m$. $Avail^{(t_k,m')}$& $Commit^{(t_k,m')}$
$\forall m'$, where $m' \neq m$. $Cand^{(t_k,m')}$
$Cand^{(t_k,m)}$
$\widehat{\Omega}^{(t_k)}$
$\widehat{Commit}^{(t_k,m)}$
$\widehat{Avail}^{(t_k,m)}$
$First^{(t_{k+1})}$
$F^{(t_k+1)}$
$\forall m'$, where $m' \neq m$. $F^{(t_{k+1},m')}$
$\Omega^{(t_{k+1})}$
$Commit^{(t_{(k+1)},m)}$
$Avail^{(t_{(k+1)},m)}$

Figure 3.5: Dependability diagram of each entitity in the Evolution Algorithm

Example 3.12 shows how the algorithm runs for Example 3.9 (which is illustrated by Figure 3.4). To shorten it, the example shows only at $t = 0$, 0.71, 1.86, and 3.85 (skipping $t = 0.92$, 1.48, 2.38, 2.65, 2.87, 4.55, and so on).

Example 3.12. The initial state is the multiset of objects A^8P^2 and empty Ω.

$Initialstate : Avail^{(0)} = A^8P^2,\ Commited^{(0)} = \lambda,\ \Omega^{(0)} = \{\}.$

$Cand^{(0,G_{0,0})} = \{(ra1m, 2), (ra2m, 4), (ra3m, 2)\},$
$\hat{Avail}^{(0)} = \lambda,\ \hat{Commited}^{(0)} = A^8P^2,$
$\hat{\Omega}^{(0)} = \{(ra3m, 0.71, G_{0,0}), (ra2m, 0.92, G_{0,0}), (ra2m, 1.48, G_{0,0}), (ra1m, 1.86, G_{0,0}),$
$(ra2m, 2.38, G_{0,0}), (ra3m, 2.65, G_{0,0}), (ra2m, 2.87, G_{0,0}), (ra1m, 5.48, G_{0,0})\},$
$First^{(0.71)} = \{(ra3m, 0.71, G_{0,0})\},\ Avail^{(0.71,G_{0,0})} = P,\ Commit^{(0.71,G_{0,0})} = A^7P,$
$\Omega^{(0.71)} = \{(ra2m, 0.92, G_{0,0}), (ra2m, 1.48, G_{0,0}), (ra1m, 1.86, G_{0,0}), (ra2m, 2.38, G_{0,0}), ...\}.$

$Cand^{(0.71,G_{0,0})} = \{\},$
$\hat{Avail}^{(0.71)} = P,\ \hat{Commited}^{(0.71)} = A^7P,$
$\hat{\Omega}^{(0.71)} = \{(ra2m, 0.92, G_{0,0}), (ra2m, 1.48, G_{0,0}), (ra1m, 1.86, G_{0,0}), (ra2m, 2.38, G_{0,0}), ...\},$
$First^{(0.92)} = \{(ra2m, 0.92, G_{0,0})\},\ Avail^{(0.92,G_{0,0})} = P,\ Commit^{(0.92,G_{0,0})} = A^6P,$
$\Omega^{(0.92)} = \{(ra2m, 1.48, G_{0,0}), (ra1m, 1.86, G_{0,0}), (ra2m, 2.38, G_{0,0}), (ra3m, 2.65, G_{0,0}), ...\}.$

$Cand^{(1.86,G_{0,0})} = \{(ra1m, 1)\},$
$\hat{Avail}^{(1.86)} = E^{10}P,\ \hat{Commited}^{(1.86)} = A^5P,$
$\hat{\Omega}^{(1.86)} = \{(ra2m, 2.38, G_{0,0}), (ra3m, 2.65, G_{0,0}), (ra2m, 2.87, G_{0,0}), (ra1m, 3.85, G_{0,0}), ...\},$
$First^{(2.38)} = \{((ra2m, 2.38, G_{0,0})\},\ Avail^{(2.38,G_{0,0})} = E^{10}P,\ Commit^{(2.38,G_{0,0})} = A^4P,$
$\Omega^{(2.38)} = \{(ra3m, 2.65, G_{0,0}), (ra2m, 2.87, G_{0,0}), (ra1m, 3.85, G_{0,0}), (ra1m, 5.48, G_{0,0})\}.$
...
...
$Cand^{(3.85,G_{0,0})} = \{(ra3m, 1)\},$
$\hat{Avail}^{(3.85)} = E^{20}P,\ \hat{Commited}^{(3.85)} = A^2P,$
$\hat{\Omega}^{(3.85)} = \{(ra3m, 4.55, G_{0,0}), (ra1m, 5.48, G_{0,0})\},$
$First^{(4.55)} = \{((ra3m, 4.55, G_{0,0})\},\ Avail^{(4.55,G_{0,0})} = E^{20}P^2,\ Commit^{(4.55,G_{0,0})} = A,$
$\Omega^{(4.55)} = \{(ra1m, 5.48, G_{0,0})\}.$
...

3.3 Grid Systems with Links

Nature has given living species the ability to sense and to follow pathways. For example, wood ants can memorize snapshot views and landmarks [19], salmon fish can sense geomagnetic fields [35], and sperm cells can sense chemotaxes to locate the ovum [33]. The pathways are either given by nature (as for salmon and sperm), created by themselves dynamically (as for ants), or a combination of both. Therefore, the model of pathways might be more complex than just random walks (Brownian motion) as is the case for chemical particles.

In order to resolve ecological problems the ability to model such movements could help in understanding related behaviours, such as the behaviour of a seasonal-migrating species. Migration is a survival strategy that some species use to preserve their population [44]. Sometimes some government policies may affect migration areas. The ability to model such behaviours and analyse the model will allow us to evaluate the impact of a given policy on the population size.

By using Grid Systems as the modelling formalism, such movements, in terms of their pathways, could be expressed through putting additional objects around the pathways to inhibit or promote the present direction to the expected direction. However, such modelling work would be very tedious and complicated. This is because such directed movement may need to be expressed as a large number of rules and parameters. Instead, an extended definition of objects, termed 'links', is introduced here which enables the pathways to be modelled as either land marking or collective memorisation.

Intuitively, a link is defined as an 'object' that carries pointers. A pointer is information which provides a dynamic addressing of a destination membrane. The pointers can be used by rules in referring to the objects in another cell. This is the third addressing method in addition to relative addressing and absolute addressing. Being used as an addressing method, different pointers carried by a link, introduce another non-determinism into the system. In order to resolve this further a non-determinism decision will be made stochastically based on the weighting of each pointer. Weightings are real numbers between 0 and 1.0, and the total weighting in the same cell is 1.0. Like ordinary objects, the number of links in a cell can be increased or decreased by applying its related rule.

3.3.1 Formal Definition of Links

We will first define pointers and then links. Basically, pointers are the addresses of other membranes carried by the links. Following spatial addressing methods, as discussed in the Spatiality part of Section 3.2.3, pointers are of both types as well: relative and absolute pointers.

Definition 3.13. A pointer is an ordered pair of integers. There are two types of pointers: relative pointers and absolute pointers. For the relative pointer the pair of integers is marked by curved brackets, as "(a, b)", where $a, b \in \mathbb{Z}$. For the absolute pointer the pair of integers is marked by squared parentheses, as "$[r, c]$", where $r, c \in \mathbb{N}$.

The links are definitely objects in Grid Systems. The objects are called links when they carry at least one pointer. When they are carrying several pointers, the pointers are weighted to represent the model's preference for any particular pointer.

Definition 3.14. G is a Grid System extended with links when any object in Σ can carry pointers. A link is an object that is carrying at least one pointer. The properties of the links are reflected in the following aspects of the system. Let $a, p \in \Sigma$.

- When a link p exists in membrane m (as the current object in the multiset $C^{(t,m)}$), an attribute $L(m, p)$ is defined in membrane m, where $L(m, p) = \{(\eta_i, w_i) \mid \eta_i$ is a pointer belonging to p, $w_i \in \mathbb{R}^{\geq 0}$, η_i can be either absolute or relative, and $0.0 \leq w_i \leq 1.0\}$ and $\Sigma_i w_i = 1.0$.

- Object p is considered to be a link in a reaction rule (as a reactant, or a product, or a promoter, or an inhibitor), p is attributed to either pointer η (literally), written as $p : \eta$, or to any pointer carried by existing pointer p' in the membrane where the rule is being applied, written as $p : p'$.

- The pointer of a link p can be used in a reaction rule to refer to the membrane in which object a is located. This fact is written as a_p.

The following definition defines equivalence between the relative and absolute links.

Definition 3.15. When object p is the link existing in cell G_{r_0,c_0}, pointer $[r_1, c_1]$ carried by p points to the cell G_{r_1,c_1}, and pointer (dr, dc) carried by p points to the cell G_{r_0+dr,c_0+dc}. Furthermore, regarding link p in G_{r_0,c_0}, pointer $[r_1, c_1]$ equals pointer (dr, dc) if and only if $r_0 + dr = r_1$, and, $c_0 + dc = c_1$. Even though they are equivalent, the pointers in L are listed in the same way as in which they were created.

Example 3.13. Let G be a grid system extended with links and $\Sigma = \{A, B, P\}$. At one time in membrane $G_{3,3}$ there are 10 A's and 5 P's. P is a link and it has absolute pointers [2,4], [4, 4], and [4, 3], and they are weighted 0.3, 0.2, 0.5 respectively. The current state of $G_{3,3}$ is the multiset $A^{10}\ P^5$, where P is denoted by $L(G_{3,3}, P) = \{([2, 4].0.3), ([4, 4], 0.2), ([4, 3], 0.5)\}$.

Given that two reaction rules in G are defined as

$$\begin{array}{lcl} r1 : A & \rightarrow & A_{(-1,-1)}\ P^5 : (-1, -1)\ [\ B \mid \lambda\ ] \\ r2 : B & \rightarrow & B_P\ [\ A_P \mid A\ ] \end{array}$$

Let the current membrane be $G_{3,3}$ and $r1$ and $r2$ are associated to $G_{3,3}$. Rule r_1 says that if B exists in $G_{3,3}$, an object A will be translated from $G_{3,3}$ to $G_{2,2}$, and 5 links P pointing to $[2, 2]$ (relatively) will also be added to $L(G_{3,3}, P)$.This addition will change the number of pointers in $L(G_{3,3}, P)$ as well as their weighting. Rule $r2$ says that object B will be translated from $G_{3,3}$ to any membrane pointed by the pointers in $L(G_{3,3}, P)$, as long as there is no A in $G_{3,3}$ but at least one A exists in the destined membrane.

For further discussion, let us name addressing objects by a link as **type-3** addressing, and the absolute addressing and relative addressing in Section 3.2.3 (Definition 3.2 and Definition 3.4) as **type-1** addressing and **type-2** addressing, respectively.

3.3.2 Operational Semantics of Links

In order to be extended with links the semantics of Grid Systems require three additional mechanisms:

1. to be able to replicate the rules containing objects which have Type-3 addressing;
2. to be able to evaluate the rule's applicability of the rules containing links as reactants or promoters or inhibitors; and
3. to be able to update the weights when the number of links is changed (some links are added/removed from the current membrane).

3.3.2.1 Replication of the Rule

The use of a type-3 addressing in a rule requires rule replication. This should be included in the steps of Evolution Algorithm that is described in 3.2.6. Rule replication is an instantiation of that rule by replacing the link (as a type-3 addressing) by its pointer. When a link of the rule has more than one pointers, the rule is instantiated in as many as pointers the link has. Furthermore, one rule may have several different links used as type-3 addressing. Replications for such a rule are based on the combinations of the pointers of each link.

Definition 3.16. Let r be the rule associated to membrane m containing Type-3 addressing by d different links, i.e., $p_1, p_2, ..., p_d$. Given that in membrane m, the links are attributed by $L(m, p_i) = \{(\eta_{i,j}, w_{i,j}) \mid i = 1, .., d, \text{ and } j = 1, ..., k_d\}$ for each $p_i \in \{p_1, p_2, ..., p_d\}$ and the size of $L(m, p_i)$ are $k_1, k_2, ..., k_d$, respectively. For all $j_e \in \{1..k_e\}$ and $e \in \{1.., d\}$,

- All instances of replication of r are generated by substituting each link $p_1, p_2, ..., p_d$ by every possible combinations of $\eta_{1,j_1}, \eta_{2,j_2}, ..., \eta_{d,j_d}$.

- $r|_{(\eta_{1,j_1}, \eta_{2,j_2}, ..., \eta_{d,j_d})}$ denotes one instance of replication of r, by a combination of pointers $\eta_{1,j_1}, \eta_{2,j_2}, ..., \eta_{d,j_d}$ whose weights are $w_{1,j_1}.w_{2,j_2}...w_{d,j_d}$, respectively.

Semantically, the Evolution Algorithm needs refinements to consider Type-3 addressing. When a rule with Type-3 addressing is being evaluated by **Step-1** of the algorithm for membrane m, its applicability is assumed when **at least one** instance $r|_{(\eta_1, \eta_2, ..., \eta_d)}$ exists that can hold applicable$(r|_{(\eta_1, \eta_2, ..., \eta_d)}, m)$. Furthermore, in **Step-3** one applicable instance is selected randomly based on their weights at that time. Pointers $r|_{(\eta_1, \eta_2, ..., \eta_d)}$ of selected instance is then attributed to r in tuple as it is inserted as $(r|_{(\eta_1, \eta_2, ..., \eta_d)}, t_k + \text{elapsed}(r), m)$ to $\hat{\Omega}^{(t_k)}$.

Its duration still takes the original duration defined for r.

Example 3.14. Let G from Example 3.13 have two other links, Q and R and attributes by.

$$\begin{aligned} L(G_{5,5}, Q) &= \{([6,4].0.3), ([5,4], 0.4), ([4,2], 0.3)\} \\ L(G_{5,5}, R) &= \{([4,4], 0.4), ([6,6], 0.6)\} \end{aligned}$$

The following rule is associated to $G_{5,5}$.

$$r3 : B \quad \xrightarrow{10} \quad B_Q \;[\; A_Q \mid A \; B^3_R \;]$$

Replication of $r3$ has instances as follows:

$$\begin{aligned}
r3|_{([6,4],[4,4])} :\ B &\rightarrow B_{[6,4]} \;[\; A_{[6,4]} \mid A \; B^3_{[4,4]} \;] \quad //\text{rate} = 1.2 \\
r3|_{([6,4],[6,6])} :\ B &\rightarrow B_{[6,4]} \;[\; A_{[6,4]} \mid A \; B^3_{[6,6]} \;] \quad //\text{rate} = 1.8 \\
r3|_{([5,4],[4,4])} :\ B &\rightarrow B_{[5,4]} \;[\; A_{[5,4]} \mid A \; B^3_{[4,4]} \;] \quad //\text{rate} = 1.6 \\
r3|_{([5,4],[6,6])} :\ B &\rightarrow B_{[5,4]} \;[\; A_{[5,4]} \mid A \; B^3_{[6,6]} \;] \quad //\text{rate} = 2.4 \\
r3|_{([4,2],[4,4])} :\ B &\rightarrow B_{[4,2]} \;[\; A_{[4,2]} \mid A \; B^3_{[4,4]} \;] \quad //\text{rate} = 1.2 \\
r3|_{([4,2],[6,6])} :\ B &\rightarrow B_{[4,2]} \;[\; A_{[4,2]} \mid A \; B^3_{[6,6]} \;] \quad //\text{rate} = 1.8
\end{aligned}$$

Note that, these instances are not persistent since in other time points the pointers of each link as well as the number of links themselves may be different.

3.3.2.2 Links as Objects in the Rules

Principally, when some links appear as reactants or promoters or inhibitors in the rule, the rule's applicability is determined by considering the proportion of each link's quantity by its weight of respective pointers. Given that p^n (link p having quantity n) exists in the membrane m and a rule r will be applied to m; when link p appears in r without a pointer, as in p^s, where $s \leq n$, it will be handled as an ordinary object. On the other hand, when it appears in a rule with a pointer η_i, as in $p^s\!:\!\eta_i$, where $(\eta_i, w_i) \in L(m,p)$, it will be handled according to its role in the rule as follows.

- As reactants, the rule can be applied when $s \leq nw_i$ and the changes will follow in Section 3.3.2.3.
- As products, the rule will be applied and the number of link p will change accordingly and the changes will follow in Section 3.3.2.3.
- As promoters, the rule can be applied when $s \geq nw_i$.
- As inhibitors, the rule can be applied when $s < nw_i$.

The applicability function defined as Definition 3.12 remains the same. The extension is made to the definition of the operations of multisets. The definitions have to consider links with pointers as different objects with multiplications are proportional by respective weights and their total is the multiplication of the link itself.

Example 3.15. Let us consider

$$r4 : P^5\!:\!(1,1) \rightarrow P^4\!:\!(1,1)[P^{10}\!:\!(1,1)|P^{40}\!:\!(1,1)]$$

Given that r is associated with $G_{3,3}$ and $L(G_{3,3}, P) = \{(G_{4,4}, 0.3), ...\}$. This rule is applicable only when $3 \leq n < 12$, where n is the quantity of P.

3.3.2.3 Updating the Links

Step-2 and **Step-5** of the Evolution Algorithm in 3.2.6 shows the changes that happen to $Avail^{(t_k,m)}$ and $Commit^{(t_k,m)}$ (or from available state to committed state). **Step-2** only moves from *Avail* to *Commit*, therefore, it does not update the weights. However, **Step-5** performs the actual changes, then it should be followed by updating their weighting. It is important to note that any addition to a link in a membrane should be performed before its removal as written in the algorithm. This sequence is intended to preserve the weighting of being totally removed. Moreover, applicability of a rule whose objects addressed by links should also consider conditions in relation to respective pointers.

- **Adding Links**

  If in membrane m, link p with quantity n, has k distinct pointers, namely, $L(m,p) = \{(\eta_1, w_1), (\eta_2, w_2), ..., (\eta_k, w_k)\}$. Then,

  – adding p^s (without a pointer) will not affect the current weighting as happens to ordinary objects.

– adding $p^s : \eta_i$ will affect its overall weighting:

$$w'_j = \begin{cases} \dfrac{nw_j}{n+s} & \text{for } \eta_i \neq \eta_j \\ \dfrac{nw_j + s}{n+s} & \text{for } \eta_i = \eta_j \end{cases}$$

Additionally, when $\eta_i \notin \{\eta_1, \eta_2, ..., \eta_k\}$, new entry (η_i, w'_i) is added to $L(m,p)$ where

$$w'_i = \begin{cases} \dfrac{s}{n+s} & \text{for } k > 0 \\ 1 & \text{for } k = 0 \end{cases}$$

- **Removing Links**

  If in membrane m, link p with quantity n, has k distinct pointers, namely, $L(m,p) = \{(\eta_1, w_1), (\eta_2, w_2), ..., (\eta_k, w_k)\}$. Then,

  – removing p^s (without a pointer) will not affect the weighting, as they behave as ordinary objects, except in the case of $s = n$, all pointers will be removed.

  – removing $p^s : \eta_i$, from that cell, where $s < nw_i$, changes its overall weighting:

$$w'_j = \begin{cases} \dfrac{nw_j}{n-s} & \text{for } \eta_i \neq \eta_j \text{ and } n \neq s \\ \dfrac{(nw_j - s)}{(n-s)} & \text{for } \eta_i = \eta_j \text{ and } n \neq s \end{cases}$$

  Additionally, when $s = nw_i$, entry (η_i, w'_i) is removed from $L(p)$.

Example 3.16. Let us consider a current situation in membrane $G_{5,5}$. P is a link that is present in the membrane with quantity 10. Its current weighting is: $\{([3,4], 0.3), ([4,4], 0.6), ([5,2], 0.1)\}$.

- Adding $P^8 : [5,2]$ into the membrane will change all weighting to be

$$\{([3,4], 0.17), ([4,4], 0.33), ([5,2], 0.5)\}$$

- After that, adding $P^2 : [3,5]$ will add a new entry to it and change all weighting to be

$$\{([3,4], 0.15), ([4,4], 0.3), ([5,2], 0.45), ([3,5], 0.1)\}$$

- After that, removing $P^5 : [4,4]$ will make all weighting to be

$$\{([3,4], 0.2), ([4,4], 0.07), ([5,2], 0.6), ([3,5], 0.13)\}$$

From the example, adding link objects will amplify its weighting of the related pointer but de-amplify the others. Similarly, removing link objects will reduce its weighting of the related pointer but emphasize the others.

Note that, all usages of a link can appear at once in a rule. However, such a practice is not recommended since the rule can be too cryptic to understand its meaning, unless there is a reason for this (no other option).

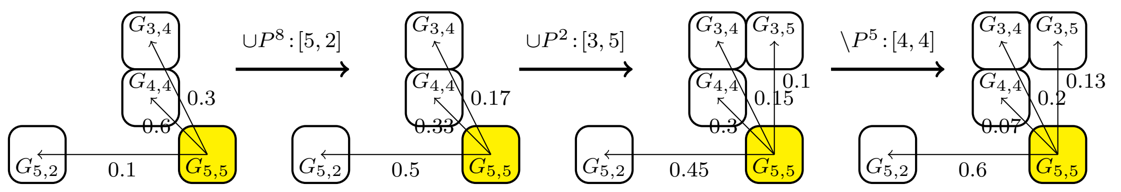


Figure 3.6: Illustration of Example 3.16

Example 3.17. Given that P is a link and A^2P^{10} already exists in membrane m, and $L(m, P) = \{((1, 1), 0.4), ((-1, 1), 0.6)\}$. Reaction rule:

$$A^2 \ \rightarrow \ A \ P_P\!:\!P$$

is syntactically valid. When it is applied it can actually be as either $A^2 \ \rightarrow \ A \ P_{(1,1)}\!:\!(1, 1)$ or $A^2 \ \rightarrow \ A \ P_{(-1,1)}\!:\!(-1, 1)$ where the latter is less weighted than the former. It means that every pair of A's in membrane m will be reduced to become one A and put P into another membrane which is pointed by P itself and the produced P will carry the currently used pointer.

3.4 Extension of Notations

Syntactically all notations of Grid Systems have already been defined in the previous sections. However, using those notations to define one example of complex biological behaviour may require a huge number of rules that are similar yet with some small differences, namely the differences of some rates caused by different temperatures. As a consequence, the model gets less readable due to the huge number of such rules. Some additional notations will be added on top of the syntax. The following notations are the ones that we have used in our case studies but it is not limited to these.

3.4.1 Regions

A group of local membranes may be identified as *a region*. Basically, they do not need to be contiguous. The main use of regions is to define associations. It is also useful in declaring initial objects that occupy each membrane of a certain region. Associations between some membranes and a rule can be simplified by an association between a region (containing the membranes) and the rule. Formally, the region is a set of membranes. It is defined that for region R, membrane m and rule r:

$$\text{if } m \in R, \text{ and } r \in Assoc(R), \text{ then } r \in Assoc(m).$$

A rectangular region, simply called a *rect*, is a special region that can be specified by two corner membranes G_{i_1,j_1} and G_{i_2,j_2}, and $j_1 \leq j_2$, as $rect(i_1, j_1, i_2, j_2)$, where $i_1 \leq i_2$, so that

$$G_{i,j} \in rect(i_1, j_1, i_2, j_2) \text{ if and only if } i_1 \leq i \leq i_2 \text{ and } j_1 \leq j \leq j_2.$$

Furthermore, a region can be defined as an operation between some other regions (through union, intersection, ...) to simplify the definition of complicated shaped regions.

3.4.2 Constants

A constant is an identifier that can be associated with a value (either number or string). The parts in a rule can be written in terms of constants or even in simple arithmetic formulae composed of the constants. Then, its actual value can be derived by computing the formulae.

The constants are used to maintain the model. The model could be more manageable since the change to the values can be done without touching the reaction rules. The constants may be further used to define Rule Templates that will be described in the next part.

3.4.3 Rule Templates

Rules with similar structures may be identified as a *single template* containing one or more variables $\langle X \rangle$, each representing a finite set of values which is called Range of Values. Actual rules can be obtained by substituting each $\langle X \rangle$ in the body of the template with its possible values in the Range of Values. Along with the use of constants, the substitution $\langle X \rangle$ will be performed before substituting the constants with the constant values.

Example 3.18. Let us define a following template:

$$\begin{aligned}&types = \{red, blue\}\\&ranges = \{1,2\}\\&\forall\langle A\rangle \in types, \forall\langle R\rangle \in ranges.\ \mathbf{R}\langle\mathbf{A}\rangle\langle\mathbf{R}\rangle : KT\langle A\rangle \xrightarrow{rate\langle R\rangle} M^{num\langle R\rangle}\end{aligned}$$

It represents rules as follows.

$$\begin{array}{lll}Rred1: & K\ Tred & \xrightarrow{rate1} & M^{num1}\\ Rred2: & K\ Tred & \xrightarrow{rate2} & M^{num2}\\ Rblue1: & K\ Tblue & \xrightarrow{rate1} & M^{num1}\\ Rblue1: & K\ Tblue & \xrightarrow{rate2} & M^{num2}\end{array}$$

Note that $K, M, Tred, Tblue$ are objects and $rate1, rate2, num1, num2$ are constants.

3.4.4 External Events

External events that are given at certain time points may play important roles in an ecosystem. In Grid Systems, $Events = \{(X,t) \mid \text{ where } X = x_1, x_2, ...x_n \text{ and } t = \tau_1, \tau_2, ...\tau_n\}$ (or events $x_1, x_2, ...$ that will be given respectively at time $t = \tau_1, \tau_2, ...$), can be expressed by the following:

1. A template representing the timer whose reaction is activated at $t = 0$, and will produce object $x_1, x_2, ...$ at $t = \tau_1, \tau_2, ...$,

$$\forall\langle X\rangle \in \{x_1, x_2, ...x_n\}, \forall\langle t\rangle \in \{\tau_1, \tau_2, ...\tau_n\}.\ \mathbf{Ev}\langle\mathbf{X}\rangle\langle\mathbf{t}\rangle : Ev\langle X\rangle\langle t\rangle \xrightarrow{1/\langle t\rangle} \langle X\rangle$$

   and

2. Rules that will affect each kind of $x_1, x_2, ...$ to the actual objects targeted by the event. Note that, when there is k different events of $x_1, x_2, ...x_n$ where $1 \leq k \leq n$, the number of such rules is $\leq k$.

Example 3.19. Two external events that will raise the temperature by one level at $t = 15$ and will also lower by one level at $t = 21$ in the global membrane can originally be expressed by the following rules:

$$\begin{array}{lll} re1: & Ev15 & \xrightarrow{1/15} & T \\ re2: & Ev21 & \xrightarrow{1/21} & L \\ Tch: & T\ L & \xrightarrow{\infty} & \lambda \end{array}$$

By defining them as *Events* $re1$ and $re2$, these rules can be reduced as follows.

$$\begin{array}{ll} Events & = \{(15, T), (21, L)\} \\ Tch: & T\ L \xrightarrow{\infty} \lambda \end{array}$$

Note that, to activate the timers all respective $Ev\langle X\rangle\langle t\rangle$'s (or $Ev15$ and $Ev21$ in the former) should be initialized in the system. Thus, the timers will be applied right away at $t = 0$. At time $t = 15$ the object T will be produced so that the number of T is increased and at time $t = 21$ the object L will be produced so that it triggers a reaction of Tch to decrease the number of T.

The timer rules and event objects are used only once in the system. Since the formalism has nothing to do with space complexity, at this conceptual level they will be ignored. Instead, they should be considered for performance optimization at an implementation level of analytical tools.

Chapter 4

Relationship to Other Formalisms

The aim of this chapter is to discuss the relationship of Grid Systems to other formalisms. However, this will be done by means of translations of models from other formalisms to Grid Systems. A complete theoretical investigation of these relationships is not the main aim of this thesis.

Grid Systems are a new formalism that is intended to model the population dynamics within an ecosystem. In order to achieve a more comprehensive formalism its features were identified in terms of its modelling needs through case studies. In order to anticipate further needs that have not yet been identified, Grid Systems were also evaluated to express the models which have been developed, based on existing formalisms.

As previously described, Grid Systems combine the features of Cellular Automata (CA) in order to express spatial dynamics and the features of P Systems, to define object-level dynamics, and also the links as the extension of Grid Systems. As a result, Grid Systems can be seen as generalized Cellular Automata, in which the state of each cell is defined by a multiset over objects, and the transition rules are able to partially change this state. On the other hand, Grid Systems can also be seen as P Systems whose membranes are arranged in a two dimensional grid and the reactions can spatially translate objects from one membrane to others.

In this chapter we will report our work to assure that Grid Systems have a backward compatibility in respect to the features of some formalisms. The first part will report how Grid Systems can emulate Turing Machines. The second part will report how some famous models of Cellular Automata can be easily expressed as Grid Systems. The last part will describe our algorithm to convert P Systems into Grid Systems.

4.1 Grid Systems and Turing Machines

The ability of Grid Systems to emulate the mechanism of Turing Machines will be presented by showing an algorithm that converts Turing Machines into Grid Systems. The idea of the algorithm is straightforward: each transition rule is converted into one reaction rule in its Grid System version, and each tape square is represented by a membrane. Thus, the grid will be one row of membranes. The number of membranes in that row will take the maximum tape length that is required to solve the problem of the Turing Machine. Formally the definition of Turing Machines will be expressed in the following part according to the reference [36].

4.1.1 Turing Machines

Definition 4.1. A Turing machine (TM) is 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where

- Q is a finite set of states, assumed not to contain h, the halt state.
- Σ and Γ are finite sets, the input and tape alphabets, respectively, with $\Sigma \subseteq \Gamma$; Γ is assumed not to contain Δ, the blank symbol.
- q_0 the initial state, is an element of Q.
- $\delta : Q \times (\Gamma \cup \{\Delta\}) \rightarrow (Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{\text{R, L, S}\}$ is a partial function (that is, possibly undefined at certain points).

The TM has a tape with a left end but infinite to the right, marked off into squares. Each square contains a symbol of $\Gamma \cup \{\Delta\}$; however, at each point of a TM's operation all but a finite number of these squares will contain the symbol Δ. Initially, the only non-blank symbols on the tape are elements of Σ. The TM also has a *tape head*, which allows it to read from and write to individual squares of the tape. If $q \in Q, r \in Q \cup \{h\}, X, Y \in \Sigma \cup \{\Delta\}$, and $D \in \{\text{R}, \text{L}, \text{S}\}$, the formula

$$\delta(q, X) = (r, Y, D)$$

means that when the TM is in state q and the symbol on the current tape square is X, the machine replaces X by Y on that square, changes its state to r, and either moves the tape head one square left, moves it one square right, or leaves it stationary, depending on whether D is L, R, or S, respectively. If in this situation $D =$ L, but the tape head is scanning the leftmost square, the tape head is not allowed to move. This is one of the situations in which the TM is said to *crash*. If this does not happen, and $r = h$, we say that the move causes the TM to *halt*. Once it has halted, the machine cannot move anymore because δ is not defined at any pair (h, X).

4.1.2 Conversion Algorithm

The parallelism in its Grid System should be "turned off" to emulate the sequential property of Turing Machines. This is performed by using activator objects $q\langle i\rangle$ which also keep the current state of the emulated Turing Machine; there is exactly one $q\langle i\rangle$ located in a membrane representing the "head's position" and current state $q_{\langle i\rangle}$.

In Turing Machines the symbols of the input string are placed on squares from position 1, 2, ... Thus, in Grid Systems the symbols are placed respectively in membrane $G_{0,1}$, $G_{0,2}$, ..., as the part of the initial configuration. Furthermore, at the beginning the head of T is placed in the leftmost square of the tape, whereas in Grid Systems, object $q0$ is placed initially in membrane $G_{0,0}$.

TM T accepts a string input s when T finally enters state h. Thus, G will produce object *halt* and then stop since there is no longer an applicable rule. On the other hand, when TM T cannot accept s, G will stop without producing object *halt*. TM T will crash when the head is moving to the left of the leftmost position. When this happens to G the reaction will produce the $q\langle i\rangle$ outside of the grid and according to the semantics of Grid Systems, the object will just disappear and G will stop since there is no applicable rule (no membrane having $q\langle i\rangle$).

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a TM. A Grid System $G = (\Sigma_G, R, A, C^{(0)})$ can emulate T when G is constructed as follows:

- The set of objects Σ_G consists of all tape symbols Γ, and objects representing each state in T, therefore $\Sigma_G = \Gamma \cup \{halt\} \cup \{q\langle i\rangle | q_{\langle i\rangle} \in Q\}$; object *halt* for halt state h, and objects $q\langle i\rangle$ for each $q_{\langle i\rangle} \in Q$.
- The set of reaction rules R is constructed according to functions δ in T. Given that $Y \in \Gamma \cup \{\Delta\}$, $q \in Q$, and $r \in Q \cup \{h\}$. For each function $\delta(q, X) = (r, Y, D)$ in T, its related reaction rule in G is defined as follows.
  - For the case $X \in \Gamma$,
  $$X\ q\ \rightarrow\ Y\ r_\eta$$
  - For the case $X = \Delta$,
  $$q\ \rightarrow\ Y\ r_\eta\ \ [\lambda|\text{Multiset}(\Gamma)]$$

  where Multiset(Γ) is a multiset where $\forall a \in \Gamma, M(a) = 1$, and
  $$r_\eta = \begin{cases} r_{(0,1)} & \text{if}\quad D = \text{R} \\ r_{(0,-1)} & \text{if}\quad D = \text{L} \\ r & \text{if}\quad D = \text{S} \end{cases}$$
- Association table $A = \{(r, G_{0,j}) | r \in R,\ \text{and } j \geq 0\}$.
- Initial configuration $C^{(0)} = (\{C^{(0,G_{0,k})}\ \mid\ k \geq 0\}, \emptyset)$, where
$$\begin{array}{lcl} C^{(0,G_{0,0})} & = & q0 \\ C^{(0,G_{0,\langle i\rangle})} & = & x\langle i\rangle \text{ for } \langle i\rangle = 1, ..., n \text{ where } x\langle i\rangle \in V \text{ and} \\ & & x\langle 1\rangle x\langle 2\rangle ... x\langle n\rangle \text{ is the input string whose length is } n \\ C^{(0,G_{0,j})} & = & \lambda, \text{ for } j = n+1, ... \end{array}$$

Example 4.1. Table 4.1 is the transition function δ of a TM that accepts language $L = \{ss|s \in \{a, b\}^*\}$ [36]. This language is recursive since it cannot be accepted by any machines lesser than a Turing Machine.

The machine T will accept input string *abab* as the transition shown in Figure 4.1. "(q_i, s')" representing a configuration at certain process steps and q_i is the state of the machine and s' is the current string in the tape. The position of the head is indicated by placing underscore below the symbol (which is the symbol in the tape square where the head is located). The symbol '$\vdash$' indicating a single step transition between the configuration and '$\vdash^*$' indicating multiple step transition reaching the next configuration.

For input string *abaa*, the machine will proceed as $(q_0, \underline{\Delta}abaa) \vdash^* (q_7, \Delta AB\Delta\underline{A})$, meaning that it crashes (terminates not in the halt state h, and so it does not accept *abaa*).

By the conversion algorithm the objects in its Grid System are $\{a, b, A, B, q0, ..., q9, halt\}$ and the reaction rules are as shown in Figure 4.2. Running the simulation of the Grid System over input string *abab* resulted in the same sequence as shown in the Table 4.2. Table 4.3 shows another running for input string *abaa*. For this string the machine does not produce object *halt*, which is consistent with the fact that *abaa* is not a member of $\{ss|s \in \{a.b\}^*\}$.

Table 4.1: Transition rules for the TM accepting $L = \{ss | s \in \{a, b\}^*\}$ (blank entries mean undefined rules)

	a	b	A	B	Δ
q_0					(q_1, Δ, R)
q_1	(q_2, A, R)	(q_2, B, R)	(q_5, A, L)	(q_5, B, L)	(h, Δ, S)
q_2	(q_2, a, R)	(q_2, b, R)	(q_3, A, L)	(q_3, B, L)	(q_3, Δ, L)
q_3	(q_4, A, L)	(q_4, B, L)			
q_4	(q_4, a, L)	(q_4, b, L)	(q_1, A, R)	(q_1, B, R)	
q_5			(q_5, a, L)	(q_5, b, L)	(q_6, Δ, R)
q_6	(q_8, A, R)	(q_7, B, R)			(h, Δ, S)
q_7	(q_7, a, R)	(q_7, b, R)		(q_9, Δ, R)	(q_7, Δ, R)
q_8	(q_8, a, R)	(q_8, b, R)	(q_9, Δ, R)		(q_8, Δ, R)
q_9	(q_9, a, L)	(q_9, b, L)	(q_6, A, R)	(q_6, B, R)	(q_9, Δ, L)

$$\begin{array}{llll}
(q_0, \underline{\Delta}abab) & \vdash (q_1, \Delta\underline{a}bab) & \vdash (q_2, \Delta A\underline{b}ab) & \vdash^* (q_2, \Delta Abab\underline{\Delta}) \\
 & \vdash (q_3, \Delta Aba\underline{b}) & \vdash (q_4, \Delta Ab\underline{a}B) & \vdash^* (q_4, \Delta\underline{A}baB) \\
 & \vdash (q_1, \Delta A\underline{b}aB) & \vdash (q_2, \Delta AB\underline{a}B) & \vdash (q_2, \Delta ABa\underline{B}) \\
 & \vdash (q_3, \Delta AB\underline{a}B) & \vdash (q_4, \Delta A\underline{B}AB) & \vdash (q_1, \Delta AB\underline{A}B) \\
 & \vdash (q_5, \Delta A\underline{B}AB) & \vdash (q_5, \Delta\underline{A}bAB) & \vdash (q_5, \underline{\Delta}abAB) \\
 & \vdash (q_6, \Delta\underline{a}bAB) & \vdash (q_8, \Delta A\underline{b}AB) & \vdash (q_8, \Delta Ab\underline{A}B) \\
 & \vdash (q_9, \Delta A\underline{b}\Delta B) & \vdash (q_9, \Delta\underline{A}b\Delta B) & \vdash (q_6, \Delta A\underline{b}\Delta B) \\
 & \vdash (q_7, \Delta AB\underline{\Delta}B) & \vdash (q_7, \Delta AB\Delta\underline{B}) & \vdash (q_9, \Delta AB\underline{\Delta}) \\
 & \vdash (q_9, \Delta A\underline{B}) & \vdash (q_6, \Delta AB\underline{\Delta}) & \vdash (h, \Delta AB\underline{\Delta}) \text{ (accept!)}
\end{array}$$

Figure 4.1: Sequence of configurations during processing input *abab*.

4.2 Grid Systems and Cellular Automata

In Chapter 2, the definition and some famous models of Cellular Automata have been briefly described. Compared to Grid Systems each cell of CA is treated as an individual entity that can be in any possible state at any time. The state of a cell in CA is represented by a number (from a finite set of numbers). On the other hand, Grid Systems treat a cell as a place of multiset over objects as the state of that cell. Some objects can be consumed as reactants for reaction rules producing other objects. It means that the state of its cell can be partially changed.

The other significant difference is that a rule in Grid Systems can only consume the objects in the cell with which the rule is associated. On the contrary, in CA function f can also take some of the states of the neighbouring cells for its computation. In general, when a CA model is defined as a Grid System it requires the preceding rules to 'know' the status of neighbouring cells so that the computation can be performed based on that.

There are so many possibilities of how f is defined, so that its conversion into Grid Systems may vary (there is no deterministic conversion rule). Furthermore, when the function in CA involves real numbers, the range value of the function should be discretised so that the precision of the function will be degraded. However, since the final value of the function in CA will be mapped back to a finite set of states, the approach could be

$$
\begin{array}{lrlcl}
\mathbf{tq0}: & & q0 & \rightarrow & q1_{(0,1)} \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq1A}: & A & q1 & \rightarrow & A\ \ q5_{(0,-1)} \\
\mathbf{tq1B}: & B & q1 & \rightarrow & B\ \ q5_{(0,-1)} \\
\mathbf{tq1a}: & a & q1 & \rightarrow & A\ \ q2_{(0,1)} \\
\mathbf{tq1b}: & b & q1 & \rightarrow & B\ \ q2_{(0,1)} \\
\mathbf{tq1\Delta}: & & q1 & \rightarrow & halt \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq2A}: & A & q2 & \rightarrow & A\ \ q3_{(0,-1)} \\
\mathbf{tq2B}: & B & q2 & \rightarrow & B\ \ q3_{(0,-1)} \\
\mathbf{tq2a}: & a & q2 & \rightarrow & a\ \ q2_{(0,1)} \\
\mathbf{tq2b}: & b & q2 & \rightarrow & b\ \ q2_{(0,1)} \\
\mathbf{tq2\Delta}: & & q2 & \rightarrow & q3_{(0,-1)} \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq3a}: & a & q3 & \rightarrow & A\ \ q4_{(0,-1)} \\
\mathbf{tq3b}: & b & q3 & \rightarrow & B\ \ q4_{(0,-1)} \\
\mathbf{tq4A}: & A & q4 & \rightarrow & A\ \ q1_{(0,1)} \\
\mathbf{tq4B}: & B & q4 & \rightarrow & B\ \ q1_{(0,1)} \\
\mathbf{tq4a}: & a & q4 & \rightarrow & a\ \ q4_{(0,-1)} \\
\mathbf{tq4b}: & b & q4 & \rightarrow & b\ \ q4_{(0,-1)} \\
\mathbf{tq5A}: & A & q5 & \rightarrow & a\ \ q5_{(0,-1)} \\
\mathbf{tq5B}: & B & q5 & \rightarrow & b\ \ q5_{(0,-1)} \\
\mathbf{tq5\Delta}: & & q5 & \rightarrow & q6_{(0,1)} \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq6a}: & a & q6 & \rightarrow & A\ \ q8_{(0,1)} \\
\mathbf{tq6b}: & b & q6 & \rightarrow & B\ \ q7_{(0,1)} \\
\mathbf{tq6\Delta}: & & q6 & \rightarrow & halt \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq7B}: & B & q7 & \rightarrow & q9_{(0,-1)} \\
\mathbf{tq7a}: & a & q7 & \rightarrow & a\ \ q7_{(0,1)} \\
\mathbf{tq7b}: & b & q7 & \rightarrow & b\ \ q7_{(0,1)} \\
\mathbf{tq7\Delta}: & & q7 & \rightarrow & q7_{(0,1)} \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq8A}: & A & q8 & \rightarrow & q9_{(0,-1)} \\
\mathbf{tq8a}: & a & q8 & \rightarrow & a\ \ q8_{(0,1)} \\
\mathbf{tq8b}: & b & q8 & \rightarrow & b\ \ q8_{(0,1)} \\
\mathbf{tq8\Delta}: & & q8 & \rightarrow & q8_{(0,1)} \;\; [\ \lambda \mid A\ B\ a\ b\ ] \\
\mathbf{tq9A}: & A & q9 & \rightarrow & A\ \ q6_{(0,1)} \\
\mathbf{tq9B}: & B & q9 & \rightarrow & B\ \ q6_{(0,1)} \\
\mathbf{tq9a}: & a & q9 & \rightarrow & a\ \ q9_{(0,-1)} \\
\mathbf{tq9b}: & b & q9 & \rightarrow & b\ \ q9_{(0,-1)} \\
\mathbf{tq9\Delta}: & & q9 & \rightarrow & q9_{(0,-1)} \;\; [\ \lambda \mid A\ B\ a\ b\ ]
\end{array}
$$

Figure 4.2: Reaction rules created from the rules in Table 4.1

Table 4.2: Running the simulation for input *abab* (Note: each row shows the membranes and their contained objects after each iteration)

Iter. no.	$G_{0,0}$	$G_{0,1}$	$G_{0,2}$	$G_{0,3}$	$G_{0,4}$	$G_{0,5}$
0	$q0$	a	b	a	b	
1		$q1\ a$	b	a	b	
2		A	$q2\ b$	a	b	
3		A	b	$q2\ a$	b	
4		A	b	a	$q2\ b$	
5		A	b	a	b	$q2$
6		A	b	a	$q3\ b$	
7		A	b	$q4\ a$	B	
8		A	$q4\ b$	a	B	
9		$q4\ A$	b	a	B	
10		A	$q1\ b$	a	B	
11		A	B	$q2\ a$	B	
12		A	B	a	$q2\ B$	
13		A	B	$q3\ a$	B	
14		A	$q4\ B$	A	B	
15		A	B	$q1\ A$	B	
16		A	$q5\ B$	A	B	
17		$q5\ A$	b	A	B	
18	$q5$	a	b	A	B	
19		$q6\ a$	b	A	B	
20		A	$q8\ b$	A	B	
21		A	b	$q8\ A$	B	
22		A	$q9\ b$		B	
23		$q9\ A$	b		B	
24		A	$q6\ b$		B	
25		A	B	$q7$	B	
26		A	B		$q7\ B$	
27		A	B	$q9$		
28		A	$q9\ B$			
29		A	B	$q6$		
30		A	B	*halt*		

Table 4.3: Running the simulation for input *abaa*. (Note: The membranes at some iterations are not shown.)

Iter. no.	$G_{0,0}$	$G_{0,1}$	$G_{0,2}$	$G_{0,3}$	$G_{0,4}$	$G_{0,5}$
0	$q0$	a	b	a	a	
...	...	...	...	...	...	
24		A	$q6\ b$		A	
25		A	B	$q7$	A	
26		A	B		$q7\ A$	

made quite similar.

Its definition could have as many combinations of the states in the neighbourhood as in every Wolfram's CA. For this case the combination of the states can be emulated by setting its inhibitors/promoters. Its definition could be based on the aggregate of a state value as in Conway's Game of Life. For this case an additional step should be made to aggregate these states.

In the following examples the results from our work in reformulating the famous CA will be presented: Rule 110 (1-D CA) and Conway's Game of Life (2-D CA).

4.2.1 One dimensional CA (Wolfram's Rule-110)

As described before, Wolfram extensively studied the behaviour of 1-D CA for some different transition functions. He concentrated on dual state CA, having [-1,0,+1] neighbourhood. Therefore, the CA require a set of eight transition functions $C'_i = f(C_{i-1}, C_i, C_{i+1}), C_i \in \{0,1\}$. All possibilities made by 8 transition functions are 256 different sets and each set is numbered from 0 to 255 according to its mapping code. One of them is the famous Rule-110 whose transition functions are as follows.

$$\begin{array}{llcll}
f(0,0,0) & = 0 & \quad & f(1,0,0) & = 0 \\
f(0,0,1) & = 1 & & f(1,0,1) & = 1 \\
f(0,1,0) & = 1 & & f(1,1,0) & = 1 \\
f(0,1,1) & = 1 & & f(1,1,1) & = 0
\end{array}$$

The initial configuration in the CA is the initial values, either 0 or 1 in each cells. In order to show the changes during the evolution, the values at each iteration are shown altogether in a grid line-by-line from the earliest (the initial values) in the top-most line to the latest in the bottom-most line, as shown in Figure 2.1.

Reaction rules for a Grid System behaving as Rule 110 can be constructed straightforwardly as follows. Each rule will be defined by taking the state of the central cell as a reactant, the state at the next step to be a product, and its neighbours' states as the promoters. To represent the binary values, two objects $\{V0, V1\}$ are needed respectively to represent state 0 and state 1. Thus, each function $f(C_{i-1}, C_i, C_{i+1}) = C'_i$ in Rule-110 CA, one rule $VC_i \rightarrow VC'_i$ [C_{i-1} C_{i+1} | λ] of its Grid System is created. Totally, this results in 8 reaction rules:

$$\begin{array}{rlcll}
r1: & V0 & \rightarrow & V0 & [V0_{(0,-1)}\ V0_{(0,1)} \mid \lambda] \\
r2: & V0 & \rightarrow & V1 & [V0_{(0,-1)}\ V1_{(0,1)} \mid \lambda] \\
r3: & V1 & \rightarrow & V1 & [V0_{(0,-1)}\ V0_{(0,1)} \mid \lambda] \\
r4: & V1 & \rightarrow & V1 & [V0_{(0,-1)}\ V1_{(0,1)} \mid \lambda] \\
r5: & V0 & \rightarrow & V0 & [V1_{(0,-1)}\ V0_{(0,1)} \mid \lambda] \\
r6: & V0 & \rightarrow & V1 & [V1_{(0,-1)}\ V1_{(0,1)} \mid \lambda] \\
r7: & V1 & \rightarrow & V1 & [V1_{(0,-1)}\ V0_{(0,1)} \mid \lambda] \\
r8: & V1 & \rightarrow & V0 & [V1_{(0,-1)}\ V1_{(0,1)} \mid \lambda]
\end{array}$$

Due to their similarities the list of rules can be shortened into just two rules.

- When a cell previously is 0 having its right neighbour 1, it will change to 1 $(r2, r6)$,

$$V0 \ \rightarrow \ V1 \ \ [V1_{(0,1)} \mid \lambda]$$

- When a cell previously is 1 having both neighbours 1, it will change to 0 ($r8$),

$$V1 \;\rightarrow\; V0 \;\; [V1_{(0,-1)}\; V1_{(0,1)} \mid \lambda]$$

The rules should be associated to each cell. Note that, the other rules ($r1, r3, r4, r5, r7$) do not change its state, so they can be omitted.

Also, in visualising the changes, this one row of cells is shown as a grid by producing the next states into the cells in the next line. This requires a modification of the rules: to put the output in the next line, and adding their respective complement actions to copy the values to the next line. All rules need to change the object $V\langle x\rangle$ to $W\langle x\rangle$ to avoid being reacted again. They are as follows.

$$\begin{array}{lcll} V0 & \rightarrow & W0\, V1_{(1,0)} & [V1_{(0,1)} \mid \lambda] \\ V0 & \rightarrow & W0\, V0_{(1,0)} & [\lambda \mid V1_{(0,1)}] \\ V1 & \rightarrow & W1\, V0_{(1,0)} & [V1_{(0,-1)}\; V1_{(0,1)} \mid \lambda] \\ V1 & \rightarrow & W1\, V1_{(1,0)} & [\lambda \mid V1_{(0,-1)}\; V1_{(0,1)}] \end{array}$$

Therefore, the initial state is placed in the top-most row (by placing $V0$ representing state 0 and $V1$ representing state 1 and the other cells are set empty (λ). As the case in the original CA, the conditions in the most left and most right cells can be handled either

- by adding boundary cells at both ends containing $V0$ and the rules are not applicable to the additional cells; or

- by adding additional rules for wrapping-around the cells (the rules in one end cell consider the cell in the other end as the neighbour of that cell).

4.2.2 Two-Dimensional CA (Conway's Game of Life)

Being inspired by John von Neumann's self-replication machine, John Horton Conway invented a solitaire game and it was named Game of Life (GOL) in 1970.

The game uses an orthogonal grid of squared cells. Each cell can be in one of two states: living or dead. The neighbour of a cell C is the set of surrounding cells in a 3x3 sub-grid (Moore's neighbourhood). A configuration consists of states of overall cells at one point in time. A configuration will evolve to another configuration after a discrete unit of time by following these rules:

- Any living cell with fewer than two living neighbours will die, as if caused by under-population.

- Any living cell with two or three living neighbours will live on to the next generation.

- Any living cell with more than three living neighbours will die, as if caused by overcrowding.

- Any dead cell with exactly three living neighbours will become a living cell, as if by reproduction.

Its functions can be expressed as follows.

$$f^{(t+1)}(i,j) = \begin{cases} 1 & \text{for } (f^{(t)}(i,j) = 1 \text{ and } (2 \geq n^{(t)}(i,j) \leq 3)) \text{ or} \\ & \qquad (f^{(t)}(i,j) = 0 \text{ and } (n^{(t)}(i,j) = 3)) \\ 0 & \text{otherwise} \end{cases}$$

Where $n(i,j) = f(i-1,j-1) + f(i-1,j) + f(i-1,j+1) + f(i,j-1) + f(i,j+1)$
$+ f(i+1,j-1) + f(i+1,j) + f(i+1,j+1)$,
which is the counting function of the living cells of Moore's neighbourhood.

Since it was published by M. Gardner in 1970 [22], GOL has become so famous as to be a field of mathematical research as well as ecological modelling.

In this part, GOL's implementation as a Grid System is presented. As previously described, the implementation needs a preceding step by each cell to inform its neighbouring cells about its state. In this case, it is a step for aggregating the surrounding living neighbourhood (function $n(i,j)$). An object H will represent a living cell and emptiness means a dead cell. This counting of the population will be performed by a reaction on H. This involves "informing" each of its Moore's neighbouring cells by producing object n for each of them. Object H should be changed to object h in order to proceed to the next stage.

$$H \to n_{(-1,-1)}\ n_{(-1,0)}\ n_{(-1,1)}\ n_{(0,-1)}\ n_{(0,1)}\ n_{(1,-1)}\ n_{(1,0)}\ n_{(1,1)}\ h \quad [\,\lambda \mid n\ \,]$$

In the next stage, each cell has 0 or more object n's indicating the number of living cells in its neighbourhood. Based on this quantity the following rules will be applied.

- For each living cell, the cell becomes dead when $n < 2$,
$$n \to \lambda\ [\,h \mid n^2\,]$$
- For each living cell, the cell becomes dead when $n > 3$,
$$n \to \lambda\ [\,n^4\ h \mid \lambda\,]$$
- For each living cell, the cell will still be alive when $n = 2$,
$$n^2 \to H \quad [\,n^2\ h \mid n^3\,]$$
- For each living cell, the cell will still be alive when $n = 3$,
$$n^3 \to H \quad [\,n^3\ h \mid n^4\,]$$
- For each dead cell, the cell will still be dead when $n < 3$,
$$n \to \lambda\ [\,\lambda \mid h\ n^3\,]$$
- For each dead cell, the cell becomes a living cell when $n = 3$,
$$n^3 \to H \quad [\,n^3 \mid h\ n^4\,]$$
- For each dead cell, the cell will remain dead when $n > 3$,
$$n \to \lambda\ [\,n^4 \mid h\,]$$
- At the same time all h should be removed.
$$h \to \lambda$$

The initial state of GOL in Grid Systems will be the same as it is in the CA except 1 will be represented by an object H and 0 will be represented by λ.

4.3 Grid Systems and P Systems

In Section 2.2, the definition of P Systems which are proposed by Georghe Păun [49], is briefly presented. As cited in that section, P Systems have been extended to become several variants of P Systems. In this part our study on backward compatibilities between Grid Systems and original variant of P Systems is reported. The compatibility will be discussed by remodelling the examples of P Systems to be in Grid Systems.

A P System arranges its membranes in a nested structure. The structure is defined by 'parent-child' relationships (in the references it is also called 'container-content' relationships) among the membranes in such a way that a container membrane can have several content membranes. The objects in a membrane can traverse from one membrane to another through these relationships. Moreover, an existing membrane in the structure can be destroyed and the objects inside, including its child membranes, will be moved to its parent membrane. On the contrary, there is no way to create a new membrane into the structure. Besides this, the rules can be prioritized higher than the others. By this setting, the lower priority rules can only be applied when the higher priority rules are not applicable.

Recall that each membrane in a Grid System is related to other membranes forming a structure of a grid. Such a structure enables the membranes to be addressed directly by other membrane so that the objects can be moved directly from one membrane to the others. This difference between structures of P Systems and Grid Systems is analogous to tree data structures and two dimensional arrays in common programming languages.

Firstly we will discuss how the structure of P Systems can be imitated by the logical structure of membranes in Grid Systems by using links. Secondly, we will discuss mechanisms to simulate the process of destroying membranes in P Systems by manipulating those links. Then, we will discuss how promoters and inhibitors in Grid Systems can be set to affect the priority arrangement in P Systems. And finally, we will summarize all these procedures in a formal conversion algorithm.

4.3.1 Imitating the Hierarchical Structures

The structure of P Systems can be imitated by using links in Grid Systems which connect the membranes. Let us construct a Grid System $G = (\Sigma, R, A, C^0)$ imitating the structure of a P System $\Pi = (V, \mu, w_1, ..., w_n, (R_1, \rho_1), ..., (R_n, \rho_n), i_0)$. According to its definition in Definition 2.4 V is the set of object symbols, and μ describes the tree structure of its membranes. The number of membrane is n and each membrane is labelled by j, where $j = 1, .., n$; let us refer to each membrane labelled by j as M_j. Each $(i, j) \in \mu$, where $\mu \subseteq \mathbb{N} \times \mathbb{N}$, denotes that M_j is contained directly in M_i.

The number of local membranes in G that will be used is also n. The global membrane of G will be used for accommodating some supporting mechanisms. Note that when the structure in Π is static (without a rule producing δ), its structure can be simply imitated by connecting the membranes in G by using absolute addressing. However, in our discussion we consider the possibility of the rules producing δ in order to provide a general conversion algorithm. For this reason, we will use links for the relationships.

Let us use membranes addressed at $G_{0,i}$ to represent M_i in Π (by ignoring $G_{0,0}$) so that both indices correspond directly. Each relation to its parent in Π is represented by link *out* and each relation to a child membrane $M_{\langle j \rangle}$ is represented by link $in\langle j \rangle$, where

$out, in2, in3, ...$ are not in V. M_1 is the root membrane, therefore we do not need $in1$ in G. Therefore, rules transporting objects to a membrane's parent/children in Π can be converted to G in a straightforward manner.

Figure 4.3 shows an example of P Systems containing 4 membranes. Its structure is imitated by a Grid System shown logically in Figure 4.3. Note that it is called logical since its real structure is a grid.

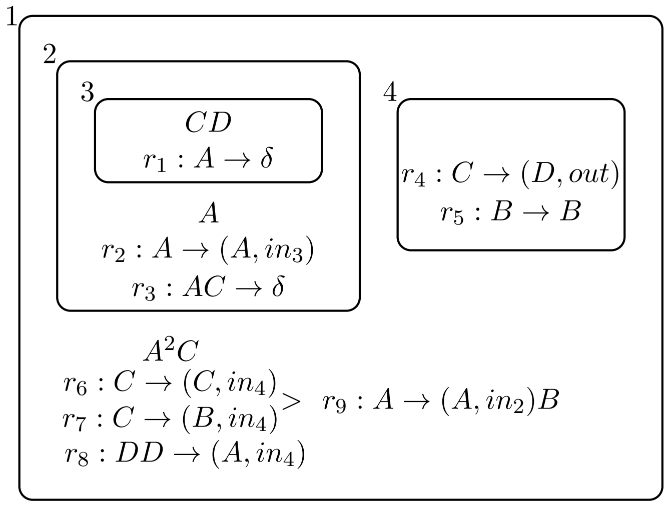


Figure 4.3: An example of a P System to be translated into a Grid System

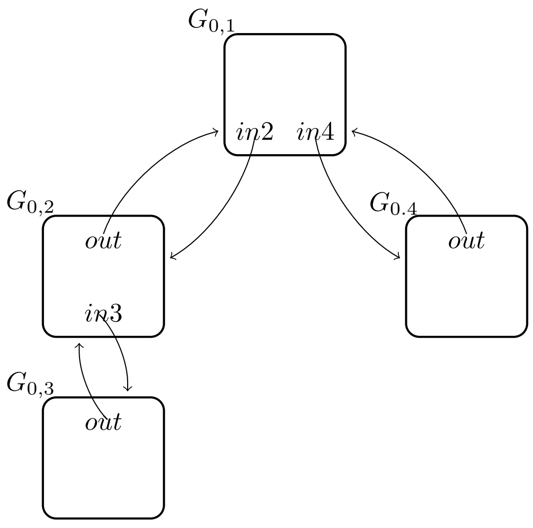


Figure 4.4: The logical structure of a Grid System imitating Figure 4.3

4.3.2 Dissolved Membranes

Unless it is the root membrane, a membrane in a P System can be dissolved (destroyed) by producing a special object δ inside it. Its objects and its child membranes are then moved (re-owned) to its parent membrane. A Grid System can simulate this by rearranging the pointers of related links. Intuitively, the steps in the arrangement for dissolving one membrane are as follows. Let $G_{0,\langle i\rangle}$ be the membrane which is the counterpart of being destroyed $M_{\langle i\rangle}$ in Π.

A1: move the objects in $G_{0,\langle i\rangle}$ to its parent, except δ.

A2: remove the link $in\langle j\rangle$ in the parent membrane of $G_{0,\langle i\rangle}$.

A3: remove the link *out* in each child membrane of $G_{0,\langle i\rangle}$.

A4: replace the pointer carried by *out* in every child membrane of $G_{0,\langle i\rangle}$ by the pointer carried out in $G_{0,\langle i\rangle}$

A5: copy each link $in\langle j\rangle$ in $G_{0,\langle i\rangle}$ including its pointer to the parent membrane of $G_{0,\langle i\rangle}$.

A6: (when necessary) clear the remaining contents (links and δ) in $G_{0,\langle i\rangle}$.

We will refer to these steps later as of A1, A2, ..., of Algorithm-A. The first three steps (A1, A2, A3) can be performed in parallel and A4 and A5 can be performed in parallel too, but after the first three. Step-6 will not affect its logical structure and it can be omitted.

Example 4.2. Figure 4.5 illustrates the steps when a δ is produced in $G_{0,2}$. Besides δ, there are some other objects represented by A^2B. The picture shows the structures before and after rearrangement respectively (except step-6).

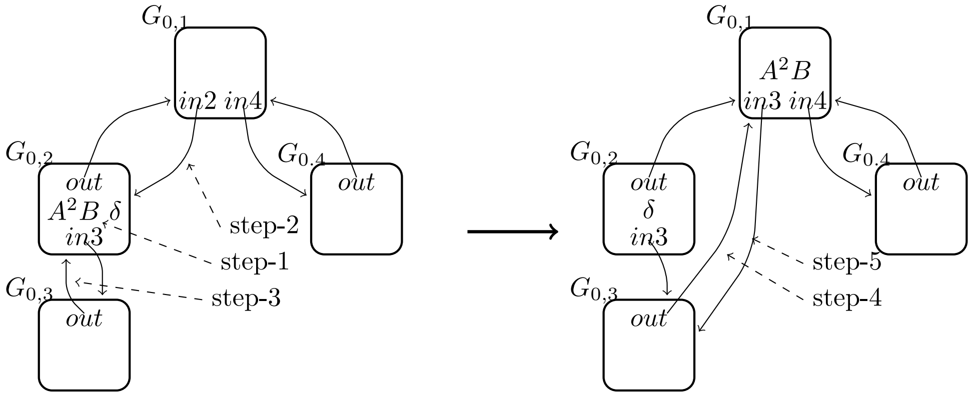


Figure 4.5: The steps of “dissolving” the membranes $G_{0,2}$ in Example 4.2

Algorithm-A can dissolve several membranes at the same time if the membranes are separated by another membrane in the structure. Otherwise, we need to modify it to cover dissolving sequential membranes. Let us name it Algorithm-B whose steps are as follows:

B1: iteratively, Performing A1 because after moving its objects a being dissolved membrane may still receive a moved object from its child which is being dissolved, too.

B2: Performing A2 only in a parent membrane of a being dissolved membrane that is not part of the sequence.

B3: Performing A3 only in the child membrane of a being dissolved membrane that is not part of the sequence.

B4: iteratively, performing A4 and A5 only in a being dissolved membrane whose objects (except δ) have been moved, to each child membrane which is not in the sequence.

Furthermore, in order to avoid being interfered with by ordinary reactions the rearrangement reactions above should be exclusively performed from those ordinary reactions. The exclusion will be performed by issuing a new object σ in global membrane every time δ is being issued. Therefore, all ordinary reaction rules should be defined to be inhibited by σ.

In dissolving a sequence of membranes the number of steps is variable. In order to provide a proper exclusion time interval, each produced σ has a lifetime of one step transition and each step in the algorithm above should provide another σ to refresh the exclusion. Therefore, after the last step of dissolving a sequence σ is no longer present and the ordinary reactions can take place.

In order to provide similar timing with Π all rearrangement reactions have zero duration time of the system. Recall that in Grid Systems all zero duration reactions will be performed step-wisely and the system timer stays.

Let σ be the blocking object and $\sigma \notin (V \cup \{E\langle i\rangle\} \cup \{F\langle i\rangle\})$. If r_i produces δ in Π, r'_i should also produce σ in the global membrane besides δ, in G. All reaction r's as well as additional reactions for handling priorities (in the previous part), should be redefined to include a σ as their inhibitor as described previously.

We will express Algorithm-B as reaction rules of Grid Systems. Let $G_{0,\langle M\rangle}$ be any membrane to be 'dissolved' whose parent is $G_{0,k}$ and child membranes are $G_{0,j1}$, $G_{0,j2}, \ldots$. Moving the objects in $G_{0,\langle M\rangle}$ to $G_{0,k}$ (B1) will be performed by the rules as follows.

$$\forall X \in V.\ \mathbf{mv\langle X\rangle} :\ \langle X\rangle \underset{0}{\rightarrow} X_{out}\ \ [\ \delta\ \sigma_E \mid \lambda\ ]$$

Regarding B2 and B3, link $in\langle M\rangle$ pointed from $G_{0,k}$ to $G_{0,\langle M\rangle}$ and every link out pointed from $G_{0,j1}$, $G_{0,j2}, \ldots$ to $G_{0,\langle M\rangle}$ will be removed by the rules as follows.

$$\begin{aligned}&\forall\langle M\rangle \in mbrn.\ \mathbf{din\langle M\rangle} :\ in\langle M\rangle \underset{0}{\rightarrow} \sigma_E\ \ [\ \delta_{in\langle M\rangle}\ \sigma_E\ \mid\ \delta\ ]\\ &\mathbf{dout} :\ out \underset{0}{\rightarrow} \sigma_E\ \ [\ \delta_{out}\ \sigma_E\ \mid\ \delta\ ]\end{aligned}$$

B4 will be performed by the following rules.

$$\begin{aligned}&\forall\langle M\rangle \in mbrn.\ \mathbf{cp\langle M\rangle} :\\ &\qquad in\langle M\rangle \underset{0}{\rightarrow} in\langle M\rangle_{out}{:}in\langle M\rangle\ \ out_{in\langle M\rangle}{:}out\ \ \sigma_E\\ &\qquad\qquad [\ \delta\ \sigma_E\ \mid\ out_{in\langle M\rangle}\ ]\end{aligned}$$

A rule to set σ has a one-step lifetime but zero duration time.

$$\mathbf{dels} :\ \sigma \underset{0}{\rightarrow} \lambda$$

Example 4.3. The structure in Figure 4.6 has a sequence of membranes: $G_{0,2}$, $G_{0,3}$, and $G_{0,6}$. Let us assume that the objects in respective membranes are multisets w_2, w_3, w_6, respectively. According to the rules (Algorithm-II), the following actions will be performed:

1) B1 moves w_2, w_3, w_6 to their respective parents so that w_6 will be in $G_{0,3}$, w_3 will be in $G_{0,2}$, w_2 will be in $G_{0,1}$, and $G_{0,6}$ is empty (except the links and δ). B2 removes in_2 from $G_{0,1}$ and B3 removes *out* in $G_{0,4}$ and $G_{0,5}$.

2) B1 further moves w_3 and w_6 so that w_6 will be in $G_{0,2}$, w_3 will be in $G_{0,1}$, and $G_{0,3}$ is empty (except the links and δ). Note that, B4 is not performed in $G_{0,6}$ but since the membrane has no child membranes (nothing to copy).

3) B1 further moves w_6 so that w_6 will be in $G_{0,1}$, and $G_{0,2}$ is empty (except the links and δ). B4 copies *out* (including its content) from $G_{0,3}$ to $G_{0,5}$ and $in5$ (including its pointer) to $G_{0,2}$. (As the result, $G_{0,5}$ becomes a child membrane of $G_{0,2}$.)

4) B4 copies *out* (including its content) from $G_{0,2}$ to $G_{0,4}$, and $in4$ (including their pointers) to $G_{0,1}$. (As A result, $G_{0,4}$ becomes child membrane of $G_{0,1}$.) However, it can be done to $G_{0,5}$ since $G_{0,5}$ has one *out*. Thus, B3 takes place to remove *out* from $G_{0,5}$.

5) B4 copies *out* (including its content) from $G_{0,2}$ to $G_{0,5}$, and $in5$ (including their pointers) to $G_{0,1}$. (As A result, $G_{0,5}$ becomes child membrane of $G_{0,1}$.)

6) At this moment, only a reaction is performed that removes σ in global membrane.

Membranes $G_{0,2}, G_{0,3}, G_{0,6}$ are not part of the structure anymore since they are not accessible from $G_{0,1}$. At this point, they contain only links and δ. Figure 4.6 shows the structures before and after the arrangement.

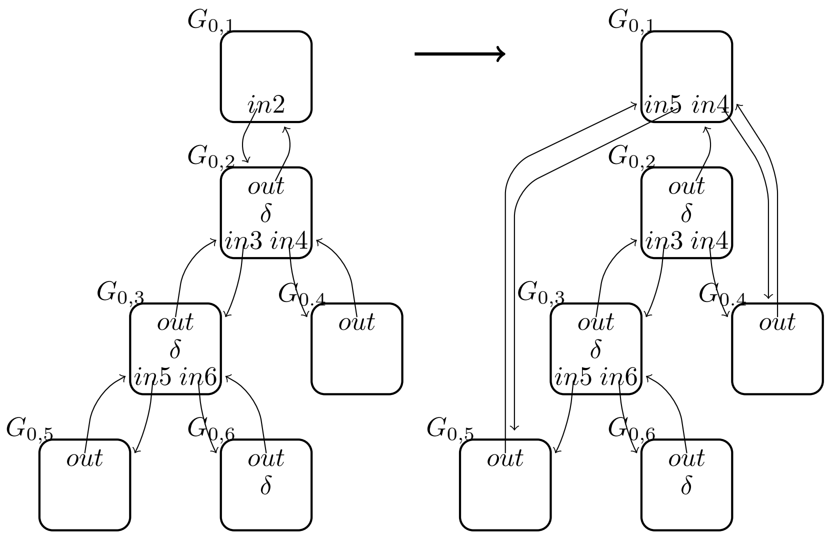


Figure 4.6: The structures before and after performing rearrangement in Example 4.3

4.3.3 Casting the Priorities

The basic idea of casting priorities of P System Π by Grid System G is to perform a preliminary phase before the actual reactions that will detect and list the reactions rules that can be applied disregarding their priorities. Then, from the result every reaction can be applied only when there is no other applicable reaction whose priority is higher than it. Some additional objects are needed to function as flags to indicate the rules' applicability.

Before further discussion, let us define that $r_{\langle i\rangle}$ is the reaction in Π whose form is:

$$\boldsymbol{\alpha}(r_{\langle i\rangle}) \;\rightarrow\; \boldsymbol{\beta}(r_{\langle i\rangle})\ [\boldsymbol{\phi}(r_{\langle i\rangle}) \;\mid\; \boldsymbol{\chi}(r_{\langle i\rangle})]$$

Note that, this form includes P Systems with promoters and inhibitors as well. In the case of the original definition of P Systems $\boldsymbol{\phi}(r_{\langle i\rangle})$ and $\boldsymbol{\chi}(r_{\langle i\rangle})$ are considered empty multisets (λ's).

According to its definition, membrane M_m has rules R_m and priority setting ρ_m. ρ_m forms a directed acyclic graph between every $r_{\langle i\rangle} \in R_m$; $\rho_m = \{(r_{\langle i\rangle}, r_{\langle j\rangle}) \mid r_{\langle i\rangle}, r_{\langle j\rangle} \in R_m\}$ where edge $(r_{\langle i\rangle}, r_{\langle j\rangle})$ denotes that the priority of $r_{\langle i\rangle}$ is higher than $r_{\langle j\rangle}$ (or $r_{\langle i\rangle} > r_{\langle j\rangle}$). $r_{\langle i\rangle}$'s related reaction rule in G is $r\langle i\rangle$ and the flags for $r\langle i\rangle$ are $F\langle i\rangle$ and $E\langle i\rangle$. $F\langle i\rangle, E\langle i\rangle \notin V$. $F\langle i\rangle$ indicates that $r\langle i\rangle$ is not applicable and $E\langle i\rangle$ indicates that $r\langle i\rangle$ is applicable.

The set of additional rules for turning on the flags is as follows.

$$\forall r_{\langle i\rangle} \in R_m.\ \mathbf{set}\langle\mathbf{i}\rangle :\ F\langle i\rangle \underset{0}{\rightarrow} E\langle i\rangle[\boldsymbol{\alpha}(r_{\langle i\rangle}) \mid \sigma_E]$$

The rules have zero time duration so that all these simulated reactions will be performed in the same time as when their related reactions in Π are being performed. To suspend other reactions from being reacted during this phase, each reaction rule $r_{\langle j\rangle}$ should be reacted only by defining $E\langle j\rangle$ as its promoter and be blocked by defining all $E\langle i\rangle$ of its predecessors as its inhibitors. Given that

$$\text{pred}(r_{\langle j\rangle}) = \{r_{\langle i\rangle} | (r_{\langle i\rangle}, r_{\langle j\rangle}) \in \rho_m \vee (r_{\langle i\rangle} \in \text{pred}(r_{\langle k\rangle}) \wedge (r_{\langle k\rangle}, r_{\langle j\rangle}) \in \rho_m)\}$$

For every $r_{\langle j\rangle} \in R_m$ of Π (which r_i is in a form of $\boldsymbol{\alpha}(r_i) \rightarrow \boldsymbol{\beta}(r_i)\ [\boldsymbol{\phi}(r_i) \;\mid\; \boldsymbol{\chi}(r_i)]$), $r\langle j\rangle$ in G will be in a form as follows.

$$\mathbf{r}\langle\mathbf{j}\rangle :\ \boldsymbol{\alpha}(r_{\langle j\rangle}) \underset{1}{\rightarrow} \boldsymbol{\beta}(r_{\langle j\rangle})\ [\ \boldsymbol{\psi}(r_{\langle j\rangle}) \cup \{E\langle j\rangle\} \;\mid\; \boldsymbol{\chi}(r_{\langle j\rangle}) \cup \{E\langle i\rangle \mid r_{\langle i\rangle} \in \text{pred}(r_{\langle j\rangle})\}\ \sigma_E\ ]$$

Additional rules are needed to reset $E\langle i\rangle$ back to $F\langle i\rangle$ as follows.

$$\forall\langle i\rangle \in \{1, ..., n\}.\ \mathbf{res}\langle\mathbf{i}\rangle :\ E\langle i\rangle \underset{1}{\rightarrow} F\langle i\rangle\ [\lambda \mid \sigma_E]$$

Applying rules r_i's and resetting the flags are performed at the same time and over the same duration so that there is no other time to spend for the mechanism. Note that when the system starts each object $F\langle i\rangle, \forall i \in \{1, ..., n\}$, should exist in the membrane with which its rule is associated.

Example 4.4. In Figure 4.3 the P System has a priority setting on r_6, r_7 and r_9 so that r_9 can only be applied when both r_6 and r_7 are not applicable. The rules of a Grid System to simulate the P System are as follows.

$$\begin{array}{rcll}
r1: & A & \underset{1}{\rightarrow} & \delta\ \sigma_E\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
r2: & A & \underset{1}{\rightarrow} & (A_{in3}\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
r3: & A\ C & \underset{1}{\rightarrow} & \delta\ \sigma_E\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
r4: & C & \underset{1}{\rightarrow} & D_{out}\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
r5: & B & \underset{1}{\rightarrow} & B\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
r6: & C & \underset{1}{\rightarrow} & C_{in4}\ \ [\ E6\ \mid\ \sigma_E\ ] \\
r7: & C & \underset{1}{\rightarrow} & B_{in4}\ \ [\ E7\ \mid\ \sigma_E\ ] \\
r9: & A & \underset{1}{\rightarrow} & A_{in2}\ B\ \ [\ E9\ \mid\ E6\ E7\ \sigma_E\ ] \\
set6: & F6 & \underset{0}{\rightarrow} & E6\ \ [\ C\ \mid\ \sigma_E\ ] \\
set7: & F7 & \underset{0}{\rightarrow} & E7\ \ [\ C\ \mid\ \sigma_E\ ] \\
set8: & F8 & \underset{0}{\rightarrow} & E8\ \ [\ D^2\ \mid\ \sigma_E\ ] \\
set9: & F9 & \underset{0}{\rightarrow} & E9\ \ [\ A\ \mid\ \sigma_E\ ] \\
res6: & E6 & \underset{1}{\rightarrow} & F6\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
res7: & E7 & \underset{1}{\rightarrow} & F7\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
res8: & E8 & \underset{1}{\rightarrow} & F8\ \ [\ \lambda\ \mid\ \sigma_E\ ] \\
res9: & E9 & \underset{1}{\rightarrow} & F9\ \ [\ \lambda\ \mid\ \sigma_E\ ]
\end{array}$$

4.3.4 Conversion Algorithm

Formally, let $\Pi = (V, \mu, w_1, ..., w_n, (R_1, \rho_1), ..., (R_n, \rho_n), i_0)$ be a P System. A Grid System $G = (\Sigma, R, A, C^{(0)})$ can be constructed to imitate Π by defining the components of G according to the following formulae.

The objects in Σ are the ones from V, dummy objects δ and σ, the flags in setting the priority, and links for simulating the membranes:
$\Sigma = V \cup \{\delta, \sigma\} \cup \Sigma_\rho \cup \Sigma_\delta$,
where $V, \{\delta, \sigma\}, \Sigma_\rho$ and Σ_δ are disjoint sets, and:

- $\Sigma_\rho = \bigcup_{m\in\{1..n\}} \bigcup_{(r_{\langle i\rangle}, r_{\langle j\rangle})\in\rho_m} \{E\langle i\rangle, F\langle i\rangle, E\langle j\rangle, F\langle j\rangle\}$
- $\Sigma_\delta = \{out\} \cup \bigcup_{\langle M\rangle\in\{2..n\}} \{in\langle M\rangle\}$

The reaction rules in R are the modified rules from Π, the rules to handle the flags, the rules for rearranging the links, a rule to remove δ:
$R = R_\Pi \cup R_\rho \cup R_\delta \cup R_\sigma$,
where:

- $R_\Pi = \bigcup_{m\in\{1,...,n\}} \{\mathbf{r}\langle\mathbf{j}\rangle(\boldsymbol{\alpha}(r_{\langle j\rangle})\ ;\ \boldsymbol{\phi}(r_{\langle j\rangle})\ ,\ \boldsymbol{\chi}(r_{\langle j\rangle})\ \sigma_E\ \cup \{E\langle i\rangle | r_{\langle i\rangle} \in \text{pred}(r_{\langle j\rangle})\}\ ,\ 1\ ,\ 1\ ,\ D) = \boldsymbol{\beta}(r_{\langle j\rangle})\ \sigma_E^{N(\delta)}\ \mid\ r_{\langle j\rangle} \in R_m \text{ and } N = \boldsymbol{\beta}(r_{\langle j\rangle})\}$.
- $R_\rho = \bigcup_{m\in\{1,...,n\}} \{\mathbf{set}\langle\mathbf{i}\rangle(F\langle i\rangle\ ;\ \boldsymbol{\alpha}(r\langle i\rangle)\ ,\ \sigma_E\ ,\ 1\ ,\ 0\ ,\ D) = E\langle i\rangle\ \mid\ r\langle i\rangle \in R_m\}$
  $\cup\ \bigcup_{m\in\{1,...,n\}} \{\mathbf{res}\langle\mathbf{i}\rangle(E\langle i\rangle\ ;\ \lambda\ ,\ \sigma_E\ ,\ 1\ ,\ 1\ ,\ D) = F\langle i\rangle\ \mid\ r\langle i\rangle \in R_m\}$

- $R_\delta = \{\mathbf{mv}\langle\mathbf{X}\rangle(\langle X\rangle; \delta, \lambda, 1, 0, D) = \langle X\rangle_{out} \mid \langle X\rangle \in V\}$
  $\cup \; \{\mathbf{din}\langle\mathbf{M}\rangle(in\langle M\rangle \; ; \; \delta_{in\langle M\rangle} \; \sigma_E \; , \; \delta \; , 1 \; , \; 0 \; , \; D) = \sigma_E \mid \langle M\rangle \in \{2, 3, ..., n\}\}$
  $\cup \; \{\mathbf{dout}(out \; ; \; \delta_{out} \; \sigma_E \; , \; \delta \; , 1, 0, D) = \sigma_E\}$
  $\cup \; \{\mathbf{cp}\langle\mathbf{M}\rangle(in\langle M\rangle \; ; \; \delta \;\; \sigma_E \; , \; out_{in\langle M\rangle} \; , \; 1 \; , \; 0 \; , \; D) =$
  $in\langle M\rangle_{out} : in\langle M\rangle \; out_{in\langle M\rangle} : out \; \sigma_E \mid \langle M\rangle \in \{2, 3, ..., n\}\}.$

- $R_\sigma = \{\mathbf{dels}(\sigma; \lambda, \lambda, 1, 0, D) = \lambda\}$

The associations in A are for affecting the rules in their related membranes:
$A = \bigcup_{m \in \{1,...,n\}} \{(\mathbf{r}\langle\mathbf{i}\rangle, G_{0,m}), (\mathbf{set}\langle\mathbf{i}\rangle, G_{0,m}), (\mathbf{reset}\langle\mathbf{i}\rangle, G_{0,m}) \mid r_{\langle i\rangle} \in R_m\}$
$\cup \; \{(r, G_{0,m}) \mid r \in R_\delta$ and $m \in \{1, ..., n\}\}$
$\cup \{(\mathbf{dels}, G_E)\}$

The initial configuration is defined as the initial configuration in Π, the initial flags, and the links connecting the membranes:
$C^{(0)} = (\{C^{0,G_{0,m}} | m \in \{1, ..., n\}\} \cup \{C^{0,G_E}\}, \emptyset)$ where
$C^{0,G_{0,m}} = w_m \cup \{F\langle j\rangle | r_{\langle j\rangle} \in P_{pri}\}$
$\cup \{in\langle j\rangle : [0, \langle j\rangle] \mid (r_m, r_j) \in \mu\}$
$\cup \{out : [0, \langle i\rangle] \mid (r_i, r_m) \in \mu\}$
$C^{0,G_E} = \emptyset$
given that $P_{pri} = \bigcup_{(r\langle i\rangle, \langle j\rangle) \in \rho_m} \{r_{\langle j\rangle}, r_{\langle j\rangle}\}$.

Chapter 5

Implementation Issues

Grid Systems as a formalism has been defined firmly in terms of its syntax and semantics. Its syntax is the rule regarding how to express the model and its semantics describes the meaning of this model and how it will behave. In Section 3.2.6, the semantics is described as the Evolution Algorithm of Grid Systems. By following the algorithm step by step, the modellers can study and analyse their models. It is clear that simpler models may be traced manually, however, in general cases the modellers might need a software tool to run the algorithm in order to carry out more thorough analyses.

The algorithm is written as a recursive mathematical function in order to focus on its general idea. In this chapter we will identify implementation aspects related to the model representation and the evolution algorithm that are implemented by using a common imperative programming language. Thus, this chapter can be used as a development guide as well as a more detailed description of the Grid Systems. Since this chapter is not intended as the main part of the thesis, our discussion rather relies upon intuitive ideas based on the experiences during the development of our tool.

In the first section we will discuss the issues at a higher level in relation to the semantics of the Grid Systems. In the second section we will discuss the issues at a lower/technical level. In the last section we will summarize our working prototype that has been developed to support our work in analysing the models.

Before further discussion, we will categorize Grid Systems based on their features. Our motivation for this categorization is due to the large technical differences between these categories when they are implemented. The categories will be named and then they will be referred to in discussing the issues.

Firstly, we categorize them based on types of reaction rules.

- **L0 (Stepwise Rule) Grid Systems** are the ones whose reaction rules are in forms of $(\alpha; \psi, \chi, 1, 1, D) = \beta$ only. Each rule has a 'plain arrow' (the rate/duration takes default value).

- **L1 (0-1 Rule) Grid Systems** include L0 Grid Systems and the ones whose rules are in forms of $(\alpha; \psi, \chi, 1, 0, D) = \beta$. Each rule has duration of either 0 or 1 time unit.

- **L2 (Stochastic) Grid Systems** have all possible types of reaction rules in Definition 3.6.

Grid Systems are also categorized based on the size of their grid: **bounded Grid Systems** and **unbounded Grid Systems**. In the former classes association table A has a finite size. The largest column/row numbers of the membrane listed in A will be the dimension of the grid. In the latter class association table A has an infinite size. Furthermore, Grid Systems are categorized into **Grid Systems without links** and **Grid Systems with links**. The difference is described in Chapter 3 as the basic form and the extended form of Grid Systems. The following are examples of some categories.

- In Section 4.1 Turing Machines are simulated by unbounded L0 Grid Systems without links.
- In Section 4.2 CA are defined as unbounded, L0 Grid Systems without links.
- In Section 4.3 P Systems are simulated by bounded L1 Grid Systems with links.
- The model of Case Study 1 (Chapter 6) is a bounded L2 Grid System without links.
- The model of Case Study 2 (Chapter 7) is a bounded L2 Grid System with links.

5.1 Evolution Algorithm in Detail

Recall that the Evolution Algorithm describes how the configuration of the model (the Grid System) evolves over time. $C^{(t_k)}$ is the configuration at time point t_k which contains the objects in every membrane and a list of on-going reactions: $C^{(t_k,m)}$ represents the objects in membrane m and Ω^{t_k} represents the list of on-going reactions. $C^{(t_k,m)}$ contains two multisets over objects: *Avail* and *Committed*. The former represents the objects that are available for the next reactions and the latter represents the objects that are already involved in some on-going reactions. Therefore, configuration contains the model's dynamic entities. Besides this, there are some static entities: a list of objects Σ and a list of reactions rules R and the association list A. In this chapter the lists are identified respectively as Σ-table, R-table and A-table. Also, Definition 3.8, assoc(m) defines the subset of R that contains the rules associated with membrane m.

The algorithm describes the steps to be performed interactively until the termination condition that is evaluated in Step-5 is true. At each step the procedure is performed on each membrane m before proceeding to the next step except when determining t_{k+1} in Step-4 and evaluating the stop condition at the end of Step-5, which are respectively performed once in each iteration. In this context we will elaborate the procedure in each step for the current membrane m representing each membrane.

5.1.1 Step-1: to Find Reaction Candidates

Step-1 performs a computation based on *Avail*. When currently $Avail = \lambda$ the procedure continues to the next membrane. Otherwise, Step-1 will find $Cand^{(t_k,m)}$ by employing logical function applicable(r, m) (to express the property of $Cand^{(t_k,m)}$). In its procedural computation, function mult(r, m) is employed instead of applicable(r, m). For Grid Systems without links, function mult(r, m) computes the maximum applicability of r in m

which is defined as follows.

$$\text{mult}(r,m) = \begin{cases} \max\{\lfloor n/k \rfloor \mid \forall a \in \Sigma.(a,n) \subseteq \mathit{Avail} \ \wedge \ (a,k) \subseteq \boldsymbol{\alpha}(r) \ \wedge \ k \neq 0\} \\ \qquad \text{for } (\forall(a,n,p) \in \boldsymbol{\psi}(r).a^n \subseteq (\mathit{Avail}^{(t,\pi(p,m))} \cup \mathit{Commit}^{(t,\pi(p,m))}) \\ \qquad \ \wedge \ \forall(a,n,q) \in \boldsymbol{\chi}(r).a^n \not\subseteq (\mathit{Avail}^{(t,\pi(q,m))} \cup \mathit{Commit}^{(t,\pi(q,m))})), \\ 0 \qquad \text{otherwise.} \end{cases}$$

Note that, mult$(r,m) > 0$ implies applicable(r,m) to be true. Furthermore, Step-1 will be expressed imperatively as follows.

1. Compute $C0$, a multiset over assoc(m), containing all reaction rules and their respective maximum applicability.

   $$C0 := \{(r, \text{mult}(r,m)) \mid r \in \text{assoc}(m) \text{ and } \text{mult}(r,m) > 0\}$$

2. Mark the objects as $\Sigma_{deficit}$ when they are required in $C0$ more than their availability in *Avail*.

   $$\Sigma_{deficit} := \{a \mid i < j \text{ where } (a,i) \subseteq \mathit{Avail} \text{ and } (a,j) \subseteq \boldsymbol{\alpha}(C0)\}$$

   Note that, as in Evolution Algorithm, $\boldsymbol{\alpha}(C0) = \bigcup_{(r,s)\in C0} s \cdot \boldsymbol{\alpha}(r)$

3. Separate $C0$ into Cn and Cd. Cn will contain the rules which are involved in non-determinism and Cd will contain the rest.

   $$\begin{aligned} Cn &:= \{(r,s) \mid (r,s) \in C0 \text{ and } \exists((a,n) \subseteq \alpha(r) \ \wedge \ a \in \Sigma_{deficit})\} \\ Cd &:= C0 \setminus Cn \end{aligned}$$

4. Resolve non-determinism in Cn by a stochastic rule selection adapted from Gillespie's SSA and the selected rules will be in Cs. This is an iterative process of computing propensities of each rule in Cn and randomly taking one out from Cn and granting the objects to be its reactants, until there are no more available objects that can be the reactants of the rest.

   - Define the propensity function $H(r,A)$ for reaction rule r, and multiset A (over Ω) is as follows:

     $$H(r,A) = \text{rate}(r) \cdot \prod_{\substack{(a^k \subseteq \boldsymbol{\alpha}(r)) \\ \wedge\,(a^n \subseteq A)}} C(n,k), \text{ where } C(n,k) = \frac{n!}{k!(n-k)!}$$

     Recall that $C(n,k) = 0$ when $n < k$. This implies $H(r,A)$ be 0. Therefore, r could be removed from Cn to reduce the number of iterations when for $(a^k \subseteq \boldsymbol{\alpha}(r)) \wedge \ (a^n \subseteq A)$, condition $n < k$ is held.

   - Initialization:
     $Cs := \lambda$
     $A := \mathit{Avail}$

- Given that $\{r_1, r_2, ..., r_N\}$ is the set of different rules in Cn, $H(r, A)$ is the propensity function of rule r by considering available objects A, and $P[0..N]$ is an array of real numbers. The following operations are to be performed iteratively after changing A:

```
$P[0] = 0$
$(\forall i \in \{1, ..., N\}).P[i] := P[i-1] + H(r_i, A)$
if $(P[N] > 0.0)$
then
      $t := P[N] \cdot X,$ where $X \sim U[0, 1]$  ($X$ is a uniformly distrib. RV)
      $s := k - 1$ where $P[k-1] \leq t < P[k]$
      $A := A \setminus \alpha(r_s)$
      $Cs := Cs \cup (r_s, 1)$
else
      return
endif
```

5. $Cand := Cd \cup Cs$ will be the result of Step-1.

For Grid Systems with links, function mult(r, m) and the procedures described above need to be more sophisticated in order to evaluate related objects to each rule by considering the possibilities as follows:

- $(a\!:\!\eta, k) \subseteq \boldsymbol{\alpha}(r)$ where η is a pointer
- $(a\!:p, k) \subseteq \boldsymbol{\alpha}(r)$ where p is a link
- $(a\!:\!\eta, n, q) \subseteq (\boldsymbol{\psi}(r) \cup \boldsymbol{\chi}(r))$ where η is a pointer
- $(a\!:\!p, n, q) \subseteq (\boldsymbol{\psi}(r) \cup \boldsymbol{\chi}(r))$ where p is a link
- $(a\!:\!\eta, n, q) \subseteq (\boldsymbol{\psi}(r) \cup \boldsymbol{\chi}(r))$ where η is a pointer and q is a link
- $(a\!:\!p, n, q) \subseteq (\boldsymbol{\psi}(r) \cup \boldsymbol{\chi}(r))$ where p and q are links

The evaluations are in accordance with the discussion to modify the Evolution Algorithm in Section 3.3.2.

5.1.2 Step-2: to Commit Objects as Reactants

Step-2 changes states of objects that are taken as reactants of each reaction in $Cand$ from 'available' to 'committed'. The task can be simply embedded in Step-1 when the contents of Cd and Cs are merged into $Cand$.

5.1.3 Step-3: to Add New On-going Reactions to Ω-table

This step creates the new entries representing on-going reactions into Ω-table. The source of the task is the contents of $Cand$ of every membrane m. For $L0$ Grid Systems, elapse(r) is ignorable since all those reactions will terminate in the next step. For $L1$ Grid Systems, elapse(r) will return either 0 or 1 according to the duration of rule r. For $L2$ Grid Systems, elapse(r) will return the duration of r when r is a deterministic duration time rule.

When r has an exponentially distributed duration time, by using the inverse method, elapse(r) can be defined as follows.

$$\text{elapsed}(r) = -\text{duration}(r)\ln X \text{ where } X \sim U[0,1] \;\; (X \text{ is a uniformly distrib. RV}).$$

For $(r, n) \in Cand$ where $n > 1$, each one reaction of r is likely to have different values of elapsed(r). Therefore, each one should be put as one entry in the Ω-table. For Grid Systems with links, replication as described in Section 3.3.2 should be performed and selected pointers should be attached in its respective entry in Ω-table. As for elapsed(r), this should be performed for every one when $n > 1$.

5.1.4 Step-4: to Get the Earliest Reactions to Terminate

For $L0$ Grid Systems all entries in Ω-table will terminate at the same time therefore this task is trivial. For $L1$ Grid System, at every time point t_k, they are reactions that will terminate either at t_k or at $t_k + 1$. We could simply use two subsets of Ω for each time respectively. When there is no rule to terminate at t_k then take from $t_k + 1$. For $L2$ the task could be performed more efficiently by using a priority queue (heap tree) data structure for Ω-table. Such a data structure makes it possible to insert and remove entries in logarithmic time. This is important when the number of on-going reactions is huge.

5.1.5 Step-5: to give Products and to Remove Reactants

Let us revisit the expression to change $Avail$ in Step-5 as follows.

$$Avail^{(t_{k+1},m)} = \widehat{Avail}^{(t_k,m)} \cup \left\{ \bigcup_{m' \in G} \left\{ \bigcup_{(a,n,p) \in \boldsymbol{\beta}(F^{(t_{k+1},m')}),\pi(p,m')=m} a^n \right\} \right\}$$

This expression is intentionally written like this due to its mathematical purpose (as a function of many entities). When it is translated into imperative expressions, $Avail$ in the current membrane m is updated by scanning every membrane which has products in m. This will be more efficient when the products are sent from the membrane where the reaction is taking place to the destined membranes, as the following imperative tasks.

$$\begin{array}{l} (\forall (r,k) \subseteq F^{(m)}).(\forall (a,n,p) \subseteq \boldsymbol{\beta}(r)). \\ \qquad Avail^{(m')} = Avail^{(m')} \cup a^{n \cdot k} \text{ where } \pi(p,m) = m' \text{ and } m' \in G \end{array}$$

5.2 Technical Considerations in the Implementation

We will discuss some strategies in the real implementation of the software tool for Grid Systems. We start by considering what and how the data structures used by the software will be useful. Then, we will focus on some problems related to real conditions in computing systems. At the end, we will discuss the possible optimisation that will be useful when the software is implemented on a multi-core architecture, like GPUs.

5.2.1 Bounded and Unbounded Grids

By considering the models which have the number of membranes is uncertain, grid systems enable us to define a model with infinite grid dimensions. Syntactically, this infinity is defined by A-table. In its implementation this could be handled by the dynamic allocation memory strategy for the membranes. Initially, the membranes are allocated for the ones whose initial objects are in them. The next allocation will be performed when there is at least one product to be put in the membrane. This is possible since in Grid Systems it is defined that the reactions can only take reactants from the membrane where the reaction is taking place.

However, it may be preferable to implement bounded Grid Systems rather than unbounded Grid Systems. This is related to the fact that in actual population models the size of the grid is normally certain. Furthermore, the implementation of the bounded Grid Systems may reduce the computation cost for checking the availability of membranes since the membranes is pre-allocated.

5.2.2 Internal Model Representation and Scalability

To optimize the performance of the computation it would be better to represent the objects in Σ-table, the rules in R-table as their entry numbers in the table. This will reduce the need for string-based searching in the tables. The most efficient string-based searching needs $O(\lg N)$ time, whereas by using their indices it needs $O(1)$ time.

As a consequence, the objects in each rule should also be represented by object numbers and the rules in the A-table and Ω-table should also be represented by rule numbers. However, those are just internal representations. In monitoring the current state of the systems, a module for converting back the numbers into their original identification should be implemented. Besides this, during loading the model, a module should be implemented to convert the model to such an internal representation.

In its internal representation, *Avail* and *Commit* will be represented by arrays of integers of the length N_Σ (for $N_\Sigma = |\Sigma|$). $L(m,p)$ will be represented as a dynamic length array (vectors in Java) or linked lists for each link p. Rule's components, α, β, ψ, and χ, can be represented by associated arrays whose elements contain (object number, multiplicity, spatial information, link information).

Scalability of the tool will rely on the parts of $C^{t)}$:

- The number of bytes for each object in *Avail* and *Commit*: If the effective number of membranes is N_G, the number of object is N_Σ, and N_B is the number of bytes, and the memory space for the objects for all membranes is $2 \times N_B \times N_G \times N_\Sigma$. For instance, a 4-byte, 100-object, 1000×1000-membrane model requires ¡ 1GByte excluding the space for $L(m,p)$.

- The largest Ω-table can be handled at once: Note that the Evolution Algorithm is based on the Breadth First Search (BFS) algorithm, so that for some cases the number of on-going reactions can explode. Therefore, an Ω-table should be implemented using dynamic length arrays so that the length can be extended whenever it is needed. However, the size of one entry is quite small ($\sim$ 5 bytes) so that the Ω-table can be extended to be as large as the memory can handle. For instance, current personal computers can provide several GBytes of memory for a running process, and each GByte can be used for 200 million entries of Ω-table.

5.2.3 Timing Resolution

For L0 and L1 Grid Systems the time is already quantified, whereas in L2 Grid Systems the time is real. Therefore in this part we discuss only L2 Grid Systems. In practical cases of highly accurate time computation, every on-going reaction could be timely unique so that it should be handled as one entry in Ω-table. This situation will explode the size of Ω-table and result in a high computation cost. Furthermore, in each time of iteration of the algorithm there could be one reaction at a unique time to terminate which causes the performance of the tool to be suffered.

Since in some practical cases the time accuracy may be compromised to some extent, the tool should enable the modeller to set up the time accuracy of the model. By lesser accuracy, firstly, there could be some number of reactions can be recorded one entry in Ω-table with multiplicity more than one. Thus, this will reduce the size of Ω-table. Secondly, the number of reactions to terminate at one time will increase, so they can be handled in one time of iteration, and when the tool is implemented in parallel it will increase the performance of the tool.

The time accuracy is defined by a time interval δ in which the time will be rounded down. In its implementation, δ will be used in computing elapse(r) as follows.

$$\text{elapsed}(r) = \lfloor (-\text{duration}(r) \ln X)/\delta \rfloor \delta$$

5.2.4 Handling Propensity Computation

In computing propensities we need a trick to handle big numbers of $C(n, k)$. We propose that the computations are performed in the log domain. Therefore, instead of computing $H(r, A)$ we compute $\log(H(r, A))$. In this domain we are working with additional real numbers as follows.

$$\log(H(r,A)) = \log(\text{rate}(r)) + \sum_{\substack{(a^k \subseteq \boldsymbol{\alpha}(r)) \\ \wedge\ (a^n \subseteq A)}} \log(C(n,k))$$

It is already shown in Section 5.1.1, if $\exists a, (a^k \subseteq \boldsymbol{\alpha}(r)) \wedge\ (a^n \subseteq A). n < k$ implies that $H(r, A) = 0$. Thus, the following is the computation for $n \geq k$.

$$\begin{aligned} \log(C(n,k)) &= \log\left(\frac{n!}{k!(n-k)!}\right) \\ &= \log(n!) - \log(k!) - log((n-k)!) \\ &= \textstyle\sum_{i=1}^{n} \log(i) - \sum_{i=1}^{k} \log(i) - \sum_{i=1}^{n-k} \log(i) \end{aligned}$$

Furthermore, by defining a lookup table $\text{SLog}[t] = \sum_{i=1}^{t} \log(i)$ (which can be precomputed in the beginning as $\text{SLog}[t] = \log(t) + \text{SLog}[t-1],$ for $t > 1$ and $\text{SLog}[1] = 0$), we get the following a less-computation function:

$$\log(C(n,k)) = \text{SLog}[n] - \text{SLog}[k] - \text{SLog}[n-k], \text{ for } n \geq k$$

Thus, by an exponentiation we can get $H(r, A)$ back. However, this can still possibly produce big numbers. To avoid this, we suggest the normalization of $H(r, A)$'s by subtracting each $H(r, A)$ by $\text{Max} = \max_i\{\log(H(r_i, A))\}$ so that

$$\exp\left(\log(H(r, A) - \text{Max}\right) = \frac{H(r, A)}{\max_i\{H(r_i, A)\}} \in [0, 1]$$

The use of normalized propensities can be continued in the selection task because the task needs only their proportions in making the ranges $(P[k-1], P[k])$.

5.2.5 Special Handling for External Events

As described in Section 3.4, an external event is a reaction that will be activated from $t = 0$ to produce an object at time d (d is the duration of the rule) where the produced object will trigger the actual reaction of the event. The actual reaction of the event is defined as one rule that is associated to the Global Membrane. However, each reaction to produce the trigger will be defined as one rule respectively; it will only be used once in a whole simulation. For a model which has many such external events, this will result in an unnecessary situation: a very long R-table but most of the contents are used once. We recommend treating the rules from the events differently, as follows. The ordinary rules are placed in the lowest indices of R-table, and they are followed by the rules from events that are placed decreasingly by the time d. The table can be compacted by removing the entries in R-table that have already been 'executed' and by removing their related entries in A-table of Global Membrane. This will reduce the time needed for Step-1 of Evolution Algorithm as well as the memory space for unused entries of A-table and R-table.

5.2.6 Model Scripting and Interpreting

Besides the syntax of Grid System described in Section 3.2 and 3.3, we propose an extension of this in order to shorten the scripting of the model, as described in Section 3.4. To facilitate this, an interpreter should also be developed to translate the model script into the model according to its basic syntax. There are two possible recommendations for scripting the format for the model: textual and XML-tagged format. The former is useful in writing the model by using ordinary text editors, and the latter is useful when the GUI based model editor is available.

The model should consist of several parts:

- Set of Objects. For example:

$$\Sigma = \{A, B, C, D, E, F\}$$

- List of Constants and their values that are literals of numbers/strings or simple arithmetic expressions of other numbers/strings/constants. For example:

$$\begin{aligned} &nrow &&= 50 \\ &maxrate &&= 10 \\ &\ldots \end{aligned}$$

- List of Regions and their contents that are respectively a set of membranes or expressions of set operations over other regions. For example:

$$\begin{aligned} &reg1 &&= \{\ldots\} \\ &reg2 &&= \text{rectangle}[ra, ca, rb, cb] \\ &regK &&= (regA \cup regB) \setminus regC \end{aligned}$$

- List of Rules and/or Rule Templates. For example:

$$\begin{array}{l}\rho\mathbf{1}(\alpha 1; \psi 1, \chi 1, r1, d1, M) = \beta 1 \\ ... \\ \forall \langle X \rangle \in \mathcal{L}.\ \rho\langle \mathbf{X} \rangle(\alpha\langle X \rangle; \psi\langle X \rangle, \chi\langle X \rangle, r\langle X \rangle, d\langle X \rangle, M) = \beta\langle X \rangle \\ ...\end{array}$$

- Set of associations where each element describe an association between a rule and either membrane or a region. For example:

$$A = \{(\rho 1, \gamma 1), (\rho 2, \gamma 2), ...(\rho 1, reg1), (\rho 2, reg2), ...\}$$

- Initial objects in terms of multisets and a membrane or a region. Note that, declaring initial objects in a region implies that the objects will be multiplied in each membrane of this region. For example:

$$\begin{array}{l}C^{m1} = \rho 1 \\ C^{m2} = \rho 2 \\ ... \\ C^{reg1} = \rho 1 \\ C^{reg2} = \rho 2 \\ ...\end{array}$$

- List of external events (if any) which are respectively pairs (rule id,time). For example:

$$E = \{(\rho 1, 11), (\rho 2, 20)\}$$

In addition to this, what and how the output should be produced by the tool should be described in the script, too. It will be further described in the next section.

5.2.7 Output Presentation Format

Internally the configuration in Grid Systems is represented as multisets over objects. In order to observe the dynamics in simulating the system a visualization tool is needed. The kind of data visualisation depends on our specific need. However, from our experience in working with the case studies and some general computation models we have several visualisation schemes as listed in the following list.

- As plain multisets of each membranes for visual inspection. This is suitable when the number of membranes and the number of objects to be observed are small.

- As CSV (comma separated values) to be easily read by other software. This is more suitable for a large number of objects/membranes which needs software for analysing/visualising the output.

- As an animation. This is suitable for the observation of the spatial behaviour of the model. For the models with less computation, the output can be directly displayed and for the ones with a heavier computation the animation can be dumped periodically so that they can be replayed later.

5.2.8 Parallel Implementation Issues

In the discussion Section 5.1 we assume that the membranes are processed serially. However, the Evolution Algorithm takes advantage of the parallelism property of Grid Systems so that each membrane can be processed simultaneously, except when the processes should follow a common timing. The common timing controller (determining t_{k+1} in Step-4) and the stop condition checker (end of Step-5) are handled by the master process, and each membrane will be handled by a slave process.

Moreover, the Ω-table can be distributed from the very beginning to each membrane so that no membrane-id m is needed to be saved in the Ω-table of membrane m. It is possible to implement the objects in the membrane (as multisets *Avail* and *Commit*) as local data for the slave process respectively, as well as providing services to other membranes. The services:

- query its local objects in checking promoters and inhibitors of the rules in other membranes (Step-1);
- accept products of the rule from other membranes (Step-5); and
- answer the query from the master process about the time of the earliest reaction in its Ω-table.

Previously we discussed that in unbounded Grid Systems the actual membranes are created when an ‘initially-empty membrane’ will accept the products of a reaction. For this case a slave process will also be instantiated to manage this newly created membrane.

5.3 Our Working Prototype

In this part we will describe the tool that we have developed according to the ideas in the previous parts. Up to this time, our tool is still in the category L3, bounded Grid Systems with links. Therefore, the only feature that is not implemented yet is the unbounded Grid Systems whose functionalities are enough for our researches. However, since the tool is still a working prototype, its performance could be further optimized. In this chapter this prototype is reported as an illustration for the previous sections.

5.3.1 Development Platform

The tool was developed using the Java language and by using Net Bean IDE as its development tool. The latest version of the tool was updated to Java 7.6 and Net Beans IDE 7.4. In order to ease in writing the model and visualising the output, a GUI-based modelling module was also developed by using JSwing, for the interface, and AWT, for the graphical output. The modelling module provides the interface for directly controlling the functionality of the simulation module and visualising the output so that the models can be instantly evaluated and analysed. The model can be written in the storage as the XML tagged files and be reloaded to the memory for analysing. The routines for reading/writing are developed by using org.w3c.dom and javax.xml packages. The graphical output module enables us to dump the output on the screen as PNG images.

5.3.2 Simulation Engine

The engine is the module that runs the Evolution Algorithm. Therefore, the idea of the engine has been described in detail in previous sections in this chapter. As it is implemented in Java language, the module is coded to have some classes and the class diagram of major classes is shown in Figure 5.1 and its sequence diagram is shown in Figure 5.2.

The main class of the engine is GridSystems. The mechanism will be roughly described as follows. The engine loads the model into ObjectTable, RuleTable, and RuleAssociation, and the initial objects into the Mebrane. Then, the method simulate() in Configuration is driven to perform the master process of the simulation. The slave processes are performed in each membrane. Note that, this prototype still runs sequentially, so that one slave process is performed at a time. The slave process drives the method getNextReactions() in OmegaList to perform Step-1 (producing *Cand*) and methods in MultisetObjects to perform Step-2 (updating *Avail* and *Commit*). After that, the master process in Configuration increases the time to the earliest time of on-going reactions to finish and drives the membranes to 'execute' the terminated reactions by producing the reaction products and removing committed reactants.

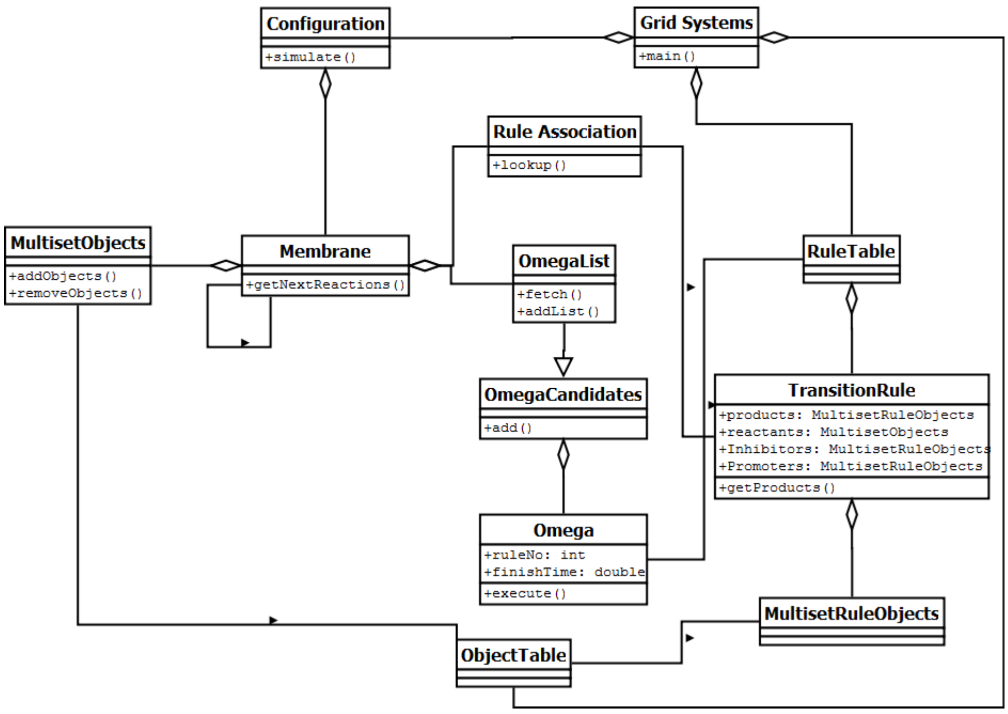


Figure 5.1: Class diagram of the simulation engine

5.3.3 An IDE for Model Development

The input of the engine can be written by using an ordinary text editor. In order to ease this modelling work, an integrated development environment (IDE) for writing the model is developed. This contains several modules as follows.

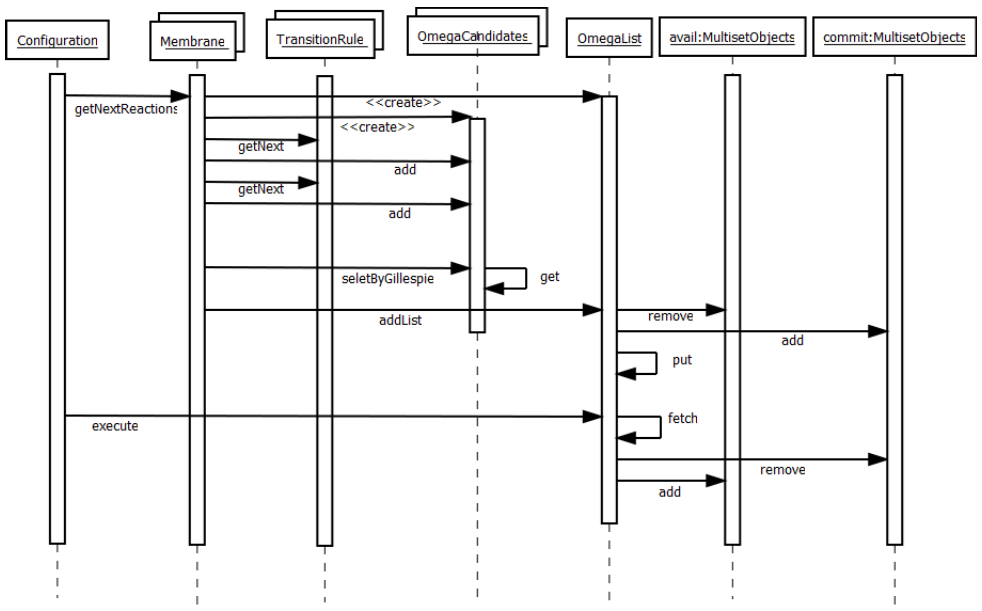


Figure 5.2: Sequence diagram of the simulation engine

- The main screen of IDE contains the list of models that are already in the repository. The modellers can load a model for editing or running, or save the currently being edited model. The screen also shows the summary of the model being edited and exports the model's rules to a text for being inserted in a LaTeXfile. From this screen the modeller can initiate a new model, too. Figure 5.3(a) shows a screen-shot of this screen.

- Objects, constants and regions are the next thing to be defined in writing a model. The second screen shows the lists of them and the buttons for maintaining (adding, deleting, ...) the lists. In defining a rule template its variables need the ranges and in this screen the ranges can be defined. Figure 5.3(b) shows a screen-shot of this screen.

- The special screen will be shown when the editing button is pressed (in the third screen) and the screen will show the grid where a region can be defined interactively. Figure 5.5(a) shows a screen-shot of this screen.

- The third screen shows the list of rules (and/or rule templates) that are already defined. The screen also shows the rules that are represented by a template to help the modeller in examining the actual rules that are made from the template. Figure 5.4(a) shows a screen-shot of this screen.

- Editing a rule or creating a new rule will open another screen for entering the rule's parameters. Figure 5.5(c) shows a screen-shot of this screen.

- The fourth screen shows the initial configuration and the buttons to add a new element of edit to the existing elements. The configuration consists of multisets of membranes or multisets of regions. A multiset of a region implies that the multiset is replicated in each membrane of that region. Besides this, the links can be placed initially in the model along with their pointers. Figure 5.4(b) shows a screen-shot of this screen.

- When the editing link button is pushed (in the fourth screen) a special screen for defining links and the pointer will be shown. Figure 5.5(b) shows a screen-shot of this screen.
- The fifth screen of the IDE shows the interface for running the simulation and presenting the output. The screen will be reported in the next section.

5.3.4 Output Presentation Modules

The simulation itself will be embedded in each module of the presentation. There are four different ways for presenting the output that were created in our prototype, as follows.

- The textual output is shown on a window and can be saved as a file. The output contains lines and each line represents the objects in every membrane in terms of multisets at a point in time. Figure 5.6(b) shows the screen-shot of an example of the charts. There is an option to present the output as CSVs as the input for other analytical tools.
- The growth charts of some populations of objects are plotted, according to the time line, in a real-time manner on a window and then the chart can be exported to a TikZ (LaTeX) text file. Figure 5.6(c) shows the screen-shot of an example of the charts.
- Real-time animation shows the population (also the links) being distributed dynamically over the space. When it is needed, the images can be dumped repetitively after a time interval creating a sequence of images that can be put together as an animated file. Figure 5.7 shows the screen-shots of the screen for entering the animation parameters and the screen of animations.
- A step-by-step tracer enables the modeller to trace how the model that is being developed will evolve. By using this presentation, each rule can be verified individually or together with several other rules. We can inspect also the contents of the individual membrane and verifies the stochasticity of in a non-deterministic situation. Figure 5.8(a) shows the controlling screen of the tracer and Figure 5.8(b) shows the configuration inside a membrane.

5.3.5 XML File Structure

As an XML file, the model description is the data framed by a pair of tags and preceded by a line of XML metadata. The detail of the file syntax/structure will be described in a separate documentation. The following one is just the outline of the structure.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<gridsystem title="the tile of the model">
.....
</gridsystem>
```

The description consists of several parts:

- The Aliases Part defines the regions, ranges, and constants.

```
<aliases>
.....
</aliases>
```

- The Objects Part lists all objects and their description if any.

```
<objectlist>
   <object id="..." desc="..."/>
    .....
</objectlist>
```

- The Rules Part lists the rules/templates, and each rule/template is defined by their components and to which membranes/regions are associated.

```
<rulelist>
   <rule id="..."  rate="..." duration="... ">
       <reactants> .... </reactants>
       <products> .... </products>
       <promoters> .... </promoters>
       <inhibitors> .... </inhibitors>
       <associations>... </associations>
    </rule>
    .....
    .....
</rulelist>
```

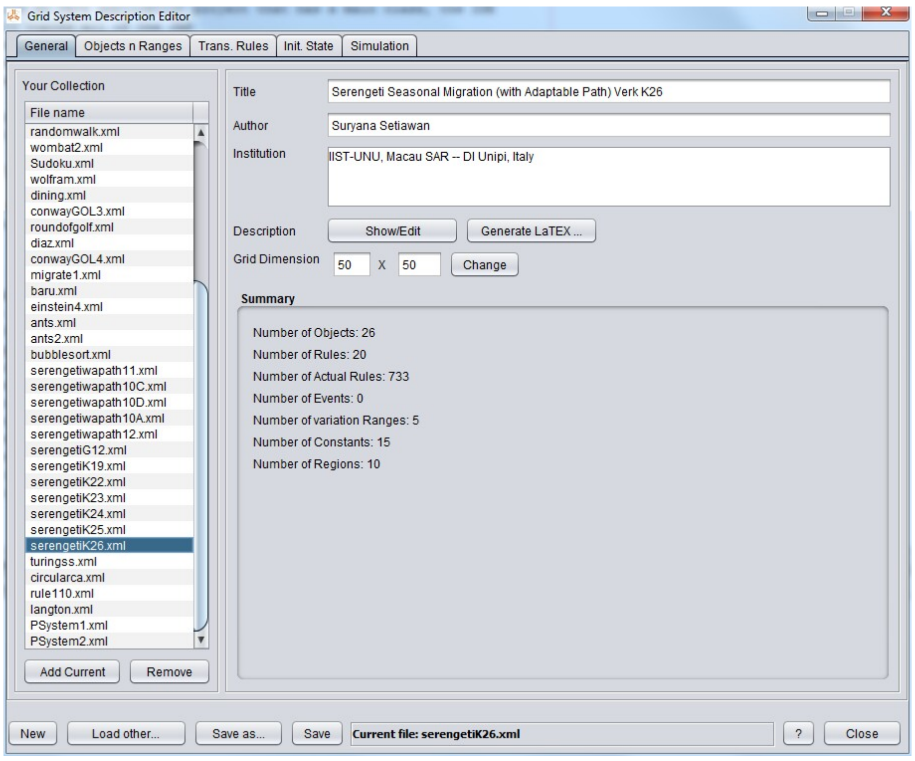
- The Configuration Part lists the initial links and objects in each membrane/region.

```
<configuration>
...
</configuration>
```

- The Simulation part contains additional information needed for running the simulations: the limits of the iteration, the objects that needed to be reported, and the colour of the objects that will be used in its animation/chart, and so on.

```
<simulation>
...
</simulation>
```

Note that, the complete syntax for writing the model will be described in Tool's Documentation.



Your Collection

File name
randomwalk.xml
wombat2.xml
Sudoku.xml
wolfram.xml
dining.xml
conwayGOL3.xml
roundofgolf.xml
diaz.xml
conwayGOL4.xml
migrate1.xml
baru.xml
einstein4.xml
ants.xml
ants2.xml
bubblesort.xml
serengetiwapath11.xml
serengetiwapath10C.xml
serengetiwapath10D.xml
serengetiwapath10A.xml
serengetiwapath12.xml
serengetiG12.xml
serengetiK19.xml
serengetiK22.xml
serengetiK23.xml
serengetiK24.xml
serengetiK25.xml
serengetiK26.xml
turingss.xml
circularca.xml
rule110.xml
langton.xml
PSystem1.xml
PSystem2.xml

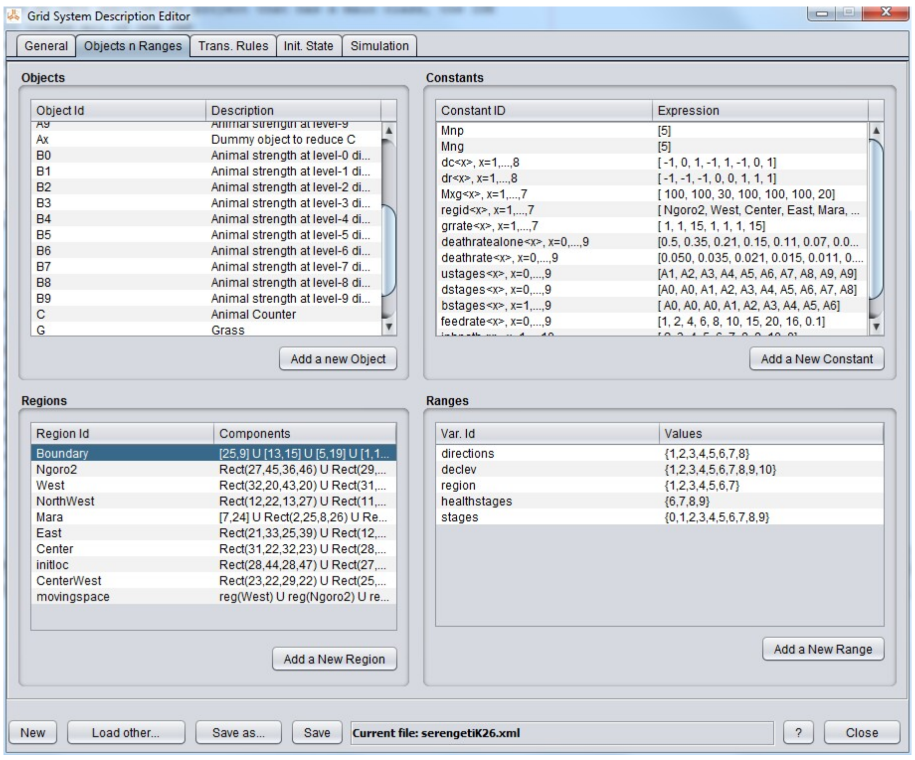
Title: Serengeti Seasonal Migration (with Adaptable Path) Verk K26
Author: Suryana Setiawan
Institution: IIST-UNU, Macau SAR -- DI Unipi, Italy
Description
Grid Dimension: 50 X 50

Summary

Number of Objects: 26
Number of Rules: 20
Number of Actual Rules: 733
Number of Events: 0
Number of variation Ranges: 5
Number of Constants: 15
Number of Regions: 10

Current file: serengetiK26.xml

(a) profiling models



Objects

Object Id	Description
A9	Animal strength at level-9
Ax	Dummy object to reduce C
B0	Animal strength at level-0 di...
B1	Animal strength at level-1 di...
B2	Animal strength at level-2 di...
B3	Animal strength at level-3 di...
B4	Animal strength at level-4 di...
B5	Animal strength at level-5 di...
B6	Animal strength at level-6 di...
B7	Animal strength at level-7 di...
B8	Animal strength at level-8 di...
B9	Animal strength at level-9 di...
C	Animal Counter
G	Grass

Constants

Constant ID	Expression
Mnp	[5]
Mng	[5]
dc<x>, x=1,...,8	[-1, 0, 1, -1, 1, -1, 0, 1]
dr<x>, x=1,...,8	[-1, -1, -1, 0, 0, 1, 1, 1]
Mxg<x>, x=1,...,7	[100, 100, 30, 100, 100, 100, 20]
regid<x>, x=1,...,7	[Ngoro2, West, Center, East, Mara, ...
grrate<x>, x=1,...,7	[1, 1, 15, 1, 1, 1, 15]
deathratealone<x>, x=0,...,9	[0.5, 0.35, 0.21, 0.15, 0.11, 0.07, 0.0...
deathrate<x>, x=0,...,9	[0.050, 0.035, 0.021, 0.015, 0.011, 0....
ustages<x>, x=0,...,9	[A1, A2, A3, A4, A5, A6, A7, A8, A9, A9]
dstages<x>, x=0,...,9	[A0, A0, A1, A2, A3, A4, A5, A6, A7, A8]
bstages<x>, x=1,...,9	[A0, A0, A0, A1, A2, A3, A4, A5, A6]
feedrate<x>, x=0,...,9	[1, 2, 4, 6, 8, 10, 15, 20, 16, 0.1]

Regions

Region Id	Components
Boundary	[25,9] U [13,15] U [5,19] U [1,1...
Ngoro2	Rect(27,45,36,46) U Rect(29,...
West	Rect(32,20,43,20) U Rect(31,...
NorthWest	Rect(12,22,13,27) U Rect(11,...
Mara	[7,24] U Rect(2,25,8,26) U Re...
East	Rect(21,33,25,39) U Rect(12,...
Center	Rect(31,22,32,23) U Rect(28,...
initloc	Rect(28,44,28,47) U Rect(27,...
CenterWest	Rect(23,22,29,22) U Rect(25,...
movingspace	reg(West) U reg(Ngoro2) U re...

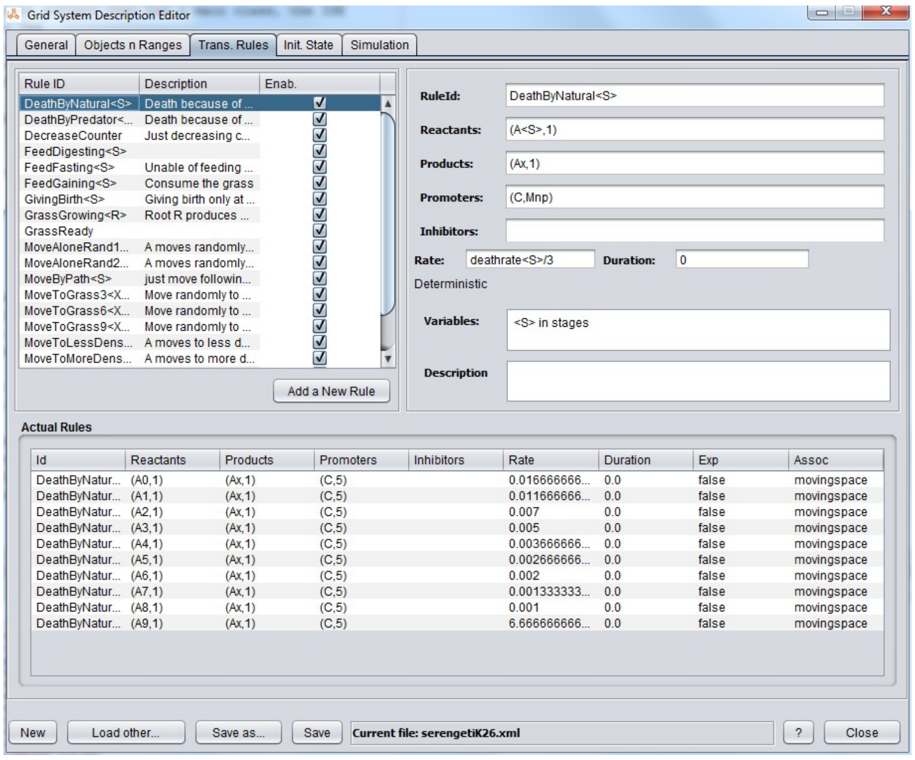
Ranges

Var. Id	Values
directions	{1,2,3,4,5,6,7,8}
declev	{1,2,3,4,5,6,7,8,9,10}
region	{1,2,3,4,5,6,7}
healthstages	{6,7,8,9}
stages	{0,1,2,3,4,5,6,7,8,9}

Current file: serengetiK26.xml

(b) defining objects, constants and regions

Figure 5.3: Screenshots of the user interfaces for editing a model (Profiling and Model Elements)



Rule ID	Description	Enab.
DeathByNatural<S>	Death because of ...	✓
DeathByPredator<...	Death because of ...	✓
DecreaseCounter	Just decreasing c...	✓
FeedDigesting<S>		✓
FeedFasting<S>	Unable of feeding ...	✓
FeedGaining<S>	Consume the grass	✓
GivingBirth<S>	Giving birth only at ...	✓
GrassGrowing<R>	Root R produces ...	✓
GrassReady		✓
MoveAloneRand1...	A moves randomly...	✓
MoveAloneRand2...	A moves randomly...	✓
MoveByPath<S>	just move followin...	✓
MoveToGrass3<X...	Move randomly to ...	✓
MoveToGrass6<X...	Move randomly to ...	✓
MoveToGrass9<X...	Move randomly to ...	✓
MoveToLessDens...	A moves to less d...	✓
MoveToMoreDens...	A moves to more d...	✓

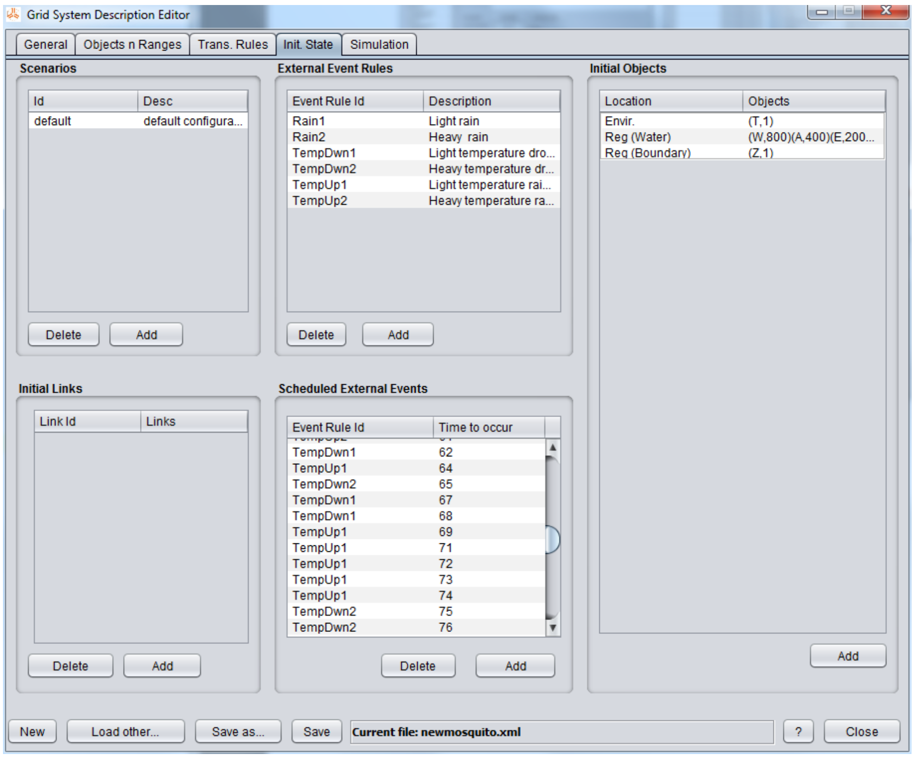
RuleId: DeathByNatural<S>
Reactants: (A<S>,1)
Products: (Ax,1)
Promoters: (C,Mnp)
Inhibitors:
Rate: deathrate<S>/3 **Duration:** 0
Deterministic
Variables: <S> in stages
Description

Actual Rules

Id	Reactants	Products	Promoters	Inhibitors	Rate	Duration	Exp	Assoc
DeathByNatur...	(A0,1)	(Ax,1)	(C,5)		0.016666666...	0.0	false	movingspace
DeathByNatur...	(A1,1)	(Ax,1)	(C,5)		0.011666666...	0.0	false	movingspace
DeathByNatur...	(A2,1)	(Ax,1)	(C,5)		0.007	0.0	false	movingspace
DeathByNatur...	(A3,1)	(Ax,1)	(C,5)		0.005	0.0	false	movingspace
DeathByNatur...	(A4,1)	(Ax,1)	(C,5)		0.003666666...	0.0	false	movingspace
DeathByNatur...	(A5,1)	(Ax,1)	(C,5)		0.002666666...	0.0	false	movingspace
DeathByNatur...	(A6,1)	(Ax,1)	(C,5)		0.002	0.0	false	movingspace
DeathByNatur...	(A7,1)	(Ax,1)	(C,5)		0.001333333...	0.0	false	movingspace
DeathByNatur...	(A8,1)	(Ax,1)	(C,5)		0.001	0.0	false	movingspace
DeathByNatur...	(A9,1)	(Ax,1)	(C,5)		6.666666666...	0.0	false	movingspace

Current file: serengetiK26.xml

(a) defining rules/templates



Scenarios

Id	Desc
default	default configura...

External Event Rules

Event Rule Id	Description
Rain1	Light rain
Rain2	Heavy rain
TempDwn1	Light temperature dro...
TempDwn2	Heavy temperature dr...
TempUp1	Light temperature rai...
TempUp2	Heavy temperature ra...

Initial Objects

Location	Objects
Envir.	(T,1)
Reg (Water)	(W,800)(A,400)(E,200...
Reg (Boundary)	(Z,1)

Initial Links

Link Id	Links

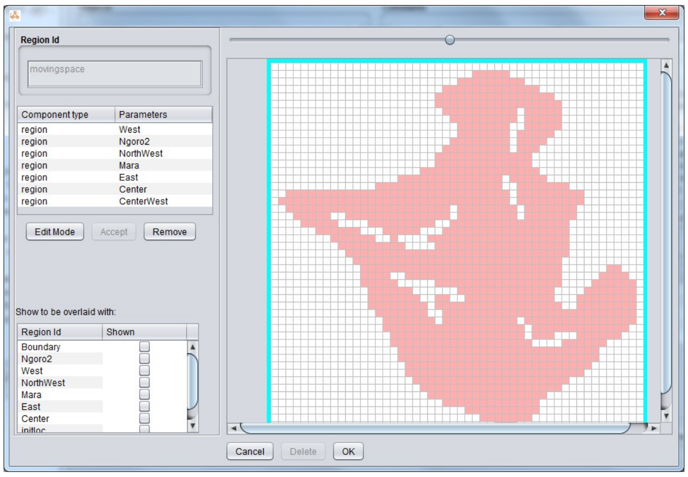
Scheduled External Events

Event Rule Id	Time to occur
TempDwn1	62
TempUp1	64
TempDwn2	65
TempDwn1	67
TempDwn1	68
TempUp1	69
TempUp1	71
TempUp1	72
TempUp1	73
TempUp1	74
TempDwn2	75
TempDwn2	76

Current file: newmosquito.xml

(b) creating initial objects

Figure 5.4: Screenshots of the user interfaces for editing a model (Rules and Initial Objects)



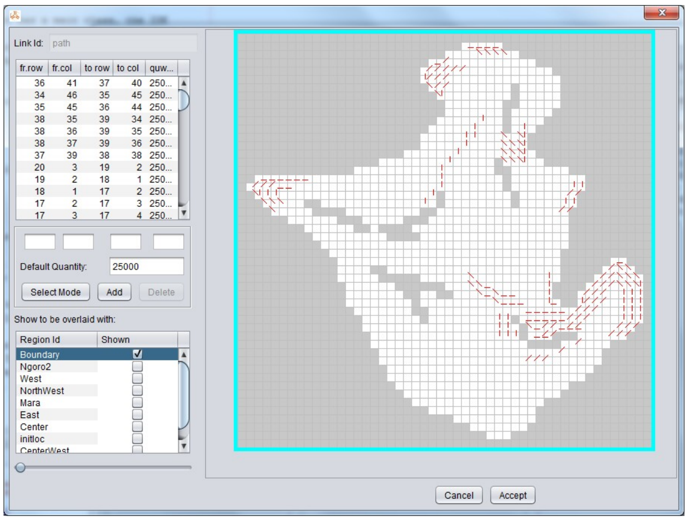
Region Id: movingspace

Component type	Parameters
region	West
region	Ngoro2
region	NorthWest
region	Mara
region	East
region	Center
region	CenterWest

Show to be overlaid with:

Region Id	Shown
Boundary	☐
Ngoro2	☐
West	☐
NorthWest	☐
Mara	☐
East	☐
Center	☐
initloc	☐

(a) editing the region



Link Id: path

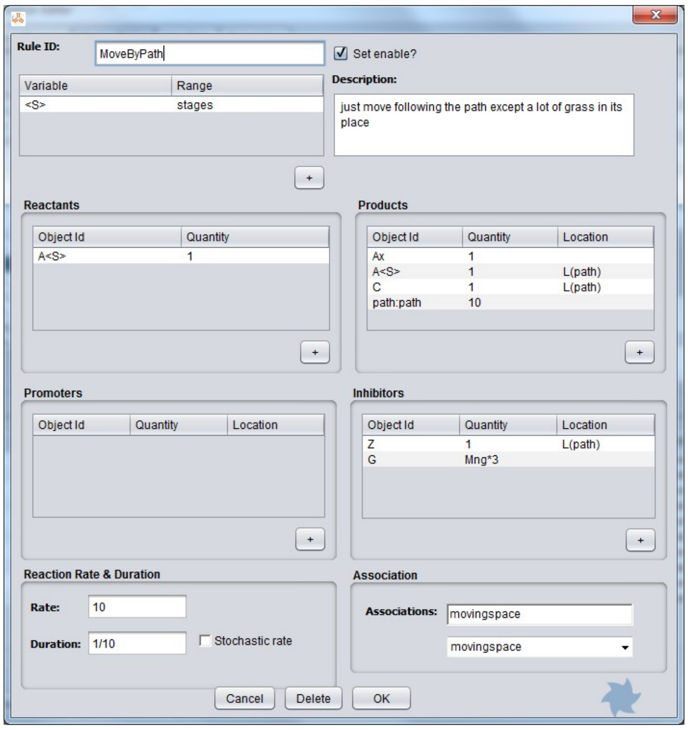
fr.row	fr.col	to row	to col	quw...
36	41	37	40	250...
34	46	35	45	250...
35	45	36	44	250...
38	35	39	34	250...
38	36	39	35	250...
38	37	39	36	250...
37	39	38	38	250...
20	3	19	2	250...
19	2	18	1	250...
18	1	17	2	250...
17	2	17	3	250...
17	3	17	4	250...

Default Quantity: 25000

Show to be overlaid with:

Region Id	Shown
Boundary	✓
Ngoro2	☐
West	☐
NorthWest	☐
Mara	☐
East	☐
Center	☐
initloc	☐
CenterWest	☐

(b) editing the initial links



Rule ID: MoveByPath ✓ Set enable?

Variable	Range
<S>	stages

Description: just move following the path except a lot of grass in its place

Reactants

Object Id	Quantity
A<S>	1

Products

Object Id	Quantity	Location
Ax	1	
A<S>	1	L(path)
C	1	L(path)
path:path	10	

Promoters

Object Id	Quantity	Location

Inhibitors

Object Id	Quantity	Location
Z	1	L(path)
G	Mng*3	

Reaction Rate & Duration

Rate: 10

Duration: 1/10 ☐ Stochastic rate

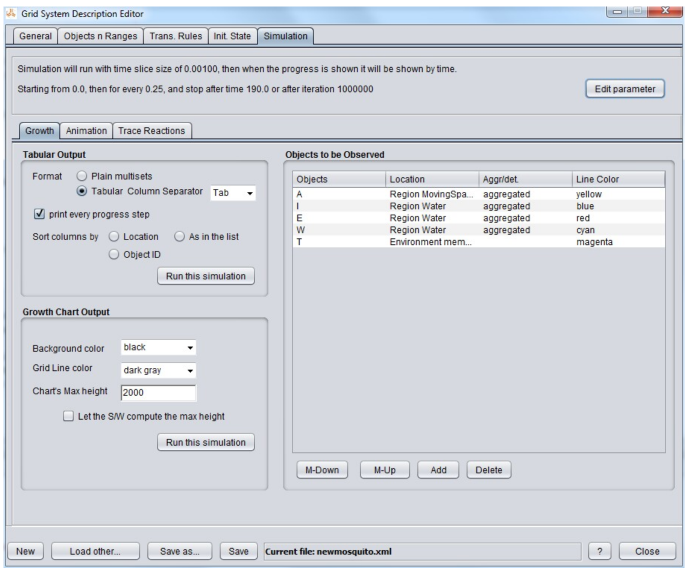
Association

Associations: movingspace

movingspace

(c) editing the rule

Figure 5.5: Screenshots of the user interfaces for editing component of the model



Grid System Description Editor

General | Objects n Ranges | Trans. Rules | Init. State | Simulation

Simulation will run with time slice size of 0.00100, then when the progress is shown it will be shown by time.

Starting from 0.0, then for every 0.25, and stop after time 190.0 or after iteration 1000000

Edit parameter

Growth | Animation | Trace Reactions

Tabular Output

Format ( ) Plain multisets [✓ (•) Tabular Column Separator] Tab

[✓] print every progress step

Sort columns by ( ) Location ( ) As in the list ( ) Object ID

Run this simulation

Growth Chart Output

Background color: black

Grid Line color: dark gray

Chart's Max height: 2000

[ ] Let the S/W compute the max height

Run this simulation

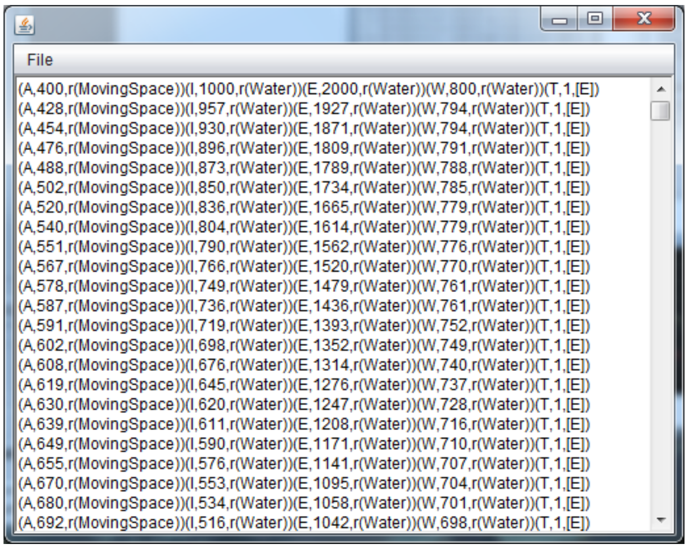
Objects to be Observed

Objects	Location	Aggr/det.	Line Color
A	Region MovingSpa...	aggregated	yellow
I	Region Water	aggregated	blue
E	Region Water	aggregated	red
W	Region Water	aggregated	cyan
T	Environment mem...		magenta

M-Down | M-Up | Add | Delete

New | Load other... | Save as... | Save | **Current file: newmosquito.xml** | ? | Close

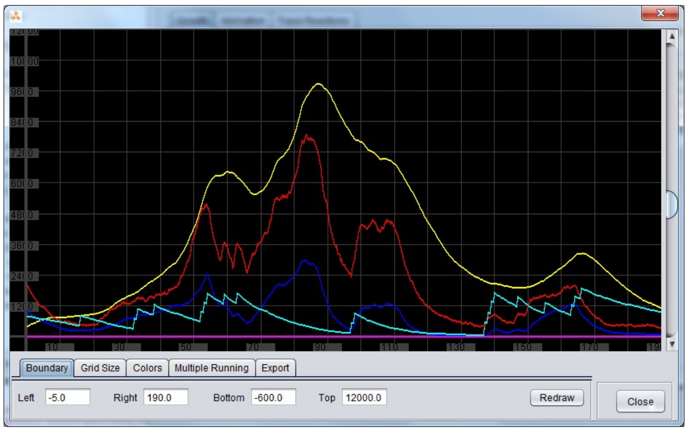
(a) setting visualisation parameters



File

```
(A,400,r(MovingSpace))(I,1000,r(Water))(E,2000,r(Water))(W,800,r(Water))(T,1,[E])
(A,428,r(MovingSpace))(I,957,r(Water))(E,1927,r(Water))(W,794,r(Water))(T,1,[E])
(A,454,r(MovingSpace))(I,930,r(Water))(E,1871,r(Water))(W,794,r(Water))(T,1,[E])
(A,476,r(MovingSpace))(I,896,r(Water))(E,1809,r(Water))(W,791,r(Water))(T,1,[E])
(A,488,r(MovingSpace))(I,873,r(Water))(E,1789,r(Water))(W,788,r(Water))(T,1,[E])
(A,502,r(MovingSpace))(I,850,r(Water))(E,1734,r(Water))(W,785,r(Water))(T,1,[E])
(A,520,r(MovingSpace))(I,836,r(Water))(E,1665,r(Water))(W,779,r(Water))(T,1,[E])
(A,540,r(MovingSpace))(I,804,r(Water))(E,1614,r(Water))(W,779,r(Water))(T,1,[E])
(A,551,r(MovingSpace))(I,790,r(Water))(E,1562,r(Water))(W,776,r(Water))(T,1,[E])
(A,567,r(MovingSpace))(I,766,r(Water))(E,1520,r(Water))(W,770,r(Water))(T,1,[E])
(A,578,r(MovingSpace))(I,749,r(Water))(E,1479,r(Water))(W,761,r(Water))(T,1,[E])
(A,587,r(MovingSpace))(I,736,r(Water))(E,1436,r(Water))(W,761,r(Water))(T,1,[E])
(A,591,r(MovingSpace))(I,719,r(Water))(E,1393,r(Water))(W,752,r(Water))(T,1,[E])
(A,602,r(MovingSpace))(I,698,r(Water))(E,1352,r(Water))(W,749,r(Water))(T,1,[E])
(A,608,r(MovingSpace))(I,676,r(Water))(E,1314,r(Water))(W,740,r(Water))(T,1,[E])
(A,619,r(MovingSpace))(I,645,r(Water))(E,1276,r(Water))(W,737,r(Water))(T,1,[E])
(A,630,r(MovingSpace))(I,620,r(Water))(E,1247,r(Water))(W,728,r(Water))(T,1,[E])
(A,639,r(MovingSpace))(I,611,r(Water))(E,1208,r(Water))(W,716,r(Water))(T,1,[E])
(A,649,r(MovingSpace))(I,590,r(Water))(E,1171,r(Water))(W,710,r(Water))(T,1,[E])
(A,655,r(MovingSpace))(I,576,r(Water))(E,1141,r(Water))(W,707,r(Water))(T,1,[E])
(A,670,r(MovingSpace))(I,553,r(Water))(E,1095,r(Water))(W,704,r(Water))(T,1,[E])
(A,680,r(MovingSpace))(I,534,r(Water))(E,1058,r(Water))(W,701,r(Water))(T,1,[E])
(A,692,r(MovingSpace))(I,516,r(Water))(E,1042,r(Water))(W,698,r(Water))(T,1,[E])
```

(b) Textual output as multisets



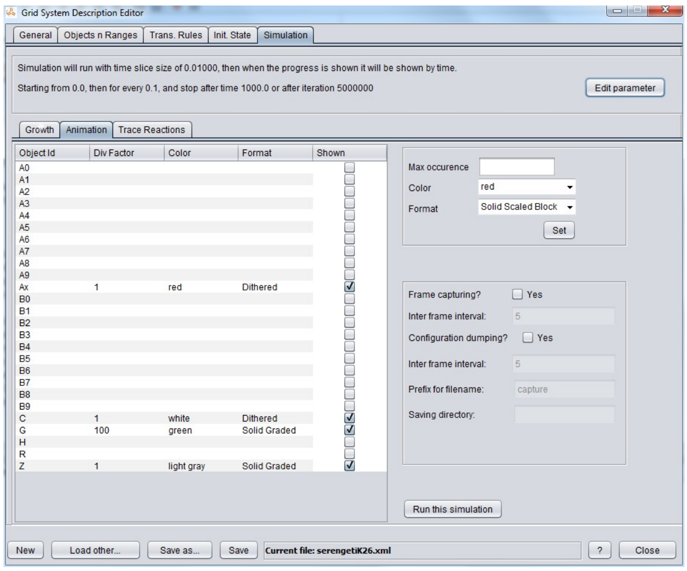
Boundary | Grid Size | Colors | Multiple Running | Export

Left -5.0 Right 190.0 Bottom -600.0 Top 12000.0

Redraw | Close

(c) Output as charts

Figure 5.6: Screenshots of the user interfaces visualising the population growth


Grid System Description Editor

General | Objects n Ranges | Trans. Rules | Init. State | Simulation

Simulation will run with time slice size of 0.01000, then when the progress is shown it will be shown by time.

Starting from 0.0, then for every 0.1, and stop after time 1000.0 or after iteration 5000000

Edit parameter

Growth | Animation | Trace Reactions

Object Id	Div Factor	Color	Format	Shown
A0				☐
A1				☐
A2				☐
A3				☐
A4				☐
A5				☐
A6				☐
A7				☐
A8				☐
A9				☐
Ax	1	red	Dithered	✓
B0				☐
B1				☐
B2				☐
B3				☐
B4				☐
B5				☐
B6				☐
B7				☐
B8				☐
B9				☐
C	1	white	Dithered	✓
G	100	green	Solid Graded	✓
H				☐
R				☐
Z	1	light gray	Solid Graded	✓

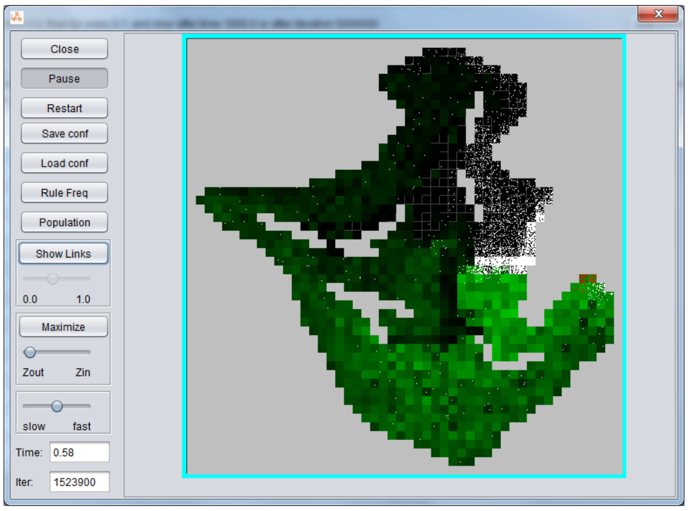
Max occurence:
Color: red
Format: Solid Scaled Block
Set

Frame capturing? ☐ Yes
Inter frame interval: 5
Configuration dumping? ☐ Yes
Inter frame interval: 5
Prefix for filename: capture
Saving directory:

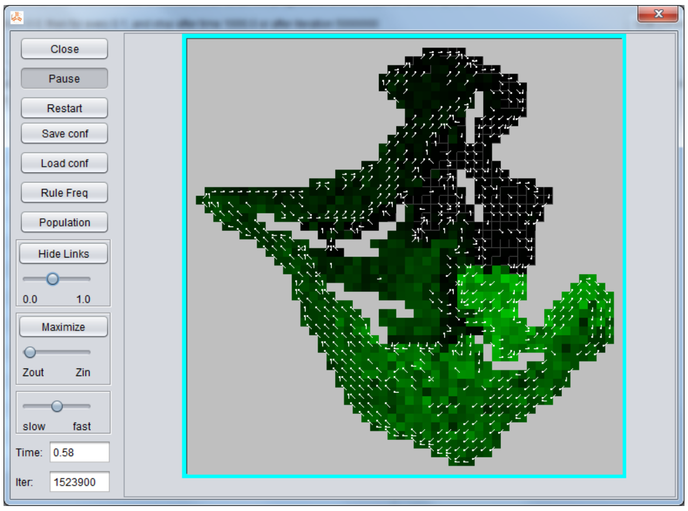
Run this simulation

New | Load other... | Save as... | Save | Current file: serengetiK26.xml | ? | Close

(a) setting animation parameters

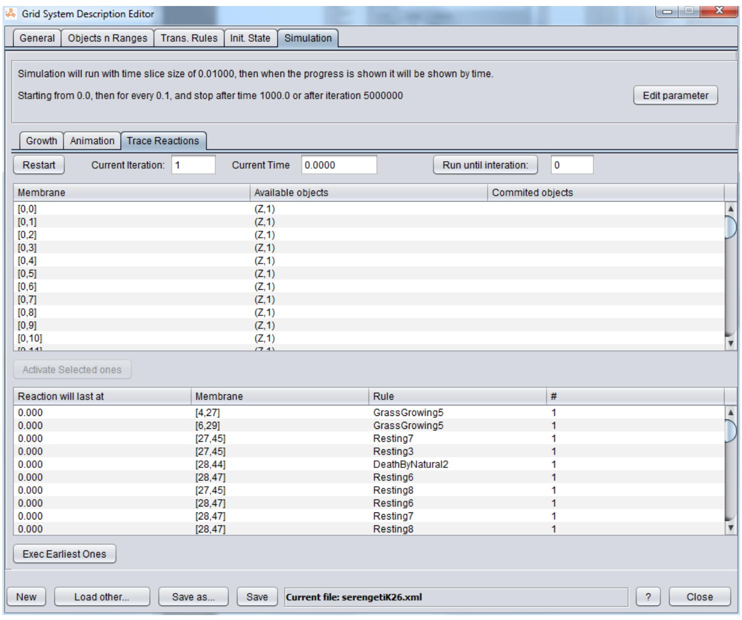


(b) controlling the animation



(c) showing the links

Figure 5.7: Screenshots of the user interfaces for animating the output



Simulation will run with time slice size of 0.01000, then when the progress is shown it will be shown by time.

Starting from 0.0, then for every 0.1, and stop after time 1000.0 or after iteration 5000000

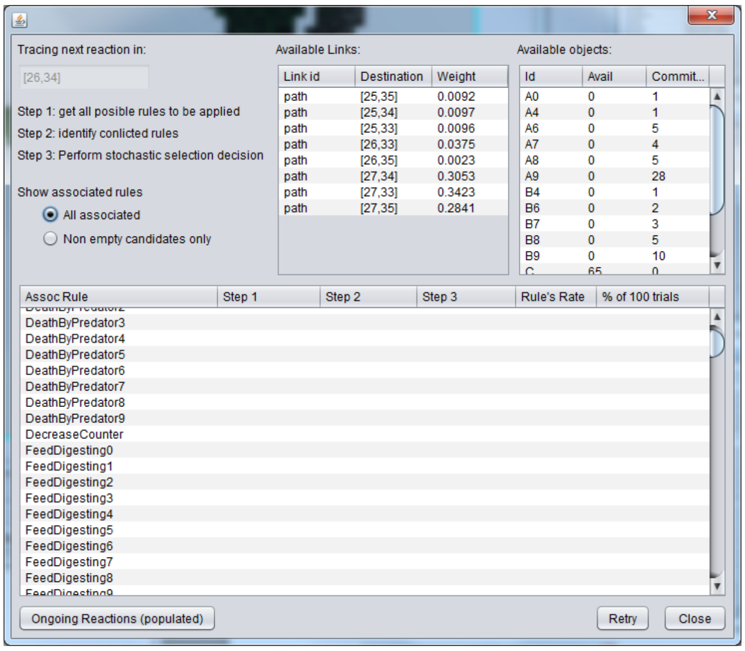
Current Iteration: 1 Current Time 0.0000 Run until interation: 0

Membrane	Available objects	Commited objects
[0,0]	(Z,1)	
[0,1]	(Z,1)	
[0,2]	(Z,1)	
[0,3]	(Z,1)	
[0,4]	(Z,1)	
[0,5]	(Z,1)	
[0,6]	(Z,1)	
[0,7]	(Z,1)	
[0,8]	(Z,1)	
[0,9]	(Z,1)	
[0,10]	(Z,1)	

Reaction will last at	Membrane	Rule	#
0.000	[4,27]	GrassGrowing5	1
0.000	[6,29]	GrassGrowing5	1
0.000	[27,45]	Resting7	1
0.000	[27,45]	Resting3	1
0.000	[28,44]	DeathByNatural2	1
0.000	[28,47]	Resting6	1
0.000	[27,45]	Resting8	1
0.000	[28,47]	Resting6	1
0.000	[28,47]	Resting7	1
0.000	[28,47]	Resting8	1

Current file: serengetiK26.xml

(a) controlling the tracer



Tracing next reaction in: [26,34]

Step 1: get all posible rules to be applied

Step 2: identify conlicted rules

Step 3: Perform stochastic selection decision

Show associated rules

[✓ All associated]

Non empty candidates only

Available Links:

Link id	Destination	Weight
path	[25,35]	0.0092
path	[25,34]	0.0097
path	[25,33]	0.0096
path	[26,33]	0.0375
path	[26,35]	0.0023
path	[27,34]	0.3053
path	[27,33]	0.3423
path	[27,35]	0.2841

Available objects:

Id	Avail	Commit...
A0	0	1
A4	0	1
A6	0	5
A7	0	4
A8	0	5
A9	0	28
B4	0	1
B6	0	2
B7	0	3
B8	0	5
B9	0	10
C	65	0

Assoc Rule	Step 1	Step 2	Step 3	Rule's Rate	% of 100 trials
DeathByPredator3					
DeathByPredator4					
DeathByPredator5					
DeathByPredator6					
DeathByPredator7					
DeathByPredator8					
DeathByPredator9					
DecreaseCounter					
FeedDigesting0					
FeedDigesting1					
FeedDigesting2					
FeedDigesting3					
FeedDigesting4					
FeedDigesting5					
FeedDigesting6					
FeedDigesting7					
FeedDigesting8					

(b) inspecting a membrane

Figure 5.8: Screenshots of the user interfaces for step-wise tracing the evolution

Chapter 6

Case Study 1: Population Dynamics and External Events

Species may be potential vectors for diseases or they may be considered as endangered. Both situations are likely to have a negative impact on the ecosystem to which these species belong. It is therefore important to develop strategies to control disease vectors and preserve endangered species. Modelling population dynamics provides a way to test and compare alternative strategies and support the process of defining and implementing policies in population control and reintroduction biology.

Aedes albopictus, also known as the Asian tiger mosquito, is a mosquito species that originated from Asia [20, 25]. This species is an important subject of study since it is well-known as a vector of some deadly pathogens, such as the West Nile virus, Yellow fever virus, St. Louis encephalitis, Dengue fever and Chikungunya fever. The *Aedes albopictus* life-cycle has 4 phases: egg, larva, pupa, and adult [25].

In its more comprehensive life cycle it is further specified in 4 larval stages and 8 adult gonothropic cycles. A gonothropic cycle contains a period in which the mosquitoes lay eggs. In this model mosquitoes have 14 stages in total in their life cycle. After obtaining a blood meal, they lay from 40 to 150 eggs. They prefer to lay eggs on the water surface in small containers like a cavity in a tree, old bucket, or tire. When the water of the rain covers the eggs, they hatch. Eggs take from 2 to 4 days to hatch. Larvae require from 7 to 10 days to become pupae, then 3 days until they develop into a standing adult. An adult can live from 4 to 8 weeks, largely depending on weather conditions. Hot, dry weather reduces life expectancy. *Aedes albopictus* is mainly found during daylight in outdoor areas. Mosquitoes can fly from several hundred yards to half a mile from their breeding spots to find a meal. They are attracted by dark clothing, perspiration, carbon dioxide and certain other smells.

Our model consists of a simple mosquito life cycle: an egg becomes an immature mosquito (first larva, then pupa), which becomes an adult. External events from the environment affect the population in forms of volume of water containers and temperature. Temperature affects the population behaviour directly by changing reaction rates. Higher temperatures will increase the rates of the transition in the life cycle. Temperature affects the population indirectly by changing the desiccation rate of water in the containers. Temperature in the environment may increase or decrease through temperature change events. Water volume in the environment may increase or decrease through rainfall events.

Spatiality will be observed in how the impact is propagated from the external events to the population in the remote locations. The diagram in Figure 6.1 shows the interactions in the ecosystem.

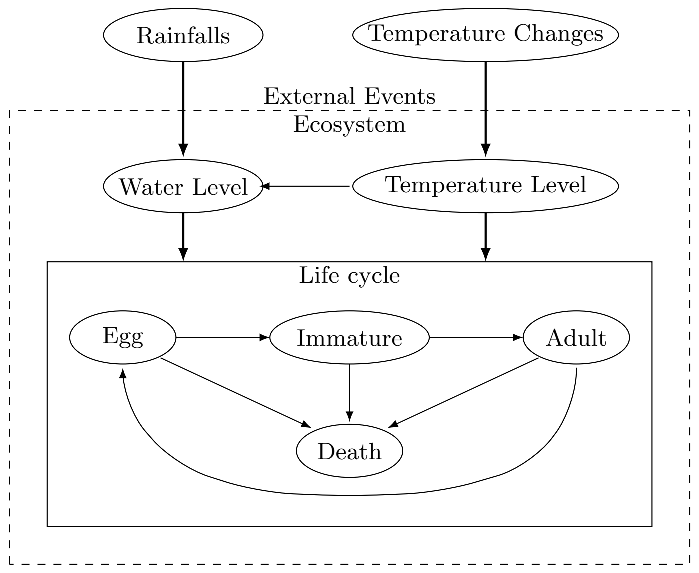


Figure 6.1: Interaction diagram between the components

6.1 The Model

The model will be described according to the structures in a Grid System. Here we only use the definition of Grid Systems without links. For compactness of the model description we will use notations described in Section 3.4.

6.1.1 Structure of the Grid

We consider a 5×5 *grid* to model the space where the mosquito population lives. However, the actual movement area is $3{\times}3$. Additional rows (top and bottom) and columns (leftmost and rightmost) are set to isolate the movement area. Access to these additional cells is denied to mosquitoes by an initial state in which such cells contain dummy objects that inhibit movement into the cell. We assume that only the central cell contains water, although we do not model how the water is distributed within the cell (e.g. uniformly or in separate containers).

Furthermore, we consider the following *regions*.

- $MovingSpace = rectangle(1, 1, 3, 3)$ (which covers $\{G_{1,1}, G_{1,2}, G_{1,3},\ G_{2,1}, G_{2,2}, G_{2,3},\ G_{3,1}, G_{3,2},\ G_{3,3}\}$), consisting of all cells that adult mosquitoes may access.
- $WaterContainer = \{G_{2,2}\}$, the central cell.
- $Boundary = rectangle(0, 0, 4, 4) \setminus MovingSpace$, consisting of the boundary cells, inaccessible to mosquitoes.

This spatial setting is depicted in Figure 6.2.

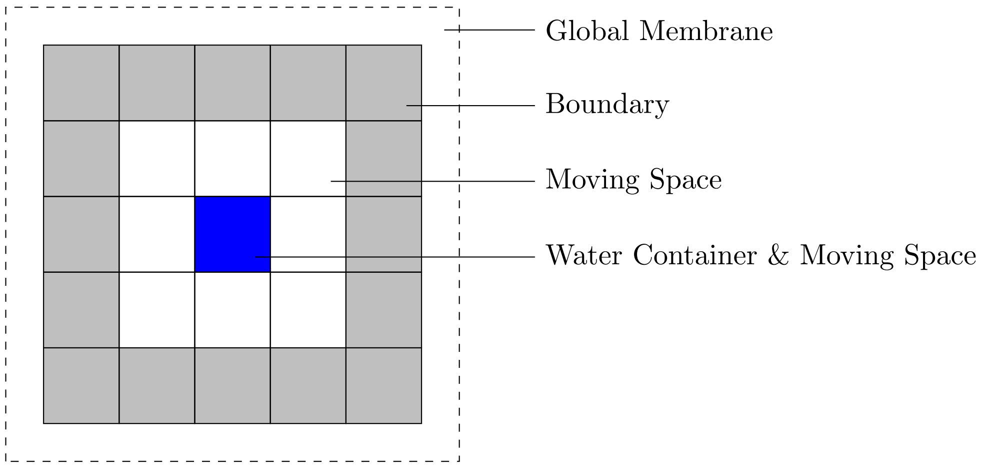


Figure 6.2: Membranes and regions of the grid

6.1.2 Objects

We use *object* E to represent eggs, I to represent immature mosquitoes and A to represent adult mosquitoes. The environment state is defined by the temperature level and the quantity of water. Temperature is represented by the number of objects T located in the global membrane and the quantity of water is represented by the number of objects W located in central cell $G_{2,2}$.

The environmental state may change depending on events that are either continuously triggered by the environmental state itself or are scheduled to occur at specific times. For instance, desiccation is a continuous event triggered by temperature and quantity of water while rainfall and temperature changes are scheduled events. Objects $Rain1$ and $Rain2$ represent light rain and heavy rain, respectively; they affect the environment by increasing the quantity of water (number of objects W). Objects $TempUp1$ and $TempUp2$ represent increases in temperature of two different scales. Objects $TempDwn1$ and $TempDwn2$ represent decreases in temperature of two different scales. They modify the number of objects T. Dummy object Z is used to be placed in boundary membranes to inhibit the movement of mosquitoes from the moving area into the boundary.. All objects are summarized in Table 6.1 and interaction among them is shown in Figure 6.3. The arrows in the figure will be represented as reaction rules that will be described in Section 6.1.4.

6.1.3 Constants

Constants are needed to express the rules in terms of templates. Some of them are listed in Table 6.2. The rest are embedded in the reaction rules. Each row of the table contains four constants. In each row the constant identification is postfixed by $\langle L\rangle$. It means that when $\langle L\rangle$ takes a value $i \in \{1, 2, 3, 4\}$, then the constant identification is postfixed by i and the constant value from respective columns will then be taken. These four sets of constants are related to the temperature intervals that will vary for some reactions. By using the constant identifications the rules of different intervals can be written as one rule template.

At the beginning of our experiments we took the constants that reflect the behaviours

Table 6.1: Components in the ecosystem as the objects in the Grid System

Object Id.	**Description**
E	eggs
I	immature mosquitoes
A	adult mosquitoes
T	temperature units
W	water volume units
$Rain1$	light rains
$Rain2$	heavy rains
T	temperature units
$TempUp1$	temperature increment
$TempUp2$	temperature double-scale increment
$TempDwn1$	temperature decrement
$TempDwn2$	temperature double-scale increment
Z	inhibiting object for blocking

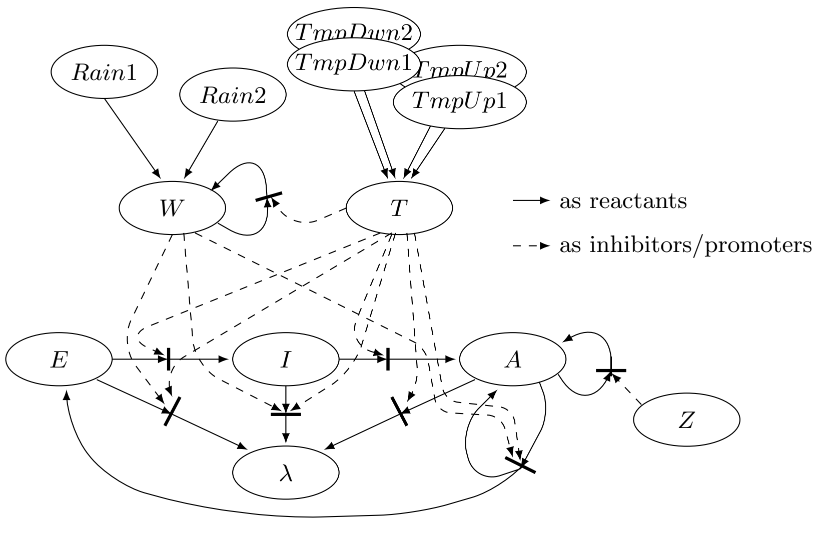


Figure 6.3: Interaction diagram between objects

as summarized in the introductory part of this chapter. The simulation was conducted followed by adjustments to those constants before the next simulation using newly adjusted constants. These simulations were repeatedly conducted until producing population growths that are close enough to the observation data that was found in nature. Figure 6.12 shows the sampling data that were taken from a real mosquito population. Figure 6.4 and Figure 6.6 show external events (temperature and rainfall, respectively) as our reference. All these data were collected in the period May–November 2009 in the province of Massa-Carrara (Tuscany, Italy).

Table 6.2: Constants for replacing the constant identifications used in rule templates

$\langle L\rangle$	1	2	3	4
$Prom\langle L\rangle$	λ	T_E	T_E^3	T_E^5
$Inh\langle L\rangle$	T_E	T_E^3	T_E^5	λ
$hatchrate\langle L\rangle$	0.3	0.4	0.45	0.5
$failrate\langle L\rangle$	0.3	0.4	0.45	0.5
$metarate\langle L\rangle$	0.1	0.2	0.5	1.0
$deathimrate\langle L\rangle$	0.2	0.25	0.3	0.35
$deathadrate\langle L\rangle$	0.4	0.45	0.5	0.55
$ovirate\langle L\rangle$	0.1	0.2	0.33	0.5
$desicrate\langle L\rangle$	0.75	1.0	1.5	2.5

6.1.4 Reaction Rules

In this section we introduce the rule templates that define the dynamics of the *Aedes albopictus* population. Dependence of rates on temperature is expressed by combining instantiations of variables $Prom\langle L\rangle$ (promoters) and $Inh\langle L\rangle$ (inhibitors), which define ranges of temperature, and instantiations of variables for rates, as shown in Table 6.2

6.1.4.1 Egg Stage

Hatched eggs become immature mosquitoes at a reaction rate which is taken from four different rates. The rate to be taken is based on the current temperature. Therefore, there are four reaction rules for this behaviour as follows.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \mathbf{Hatch\langle L\rangle} : E \xrightarrow{hatchrate\langle L\rangle,M} I \quad [Prom\langle L\rangle | Inh\langle L\rangle\,]$$

if $Hatch\langle L\rangle \in Assoc(WaterContainer)$.

Eggs failing to hatch are modelled by the following rule template, whose fail rate depends on temperature.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \mathbf{Fail\langle L\rangle} : \quad E \xrightarrow{failrate\langle L\rangle,M} \lambda \quad [Prom\langle L\rangle | Inh\langle L\rangle\,]$$

if $Fail\langle L\rangle \in Assoc(WaterContainer)$.

When the water level is extremely low (below a minimum threshold) eggs may die due to dehydration. Moreover, when the water level is extremely high (above or equal a maximum threshold) eggs may be flooded away and destroyed.

$$\begin{aligned} \mathbf{DryE}: &\quad E \xrightarrow{8.000,M} \lambda \ \ [\lambda | W^{500}\,] \\ \mathbf{FloodE}: &\quad E \xrightarrow{8.000,M} \lambda \ \ [W^{1500}\,|\lambda] \end{aligned}$$

if $DryE, FloodE \in Assoc(WaterContainer)$.

In this rule template the maximum threshold W^{500} for dehydration is modelled as an inhibitor and the minimum threshold W^{1500} for flooding is modelled as a promoter. Here rates are much higher than 0.5 and dominate the effect of $Hatch\langle L\rangle$ and $Fail\langle L\rangle$ rules above.

6.1.4.2 Immature Stage

Immature mosquitoes may become adult with a rate that depends on temperature.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \mathbf{Meta\langle L\rangle}:\quad I \xrightarrow{metarate\langle L\rangle, M} A\ \ [Prom\langle L\rangle | Inh\langle L\rangle\ ]$$

if $Meta\langle L\rangle \in Assoc(WaterContainer)$.

Immature mosquitoes failing to become adults are modelled by the following rule template, whose rate depends on temperature.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \mathbf{DeathI\langle L\rangle}:\quad I \xrightarrow{deathimrate\langle L\rangle, M} \lambda\ \ [Prom\langle L\rangle | Inh\langle L\rangle\ ]$$

if $DeathI\langle L\rangle \in Assoc(WaterContainer)$.

Moreover, immature mosquitoes may be either flooded away or dehydrated by extreme water conditions.

$$\begin{array}{ll} \mathbf{DryI}: & I \xrightarrow{8.000, M} \lambda\ \ [\lambda | W^{500}\ ] \\ \mathbf{FloodI}: & I \xrightarrow{8.000, M} \lambda\ \ [W^{1500}\ | \lambda] \end{array}$$

if $DryI, FloodI \in Assoc(WaterContainer)$.

6.1.4.3 Adult Stage

Mosquitoes may either move from one cell to one of its four adjacent cells or remain in the initial cell. Boundary cells cannot be entered.

$$\begin{array}{lll} \mathbf{Move1}: & A \xrightarrow{0.500, M} A_{(-1,+0)} & [\lambda | Z_{(-1,+0)}\ ] \\ \mathbf{Move2}: & A \xrightarrow{0.500, M} A_{(+0,-1)} & [\lambda | Z_{(+0,-1)}\ ] \\ \mathbf{Move3}: & A \xrightarrow{0.500, M} A_{(+0,+1)} & [\lambda | Z_{(+0,+1)}\ ] \\ \mathbf{Move4}: & A \xrightarrow{0.500, M} A_{(+1,+0)} & [\lambda | Z_{(+1,+0)}\ ] \\ \mathbf{Move5}: & A \xrightarrow{0.500, M} A & \end{array}$$

if $Move1, Move2, Move3, Move4, Move5 \in Assoc(MovingSpace)$.

Adults may lay eggs only in the cell containing water. We assume that every individual lays exactly 20 eggs.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \ \mathbf{Ovi\langle L\rangle}:\quad A \xrightarrow{ovirate\langle L\rangle, M} A\, E^{20}\ \ [W\ Prom\langle L\rangle\ |\ Inh\langle L\rangle\ ]$$

if $Ovi\langle L\rangle \in Assoc(WaterContainer)$.

Adult death rate depends on temperature.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \ \mathbf{DeathA\langle L\rangle}:\quad A \xrightarrow{deathadrate\langle L\rangle, M} \lambda\, [Prom\langle L\rangle | Inh\langle L\rangle\ ]$$

if $DeathA\langle L\rangle \in Assoc(MovingSpace)$.

6.1.4.4 External Events

Desiccation is a continuously occurring event that decreases the volume of water according to a desiccation factor. Such a factor depends on temperature. We assume that a desiccation event decreases the quantity of water by 4%. This is modelled in the following rule template by removing one object W out of 25.

$$\forall\langle L\rangle \in \{1,2,3,4\}.\ \textbf{Desic}\langle\mathbf{L}\rangle : W^{25} \xrightarrow{desicrate\langle L\rangle, M} W^{24}\,[Prom\langle L\rangle | Inh\langle L\rangle\,]$$

if $Desic\langle L\rangle \in Assoc(WaterContainer)$.

Scheduled events, such as rainfall and temperature changes, are defined by rules that consume a dummy object that are in the initial environmental state and produce rain or temperature change objects. In this way, if the rule deterministically spends time t to consume the dummy object, then the rain or temperature change object is produced exactly at the scheduled time t. Once the dummy object is produced, the rule that has produced it is disabled forever. For example, a high increase in temperature ($TempUp2$) at time $\langle S\rangle$ is modelled by the following rule template

$$\forall\langle S\rangle \in \mathbb{N}.\ \textbf{SchedTempUp2}\langle\mathbf{S}\rangle : TempUp2At\langle S\rangle \xrightarrow{1/\langle S\rangle} TempUp2$$

if $SchedTempUp2\langle S\rangle \in Assoc(G_E)$.

Exactly one dummy object $TempUp2At\langle S\rangle$ in the initial state enables rule $SchedTempUp2\langle S\rangle$ just one at time $\langle S\rangle$, thus producing temperature increase object $TempUp2$ at time $\langle S\rangle$.

For simplicity, we assume that each rainfall lasts 1/5 of a day (4.8 hours) and that water flows away without being collected in all cells apart from a central cell $G_{2,2}$.

$$\forall\langle V\rangle \in \{1,2\}.\textbf{SchedRain}\langle\mathbf{V}\rangle : Rain\langle V\rangle \xrightarrow{5} W_{2,2}^{\langle V\rangle\times 100}$$

if $SchedRain\langle S\rangle \in Assoc(G_E)$.

The number of objects T represents four temperature thresholds as shown by the values for promoter and inhibitor variables in Table 6.2. We model two possible decrements or increments of temperature using the following rule templates

$$\begin{array}{llll}\forall\langle C\rangle \in \{1,2\}. & \textbf{SchedTempDwn}\langle\mathbf{C}\rangle : & TempDwn\langle C\rangle\, T^{\langle C\rangle} & \xrightarrow{10} \lambda \\ & \textbf{SchedTempUp}\langle\mathbf{C}\rangle : & TempUp\langle C\rangle & \xrightarrow{10} T^{\langle C\rangle}\end{array}$$

if $SchedTempDwn\langle C\rangle, SchedTempUp\langle C\rangle \in Assoc(G_E)$.

6.2 Simulation

The simulation run for 190 time units (i.e. 190 days) to imitate the observation data (collected in the period May–November 2009 in the province of Massa-Carrara, Tuscany, Italy).

The temperature data was originally in the form of a chart ranging from 0 to 14 scale units. It was represented in the model as a sequence of events that raise or lower the temperature level as shown in Figure 6.4. As described in the model each event raises or lowers the temperature level that is represented by the number of T's. Figure 6.5 shows the

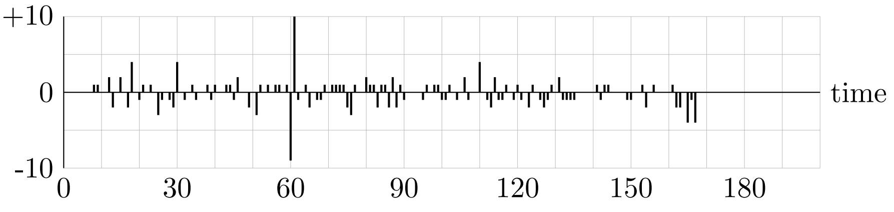


Figure 6.4: External events: temperature changes

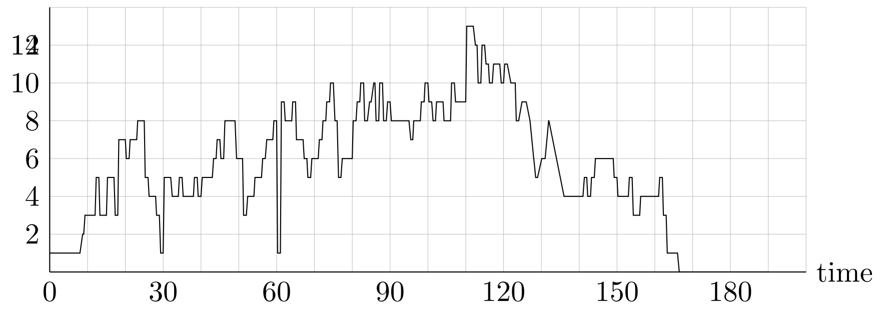


Figure 6.5: Simulated temperature level chart fluctuated by the temperature changes

chart of this temperature level during the simulation caused by the events in Figure 6.4. This chart is a reproduction of the original chart.

The rainfall data were recorded as the rainfall events. They were directly coded as events *Rain*1 and *Rain*2 representing a light rainfall and a heavy rainfall, respectively. The events are shown in Figure 6.6. There were 20 rainfall events spread over an interval of 180 days that are displayed as vertical lines. Most rainfalls are light (*Rain*1). Rainfall events and water dessiccation change the water level as seen in Figure 6.7. Note that water dessiccation has a higher rate at a higher temperature level.

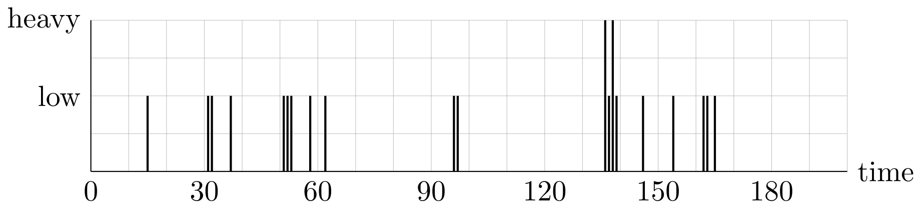


Figure 6.6: External events: rainfalls

Simulations run with some different initial objects and different parameters. Figure 6.9 shows the result from running with the following initial objects: 2000 eggs, 1000 immature mosquitoes and 400 adult mosquitoes and the parameters as written in the rules.

By running the simulation several times (using the same initial objects), it was observed that the shapes of the output curves present only small differences. Differences become smaller for larger numbers of initial objects (e.g., 4000 eggs, 2000 immature mosquitoes and 800 adult mosquitoes) and bigger for smaller numbers of initial objects (e.g., 50 eggs, 30 immature mosquitoes and 20 adult mosquitoes). Figure 6.9 shows the results from running 35 times with the parameters and initial objects as the same as the ones in Figure 6.8.

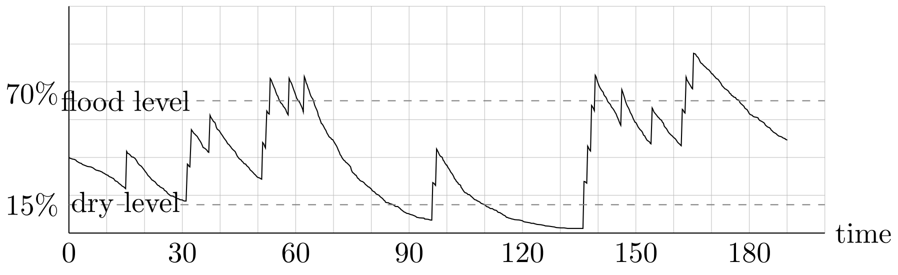


Figure 6.7: Simulated water level chart due to desiccation and rainfalls

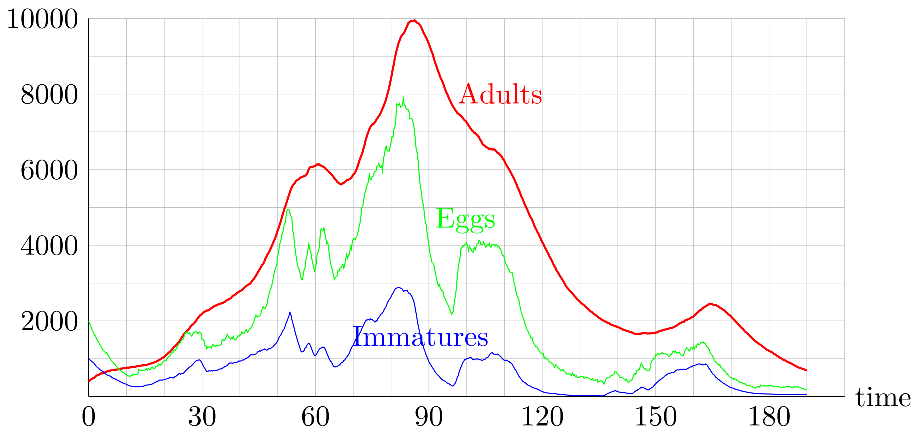


Figure 6.8: Simulated population growth: adult mosquitos, immature mosquitos and Their eggs

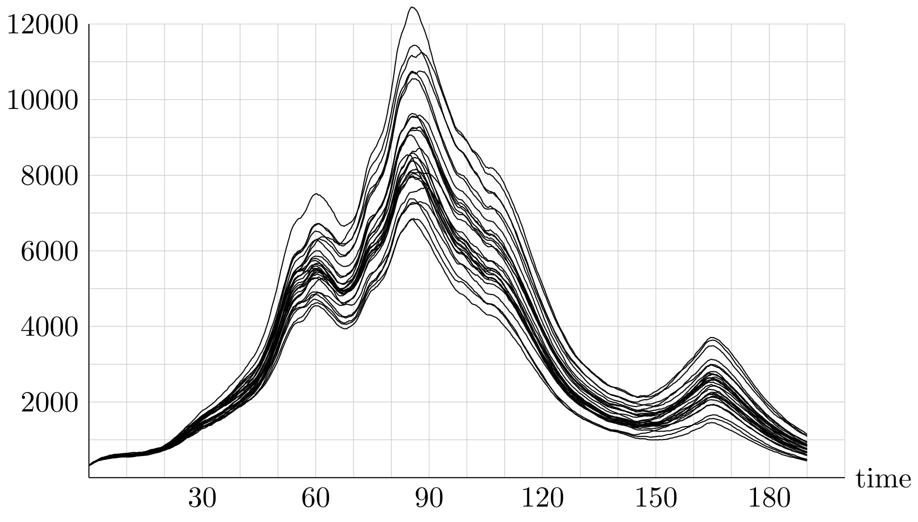


Figure 6.9: Overlaid results (total population of adult mosquitoes) from 35 times of simulation

Spatiality representation can be appreciated by considering the number of adults in each cell. Since the spatial model is very simple this number tends to be stable over time, i.e. forming a bell-shape distribution, unless extreme conditions of the environment occur. In our model the water level directly affects the number of eggs and immature mosquitoes, and consequently, after some delay, the number of adult mosquitoes. Figure 6.10 shows the growth charts of adult mosquitoes on each membrane. The fluctuations from the water containers are more abrupt than the others since the fluctuations were delayed due to distance. Figure 6.11 shows the profile of each membrane at $t = 20$ (when the population in the water container is just raised) and $t = 70$ (at the end of a decrement). From this figure we can observe that changes in surrounding membranes are delayed. Furhermore, the corner membranes are the lowest since they are the farthest from the water container. By animating the profile time to time the delays can be observed better.

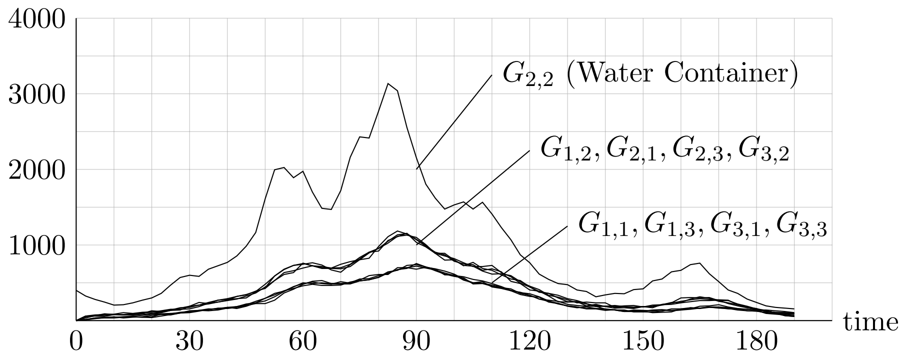


Figure 6.10: Spatial distribution of adult mosquitoes over time

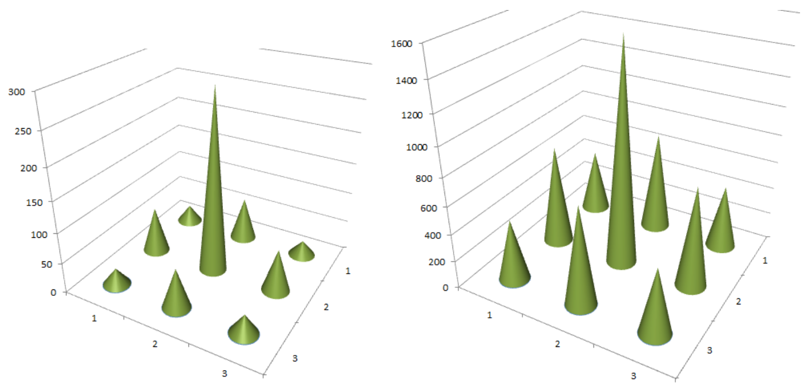


Figure 6.11: Spatial distribution of adult mosquitoes at $t = 20$ (left) and $t = 70$ (right)

Figure 6.12 shows the plot of sampling data collected during May–November 2009 in the province of Massa-Carrara using 11 CO_2 mosquito traps.

The simulation was performed by using a PC (Pentium D 3 GHz, 1 GB RAM, Windows XP SP3), for about 32 seconds. By doubling the initial population each time the simulation takes respectively 47, 70, 100, and 155 seconds.

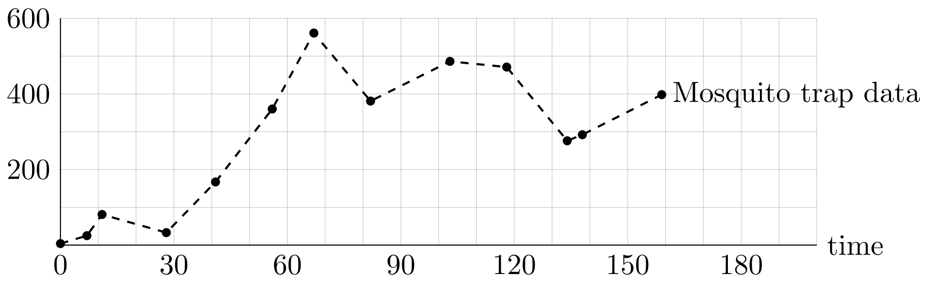


Figure 6.12: Plot of CO_2 trap data: number of adult mosquitoes

6.3 Discussion

Grid Systems support the modelling of spatiality in terms of cells in a grid. Objects may move between cells. Spatiality dynamics are obtained through varying applicability of the reaction rules over the cells. The impact of this spatiality are shown in the simulation as the impact of the events are propagated from event sources to the adult mosquitoes in the closest cell and finally to the mosquitoes in the furthest cells within time delays.

By using the syntax of Grid Systems the external events can be simply defined as the ordinary reactions. For this case the events are temperature changes and rainfalls that change the water condition. Both water and temperature levels are the factors that change the behaviours (death rates, birth rates,...) of the population. Semantically such reactions functions as the timers and the events that are triggered when the triggers are lasting. This feature is important for the modeller, providing the ability to analyse the policies given to the ecosystem as discussed as our motivation in Chapter 1.

Our model of biological aspects of the mosquito population is much simpler than the model developed by Basuki *et al.* [9], the use of spatiality addresses a wider range of practical applications. In our simple case study spatiality is used to model an isolated area (e.g., an island), from which mosquitoes cannot enter or exit. This spatiality aspect is modelled by using *boundary cells*, which do not contain mosquitoes but, instead, contain inhibitors that prevent rules from moving mosquitoes into them. A more sophisticated use of promoters and inhibitors could support the modelling of occasional or regular flows of individuals between specific areas. For example, mosquitoes may occasionally move beyond their flight range when carried inside vehicles; however, when there is considerable movement of vehicles between two specific areas (e.g., daily traffic between residential and commercial areas or through a bridge between two islands) the flow of mosquitoes becomes regular and facilitates the spreading of diseases associated with the vector.

More generally, human movement can also contribute to the spreading of diseases carried by mosquitoes, even in the absence of the movement of mosquitoes beyond their flight range. After the blood sucking of infected humans in the endemic area, moving mosquitoes turn in to the vectors that spread the disease to another area. In this sense, our approach to spatiality dynamics supports the inclusion in the ecosystem model of aspects of the interaction between the mosquito population and human population. This makes our approach suitable not only for the analysis of population dynamics but also for an epidemiological simulation.

Chapter 7

Case Study 2: Seasonal and Spatial Migration

Seasonal migration is the long-distance movement of a large number of animals belonging to one or more species that occurs on a seasonal basis. It is an important phenomenon that often has a major impact on one or more ecosystem(s). It is not fully understood how this population dynamic phenomenon emerges from the behaviours and interactions of a large number of animals. We propose an approach to the modelling of seasonal migration in which dynamics is stochastically modelled using Grid Systems, and in which spatiality is approximated by a grid of cells. We apply our approach to the migration of several wildebeest species in the Serengeti National Park, Tanzania. Our model relies on the observations that wildebeest migration is driven by the search for grazing areas and water resources, and animals tend to follow the movements of other animals. Moreover, we assume the existence of dynamic guiding paths. These paths could either be representations of the individual or communal memory of wildebeests, or physical tracks marking the land. Movement is modelled by translation rules between adjacent cells, driven by the conditions in the origin and destination cells. As conditions, we consider number of animals, grass availability, and dynamic paths. Paths are initialised with the patterns of movements observed in reality, but dynamically change depending on the variation of movement caused by other conditions. This methodology has been implemented in a simulator that visualises grass availability as well as population movement. By conducting some experiments of several combinations of parameters "seasonal migration" could be produced which imitates, to some extent, the real one.

In this case study a new feature for expressing the dynamic movement of the population around its habitat is introduced. This feature is expected to enable further analysis of the movement patterns, resulting from changes in the ecosystem. To examine this feature, the migration of wildebeests in the Serengeti National Park, Tanzania was used as our case study. The migration is massive, since it involves about 1.2 million wildebeests together with hundreds of thousands of zebras, gazelles, impalas, and other herbivores. The route typically ranges over about 1400 km from the Ngorongoro crater (in Tanzania) in the south to the Grumeti reserve in the west, then across the border to the Masai Mara reserve in the north (in Kenya) and finally back to the Ngorongoro, covering an area over 30,000 km^2. The main variables affecting the movement of the wildebeests are grass availability and the dynamic pathways that are formed by geographical boundaries

Figure 7.1: Thousands of wildebeest wind through the Masai Mara National Reserve in Kenya, Aug. 3, 2002. More than a million wildebeest annually cross the border between Tanzania and Kenya. (Pedro Ugate/Getty Images)

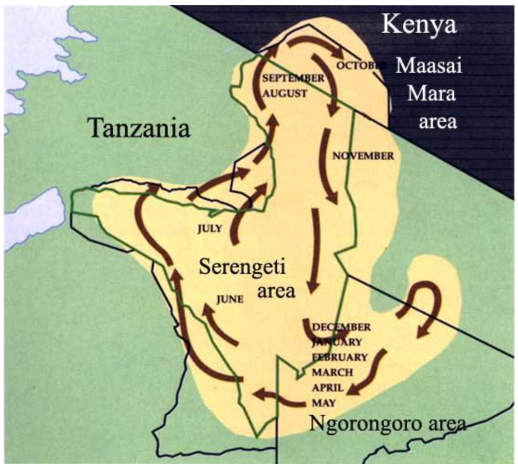
(rivers and hills) and the communal memorization of the route. Figure 7.1 illustrates the migration pathway of wildebeests. The geographical facts (topography of the land, rivers,...) may play an important role to shape the migration pathways. Figure 7.2 shows observed annual pathways and the topographic facts of the land.

7.1 The Model

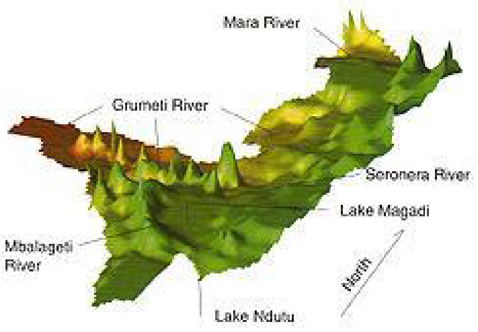
Many hypotheses have been proposed to describe the migration phenomenon. Boone *et al.* reported that at least 16 explanations have been given for the cause or timing of the migration in the Serengeti [10]. Furthermore, they have observed that the main reason driving the direction of the wildebeest migration is the search for a grazing area rather than to follow rainfall. By using evolutionary programming, Boone *et al.* have approximated a proportion of 75% to 25% for grazing area versus rainfall as the reason for the migration. By using dynamic model fitting, Holdo *et al.* report a different result, namely, an opposing rainfall and fertility gradient as the main reason for the migration [29]. The authors above go on to conclude that the rainfall affects the availability of the grass. However, the conclusions of both studies focussed on grass availability, except in the latter, rainfall also played an additional role as an external factor affecting grass availability.

7.1.1 Pathways of Migration

In our work, the primary movements will be modelled as the result of the animals finding a nearby place for grazing. This abundance of grass is categorized into three levels: very high, high, medium, and low. These movements are due to the grass in the current area that is already low, and at the same time, a nearby area where there is plenty of grass. The movements to the higher levels are modelled by the rules with the higher rates which reflect their preference. The animals start to move into a higher level place only when the level of grass is low and there is such a place close to them.



(a) Migration pattern



(b) Topography of the land

Figure 7.2: Serengeti National Park

The secondary movements are modelled as the result of the animals following existing pathways. The rules for such movements are defined with rates lower than the primary movements. The pathways are represented as links. When there are some possible directions in their area, the directions are weighted so that their preference is at the higher weights.

Additionally, there is the model for tertiary movements. This is due to denseness or quietness of the current area. The animals tend to be grouped to protect themselves from predators and therefore, they avoid being alone in a quiet location. However, it is logical that each area has a maximum capacity so that it is not a comfortable area for grazing when the area is already overcrowded with the animals. Less crowded locations are desirable to maximize their chance to access the grass. Therefore, the rules for this reasons is defined at the lowest rates.

On each movement they leave more marks (augmenting the pathway) which others will follow. The marks are the means for the animals to memorise the pathways of the previous movement. They are modelled as physical landmarks, although it could also be their cognitive ability that is communally developed. Since they are physical landmarks, they tend to fade naturally over time. In our model this fade is modelled as the marks which will decay by a factor at each time interval. In order to give a ‘momentum’ direction for the movements the initial pathways were given from the beginning.

7.1.2 Animals’ Life Cycle

Ideally, there should be at least three dimensions for their state space in the animal life cycle: age, health, and physical periods. For this experiment, we simplified them to be one dimension with 10 stages of strength/wellness from $A0, A1, ..., A9$. In every different stage animals will have their own behaviour parameters: death rates, feeding rates, birth rates. Pregnancy is limited only to stages $A6$ to $A9$. A new born baby will be at $A0$. $A9$ is the last stage and cannot be exceeded, although they can drop down a level if food is lacking. Reversely, a lack of food will lead to a drop to a lower stage except at the lowest stage. The death rate will be higher at lower stages and the birth rate will be higher to a higher stage. Their feeding rate will peak in $A6$ and $A7$. After giving birth they drop their condition by 3 stages. The transition diagram in Figure 7.3 show an animal's state change when an event occurs.

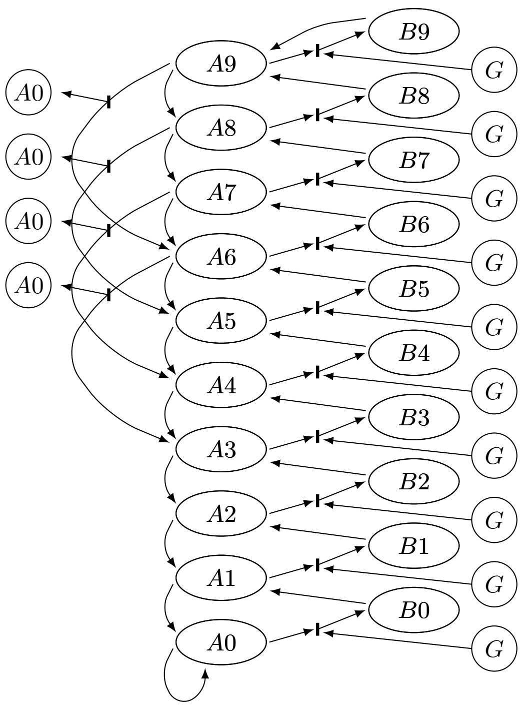
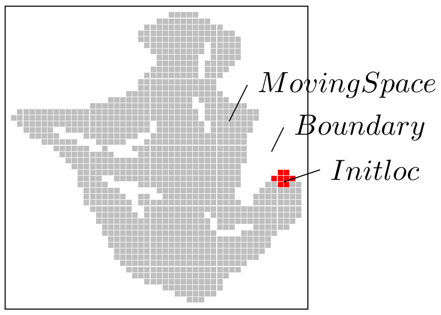


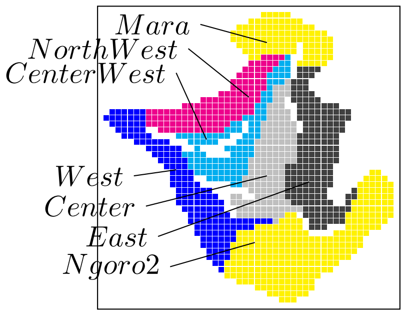
Figure 7.3: Transition change diagram of Wildebeest's life cycle (wellness model)

7.1.3 Objects

- Wildebeests: The animals of each stage will be represented as objects $A0, A1, ...,$ and $A9$, movable animals, or $B0, B1, ...,$ and $B9$, in-digestion animals. As grass is being consumed, $A\langle S\rangle$ will immediately become $B\langle S\rangle$, then will revert back to A at the higher stages, $A\langle S+1\rangle$, after digestion is complete. This differentiation is intended to avoid other rule applicability being affected by the grass which is already being consumed.
- Grass: Grass will be represented as a root R in a cell and its quantity produced in a cell as a number of G. To create a delayed effect, that forces the animal to move, the root will produce H first and after a delay, H will become G. Moreover, R can still continuously produce H.
- Counters: To ease some rules in considering the number of animals in some cells, object counter C is used to represent the number of animals in the same cell. One object C is created when a birth rule is applied. In the death rule, A will become Ax firstly in order to avoid object C becoming a reactant, and then it is followed by the rule to remove a pair of Ax and C. Moreover, the number of C changes due to the movement. Such a mechanism is performed by creating Ax in its origin and creating C in its destination.



(a) *Boundary* and *MovingSpace* are complementary; *Initloc* is a subset of *Movingspace* where wildebeests were initially located.



(b) The regions are defined as the subsets of *MovingSpace* whose different chararteristics of their vegetation (grass).

Figure 7.4: Regions defined over the cells.

- Pathways: A pathway is represented by object *path* which is a link. It will decay geometrically at a certain rate as time passes. The pathways will be refreshed by the animal movements by creating more objects *path* in the cell. The number of object *path*s created by that movement may vary depending on each movement's significance as already described in Subsection 7.1.1.

- Boundary objects: To limit the movement within an area, in each boundary cell a dummy object Z will be placed. Each movement rule will consider this object as the inhibitor in the destination cell.

7.1.4 Regions

A region is defined as a set of cells identifying an area. Associating a region and a rule implies associating its cells to the rule. Serengeti and its surrounding area are defined as 50x50 grid cells and it is divided into *MovingSpace* region and *Boundary* region. *MovingSpace* is the grazing area. It is further divided into seven regions whose grass characteristics are different: *Ngoro*2, *West*, *Center*, *East*, *Mara*, *Northwest*, and *Centerwest*. They are characterized by the grass growth duration and by the maximum grass quantity that can be produced by the land. *Initloc* is the region where the animals have initially been placed. The regions are shown in Figure 7.4.

7.1.5 Reaction Rules

Firstly the rules were written and their parameters and initial objects were just roughly given. After running several combinations some adjustments were made. Insignificant rules were removed, except the ones representing important behaviours which increased: increasing their rates or lengthening/shortening their durations. Their death/birth rates were also adjusted to approximate reasonable actual death/birth rates and represent no more than 25% of the total population in one cycle. The initial pathways were given as little as possible avoiding being trapped at the corners. Grass growth parameters were set to assure the right direction of migration. Ideally the time frame was set up before defining parameters. However, the definition of "one year" was eventually justified based on the result. Then, the migration cycle time length resulted from the simulation which was taken as "one year" and then the parameters were re-adjusted based on this. This method was used to repeatedly work through many simulation runs until a reasonable one was produced. The rules and parameters shown below illustrate one example that resulted in a stable migration cycle.

In specifying the rules the following constants are listed in Table 7.1 and Table 7.2. Also, the rules refer to the following identification.

$$\begin{array}{lcl} stages & = & \{0,1,2,..,9\} \\ directions & = & \{1,2,..,8\} \\ declev & = & \{1,..,10\} \\ Mnp & = & 5 \\ Mng & = & 5 \end{array}$$

All pairs $(dr\langle x\rangle, dc\langle x\rangle)$, where $\langle x\rangle = 1,..,8$, are the relative pointers to 8 neighbouring cells as their values are listed in Table 7.3.

Table 7.1: Constants table for rules of animal's life cycle

stages	$\langle S\rangle$	0	1	2	3	4	5	6	7	8	9
upstages	$ug\langle S\rangle$	A1	A2	A3	A4	A5	A6	A7	A8	A9	A9
downstages	$dg\langle S\rangle$	A0	A0	A1	A2	A3	A4	A5	A6	A7	A8
birthstages	$bs\langle S\rangle$	-	-	-	-	-	-	A3	A4	A6	A7
birth rates	$br\langle S\rangle$	-	-	-	-	-	-	0.0017	0.0033	0.005	0.008
death rates (nat.)	$dn\langle S\rangle$	0.05	0.035	0.021	0.015	0.011	0.008	0.006	0.004	0.003	0.002
death rates (prey)	$dp\langle S\rangle$	0.5	0.35	0.21	0.15	0.11	0.07	0.005	0.003	0.03	0.015
feeding rates	$fr\langle S\rangle$	1	2	4	6	8	10	15	20	16	0.1

Table 7.2: Constants table For grass growth rates

Region	$\langle R\rangle$	Ngorongoro	West	Center	East	Mara	Northwest	Centerwest
Max p. cell	$mg\langle R\rangle$	100	100	30	100	100	100	20
Duration est.	$gr\langle S\rangle$	1	1	15	1	1	1	15

Table 7.3: Constants table for pointers of pathways

$\langle x\rangle$	1	2	3	4	5	6	7	8
$dr\langle x\rangle$	-1	-1	-1	0	0	1	1	1
$dc\langle x\rangle$	-1	0	1	-1	1	-1	0	1

Consuming the grass, A becomes B,
$\forall\langle S\rangle \in stages.$ **Feeding⟨S⟩** : $G \;\; A\langle S\rangle \xrightarrow[1/4,M]{fr\langle S\rangle} B\langle S\rangle$
if $Feeding\langle S\rangle \in assoc(MovingSpace)$.

Food digesting is needed to create delay after the grass is removed and the wellness level increases by one level except at $A9$,
$\forall\langle S\rangle \in stages.$ **Digesting⟨S⟩** : $B\langle S\rangle \xrightarrow{1,M} ug\langle S\rangle$
if $Digesting\langle S\rangle \in assoc(MovingSpace)$.

Being unable to feed because of grass shortage, the wellness level decreases by one stage except at $A0$,

$\forall\langle S\rangle \in stages.$ **Starving⟨S⟩** : $A\langle S\rangle \xrightarrow{1/2,M} dg\langle S\rangle \quad [\ \lambda \quad |\ G^{Mng/2}\ ]$
if $Starving\langle S\rangle \in assoc(MovingSpace)$

Time needed to balance the death propensities,

$\forall\langle S\rangle \in stages.$ **Resting⟨S⟩** : $A\langle S\rangle \xrightarrow[1/2,M]{1} A\langle S\rangle$
if $Resting\langle S\rangle \in assoc(MovingSpace)$.

Giving birth only at stages $A6, A7, A8, A9$ with different birth rates; after delivering a baby (at state $A0$) its wellness level decreases three levels,

$\forall\langle S\rangle \in stages.$ **Birth⟨S⟩** : $A\langle S\rangle \xrightarrow[5,M]{br\langle S\rangle} bs\langle S\rangle\ \ A0\ \ C$
if $Birth\langle S\rangle \in assoc(MovingSpace)$.

Mortality because of natural factors; more healthy having a lower death rate,

$\forall\langle S\rangle \in stages.$ **DeathNat⟨S⟩** : $A\langle S\rangle \xrightarrow[0.5,M]{dn\langle S\rangle} Ax \quad [\ C^{Mnp} \quad |\ \lambda\ ]$
if $DeathNat\langle S\rangle \in assoc(MovingSpace)$.

Mortality because of predators and natural causes due to grazing in a quiet place; the rate is higher than normal,

$\forall\langle S\rangle \in stages.$ **DeathPred⟨S⟩** : $A\langle S\rangle \xrightarrow[0.1,M]{dp\langle S\rangle} Ax \quad [\ \lambda \quad |\ C^{Mnp}\ ]$
if $DeathPred\langle S\rangle \in assoc(MovingSpace)$.

Decreasing counter C after being mortality,

DecCount : $Ax\ \ C \xrightarrow{\infty} \lambda$
if $DecCount \in assoc(MovingSpace)$.

Initial growth of grass; root R produces one unit of grass H until the maximum capacity that the land can produce is reached,

$\forall\langle R\rangle \in regions.$ **Grass⟨R⟩** : $R \xrightarrow[gr\langle R\rangle,M]{1} R\ \ H \quad [\ \lambda \quad |\ G^{mg\langle R\rangle}\ ]$
if $Grass\langle R\rangle \in assoc(reg\langle R\rangle)$.

Grass growing to be available for future grazing,

GrassReady : $H \xrightarrow[25,M]{1} G\ \ \lambda$
if $GrassReady \in assoc(MovingSpace)$.

Movement along the path unless there is enough grass for grazing,

$\forall\langle S\rangle \in stages.$ **MoveByPath⟨S⟩** :
$A\langle S\rangle \xrightarrow{10,M} Ax\ \ A\langle S\rangle_{path}\ \ C_{path}\ \ path^{10}{:}path \quad [\ \lambda \quad |\ Z_{path}\ \ G^{7\times Mng}\ ]$
if $MoveByPath\langle S\rangle \in assoc(MovingSpace)$.

Random movement to a place with plenty of grass (3, 6, and 9 times minimum quan-

tity),
$\forall\langle G\rangle \in \{3,6,9\}, \forall\langle X\rangle \in \mathit{directions}, \forall\langle S\rangle \in \mathit{stages}.$ **MoveToGrass⟨G⟩⟨X⟩⟨S⟩** :

$$A\langle S\rangle \xrightarrow[1/2,M]{mr\langle G\rangle} Ax \;\; A\langle S\rangle_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; C_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; path^{pn\langle G\rangle} : (dr\langle X\rangle, dc\langle X\rangle)$$
$$[\;\; G^{Mng\langle G\rangle}_{(dr\langle X\rangle,dc\langle X\rangle)} \;\;|\;\; G^{Mng} \;\; Z_{(dr\langle X\rangle,dc\langle X\rangle)} \;\;]$$

if $MoveToGrass\langle G\rangle\langle X\rangle\langle S\rangle \in assoc(MovingSpace)$, and $(mr\langle x\rangle | \langle x\rangle = 3,6,9) = (20,25,35)$, and $(pn\langle x\rangle | \langle x\rangle = 3,6,9) = (10,25,50)$.

Movement to a cell with a less dense grouping of animals due to overcrowding,
$\forall\langle X\rangle \in \mathit{directions}, \forall\langle S\rangle \in \mathit{stages}.$ **MoveToLessDens⟨X⟩⟨S⟩** :

$$A\langle S\rangle \xrightarrow[1/2,M]{4} A\langle S\rangle_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; C_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; Ax \;\; path^{7} : (dr\langle X\rangle, dc\langle X\rangle)$$
$$[\;\; C^{10Mnp} \;\; C^{Mnp}_{(dr\langle X\rangle,dc\langle X\rangle)} \;\;|\;\; C^{5\times Mnp}_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; Z_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; path \;\;]$$

if $MoveToLessDens\langle X\rangle\langle S\rangle \in assoc(MovingSpace)$.

Movement to a cell with a more dense grouping of animals due to quietness,
$\forall\langle X\rangle \in \mathit{directions}, \forall\langle S\rangle \in \mathit{stages}.$ **MoveToMoreDense⟨X⟩⟨S⟩** :

$$A\langle S\rangle \xrightarrow[1/2,M]{8} A\langle S\rangle_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; C_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; Ax \;\; path^{7} : (dr\langle X\rangle, dc\langle X\rangle)$$
$$[\;\; C^{2Mnp}_{(dr\langle X\rangle,dc\langle X\rangle)} \;\;|\;\; C^{Mnp} \;\; Z_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; C^{10\times Mnp}_{(dr\langle X\rangle,dc\langle X\rangle)} \;\; path \;\;]$$

if $MoveToMoreDense\langle X\rangle\langle S\rangle \in assoc(MovingSpace)$.

Decaying of 10% per interval of time (7 time units),
$\forall\langle P\rangle \in \mathit{declev}.$ **Decay⟨P⟩** :

$$path^{\langle P\rangle} \xrightarrow[7,M]{1} path^{\langle P\rangle-1} \quad [\;\; \lambda \;\;|\;\; path^{ip\langle P\rangle} \;\;]$$

if $Decay\langle P\rangle \in assoc(MovingSpace)$, and $(ip\langle x\rangle \mid \langle x\rangle = 1,..,10) = (2,..,9,10,0)$.

7.1.6 Initial Configuration

7.1.6.1 Initial population size and location

A total population of 2800 wildebeests is initially placed evenly in the cells of *Initloc* region. They were distributed in proportion to their wellness/strength stage, as shown in Table 7.4.

Table 7.4: Initial numbers of animals according to their levels

Level	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	Total
Number in a cell	10	10	15	25	30	50	60	75	50	25	350
Number in all cells	80	80	120	200	240	400	480	600	400	200	2800

7.1.6.2 Grass and Grass Root

The numbers were set according to the figures from previous migration. Each region has a different duration in producing grass available for grazing. Grass growth is bounded to a maximum quantity per cell which is assumed as being caused by the different condition of the soil and water in each region. The following are the initial quantities of grass G given to each per cell: *Ngorongoro* 50 (50%), *West* 40 (40%), *Northwest* 20 (20%), *Mara* 20 (20%), *East* 2 (20%), *Center*12 (40%), and *Centerwest* 2 (10%).

7.1.6.3 Initial Pathways

Initially some pathways were placed in some parts of areas, especially at the corners (*West*, *Mara* and *East*, to avoid being isolated at the corners), at turning positions (*East* to the tail of *Ngoro*2), and at the *Center* (to force them to go downwards). Visually the initial pathways are shown in Figure 7.6(a).

7.2 Simulation

A simulation was run producing several stable cycles of migration. Figure 7.5 shows each "month" in the second cycle. In the fourth cycle there is a small group of animals which were trapped in the tail part of Ngorongoro. They were misled since the initial pathways were already overwritten by the new pathways created later. As more groups were being trapped, the size of the overall population declined. By increasing the decay interval from 7 time units to 20 (to sustain the paths) there were no trapped groups until 7 cycles later.

The movement of frontiers were affected by the grass and movement of the followers were affected by the pathways. As the followers moved forward they become the next frontiers. Figure 7.6 shows the pathways in the beginning ($t = 0$) and at about the end of the second cycle ($t = 17$).

There are some interesting observations from the experiments by changing the rule parameters or adding/removing some other rules.

- When there was no initial pathway, the animals were blocked by each other at the corner creating isolated groups. They stayed there until the grass regrew or just died. Then, the population size declined faster since in cells where isolated groups were located there was no more grass.
- When the initial pathways were given in a small quantity, a similar isolated situation happened after those initial pathways decayed away. The population size also declined sooner.
- When the grass characteristics were made equal for all regions, the animals were divided into small groups moving without a pattern in the area. Over time, some groups blocked each other when they met from opposite directions. As a result the overall population size declined.
- When the rate of rule MoveToPath was made exceeding the rate MoveToGrass, the migration went faster and the population was spread along the migration route and the average of their wellness dropped, increasing the death rate.
- When initial grass quantities were made much smaller and the migration had not been achieved, the animals spread themselves over the area and the overall population size immediately declined much faster.
- When the rate of random movement was set higher the animals were evenly spread throughout an area and there was no significant migration. Since the death rate for isolated animal was high the population size immediately declined.

7.2.1 Results: Migration Pathways

The pathways created during the migration (at the end of the second cycle) are shown in Figure 7.6(b). They can be compared to the initial path in Figure 7.6(a). In each cell the animals passed through pathways which were created in many directions. However, the Figure shows only the pathway whose weighting is greater than 0.30. Figure 7.6(b) shows that most initial paths were still there. After the fourth cycle the most original pathways were overwritten by the new pathways.

7.3 Discussion

As described previously, the model was simplified in some aspects compared to the real situation. The state space for life-cycle could be more complex than our ten-stage life cycle. The topography of the area was not fully represented. The parameters were not based on real-world measurements. The external events (rainfalls, temperature) and water availability which may affect the animal movements were not included. The resulting migration pattern has not been compared with actual migration patterns. However, this modelling work was carried out with an experimental purpose rather than as a theoretical work in the biological domain. All parameters and behaviours expressed as the rules need to be further validated by biologists. The main motivation for working on this case was to explore a new feature of Grid Systems. As seen in this case, its modelling required us only to express the behaviours in terms of rules and parameters. Then, the simulator can preview the result by evolving the model. A working prototype simulator was developed and used in our work, which was developed based on the semantics of Grid Systems.

A new feature of Grid Systems is introduced. The new feature is the 'link' which is a special object that can carry pointers. The pointers carried by the links make it possible to model the pathways for population movement in a more dynamic way. To show how this was carried out, a case was modelled utilizing the feature and applied using a simulator. The case study took a wildebeest population in the Serengeti National Park, which performs seasonal migration phenomena around the park area, as its subject. A simulation of the model imitated the migration.

A future model could be made using real data for the same case or using other cases with similar problem characteristics, namely, modelling the spread of epidemic diseases.

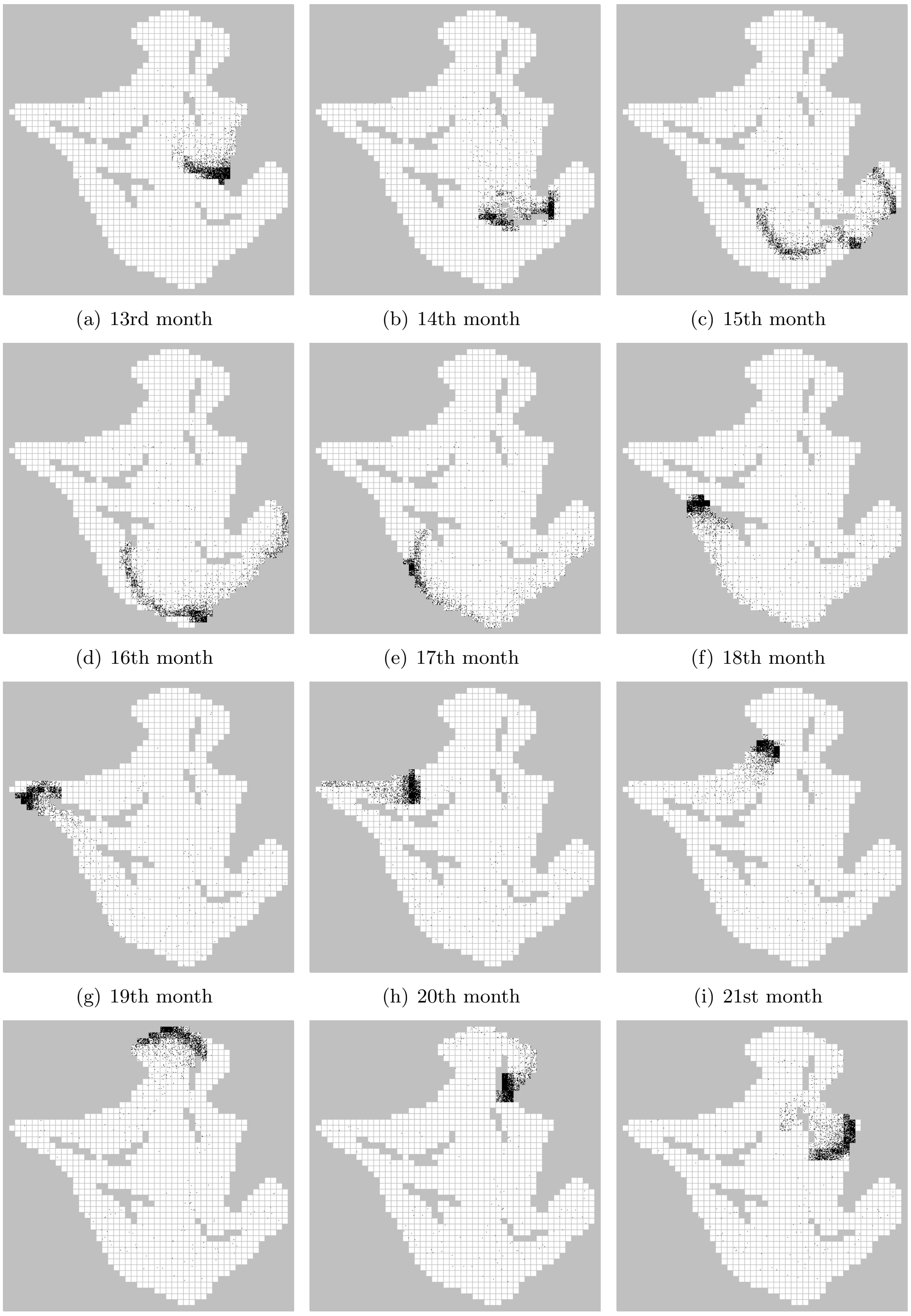
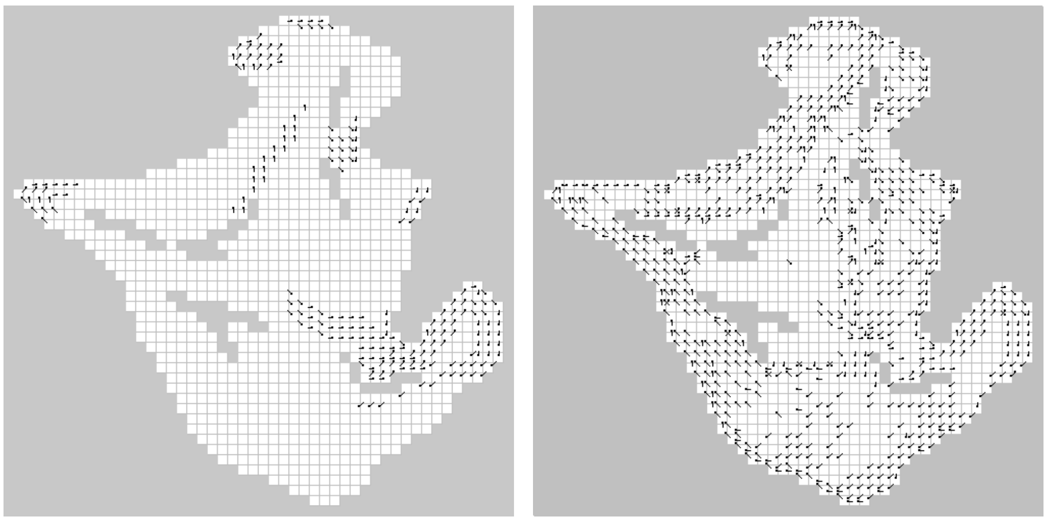

(a) 13rd month
(b) 14th month
(c) 15th month
(d) 16th month
(e) 17th month
(f) 18th month
(g) 19th month
(h) 20th month
(i) 21st month
(j) 22nd month
(k) 23th month
(l) 24th month

Figure 7.5: "Monthly" sequence of migration at the second cycle since having started.



(a) $t = 0$ (b) $t = 170$

Figure 7.6: Initial pathways (at $t = 0$) and pathways at $t = 170$ (about the end of the second cycle). There are more pathways actually but it shows only the ones whose weights are greater than 0.3.

Chapter 8

Conclusions and Future Works

Being motivated by the intention of the world to promote sustainable developments, especially ecological sustainable developments, a new formalism is constructed and it has been the main substance of this thesis. The formalism is aimed at developing models of ecosystems. Then, by modelling and analysing the ecosystem, the properties of this ecosystem can be studied. Furthermore, the impacts of given policies to the populations inside the ecosystem can be evaluated by putting the policies into the model.

Our formalism is constructed by considering several aspects that usually exist both in the ecosystem and in each population inside the ecosystem. The technical aspects are spatiality, stochasticity, and parallelism. Spatiality is important since the ecosystems are generally located in a wide range area that may vary the behaviours of the populations. Stochasticity is needed due to some phenomena that are unknown or cannot be modelled as simple behaviours. Parallelism is important since ecosystems consist of components that interact simultaneously with one another.

The other aspect is related to the usability of the formalism from the point of view of the modellers who are generally biologists. The syntax and the semantics of the formalism have been formally defined. The models can be written according to the syntax, and the analytical tools can be constructed based on the semantics. The usability aspect requires the formalism to enable modellers to express the behaviours in a more intuitive way.

In our study of existing formalisms, there are some formalisms that provide some of these required aspects, however, there is no single formalism that provides all these aspects. Therefore, we have constructed a new formalism that adopts the features from some existing formalisms to fulfil the required aspects. The major features were taken from Cellular Automata and P Systems. The spatiality that is provided by the grid structure of the CA was combined with the parallelism of reactions of P Systems by considering the cells of CA as the membranes of P Systems. It is then called Grid Systems to preserve these inspiring ideas.

By the idea of the grid we can consider the habitat area divided into membranes (cells) representing a reasonably small area. By this division into membranes, the behaviours of the components of the ecosystem can vary from one another. Furthermore, the distribution of the populations over the area can be modelled using an object tabulation of the membranes in terms of multisets. The behaviours of the components are defined as the reaction rules that were derived from the idea of the evolution rules in P Systems. The stochasticity of Gillespie's SSA was applied in order to resolve the non-deterministic decision to select reactions from all possible ones.

With combination ideas of CA and P Systems, Grid Systems can be seen as an extension of CA where each cell is represented by a multiset over objects and the behaviour in each cell is defined by transition rules. Grid Systems can be seen as a modification of P Systems where the membranes are in the grid structure instead of in a hierarchical structure. The grid structure of membranes enables the transposition of objects between membranes in a more direct way by referring to the destined membrane's actual/absolute address or relative address.

Besides this, the reactions in Grid Systems are enriched by an additional property, reaction duration, which can be either a fixed or an exponentially distributed value. Taking advantage of the idea of duration, external events can be defined in the model without introducing any *ad-hoc* processes to the semantics. Therefore, in Grid Systems the reaction rules can be defined to incorporate reaction rates and reaction stochastic durations or remain as step-wise reaction rules as in P Systems.

How these features can express various behaviours has been shown by modelling a mosquito population as in the first case study. The external events were defined to affect the population. The environmental temperature affects the water level and the population behaviours. The water level affects the life of eggs/pupae/larvae. Through simulating the model, the overall impact caused by the external events was propagated after certain delays from one another and from the mosquitoes closest to the water and those farthest away.

Other than the above features, Grid Systems have a new feature called 'links'. A link is an object that can carry one or more pointer. Through the links, the membranes can be linked logically, but also dynamically, forming a higher level structure, like trees/graphs over the actual grid structure. Through this feature basic-form P Systems can be simulated well by Grid Systems. The links are also useful for representing the population's movement pathways so that the seasonal migration of some species can be modelled as a Grid System. This has been shown in our second case study concerning the phenomenal seasonal migration of wildebeests in the Serengeti National Park, Tanzania.

From the case studies we observe that all specified aspects have been accommodated in Grid Systems. In order to anticipate the aspects other than those, further research work regarding Grid Systems' relationships to other formalisms were also conducted. The formalisms are Turing Machines, CA, and P Systems. In this work we translated some models that were created by those formalisms into Grid Systems. For this work we summarized that the models could be completely translated into Grid Systems. This could be the initial work towards our future theoretical works regarding Grid Systems. From these works all models of ecosystems built, based on the formalisms, can also be accommodated by Grid Systems.

The implementation issues of the analytical tool for Grid Systems were also discussed. A brief description of our prototype of the analytical tool was presented as an illustration of the discussion. The prototype works well even though it is still a working prototype. The case studies were developed and analysed by using this tool. This has also led us to conclude that the evolution algorithm, which represents the semantics, is already well defined.

Further research could continue on from our work considering relationships to other formalisms; for example, Markov Processes, Petri-Nets, and various rewriting systems. More theoretical work could be conducted in order to prove more properly the compatibilities of Grid Systems to these formalisms. Regarding to its expressiveness, the formalism of Grid Systems is a quite high-level modelling language. Some models constructed as grid systems could be mathematically intractable. For such models simulative approaches is more suitable instead. However, further investigation of the classes of Grid Systems and their related well-known formalism, such as CTMCs, would be so beneficial. Their well-known properties will help the modellers to analyse the models that they are developing. Such investigations are left for possible future works.

Besides this, it would be interesting to consider the structures of objects, like the ones in CLS (Calculus of Looping Sequences), in Grid Systems instead of just in multisets. Some modelling cases, such as animal nesting tend to be formed in certain structures.

The expressiveness of Grid Systems should be further verified with the involvement of biologists. This would be confirmation of the usability of Grid Systems from the point of view of the modellers. Wider application to various case studies may contribute to the improvement of Grid Systems. Besides ecosystems, Grid Systems have potential to be used for other stochastic modelling, namely weather forecasting problems, global warming studies, global financial dynamics, and so on.

More professional tools could be developed, including the implementation on parallel (multi-core) computing systems including the grid/cluster computing systems. Its scalability could be raised by implementing better data structures in order to handle the tables and to properly select number representations. In the future, it could be possible to implement a special multi-core architecture which is designed specifically for Grid Systems (just like the GPU for computer graphics).

Bibliography

[1] Our Common Future, chapter 2: Towards Sustainable Development. World Commision on Environment and Development, 1987.

[2] Rio Declaration on Environment and Development. United Nations Conference on Environment and Development (UNCED), Rio de Janeiro, Brazil, 1992.

[3] A. Adamatzky. *Identification of Cellular Automata.* Taylor and Francis, London, 1994.

[4] H. Balzter, P. W. Braun, and W. Köhler. Cellular automata models for vegetation dynamics. *Ecological modelling*, 107(2):113–125, 1998.

[5] S. Bandini, S. Manzoni, H. Umeo, and G. Vizzari, editors. *Cellular Automata - 9th International Conference on Cellular Automata for Research and Industry, ACRI 2010, Ascoli Piceno, Italy, September 21-24, 2010. Proceedings*, volume 6350 of *Lecture Notes in Computer Science*. Springer, 2010.

[6] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Cerone, and S. Setiawan. Modelling population dynamics using grid systems. *MoKMaSD 2012, Thessaloniki, Greece*, 2012.

[7] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, and A. Rama. A process calculus for molecular interaction maps. *Membrane Computing and Biologically Inspired Process Calculi (MeCBIC) 2009*, pages 35–49, 2009.

[8] M. Barrio, K. Burrage, A. Leier, and T. Tian. Oscillatory regulation of hes1: Discrete stochastic delay modelling and simulation. *PLoS Computational Biology*, 2(9), 2006.

[9] T. A. Basuki, A. Cerone, R. Barbuti, A. Maggiolo-Schettini, and P. Milazzo. Modelling the dynamics of an aedes albopictus population. *Proceedings of Application of Membrane Computing, Concurrency and Agent-based Modelling in Population Biology*, 2010.

[10] R. B. Boone, S. J. Thirgood, and J. G. C. Hopcraft. Serengeti wildebeest migratory patterns modeled from rainfall and new vegetation growth. *Ecology*, 87(8):1987–1994, 2006.

[11] P. Bottoni, C. Martn-Vide, G. Paun, and G. Rozenberg. Membrane systems with promoters/inhibitors. *Acta Inf.*, 38(10):695–720, 2002.

[12] D. Cacciagrano, F. Corradini, E. Merelli, and L. Tesei. Multiscale bone remodelling with spatial p systems. In G. Ciobanu and M. Koutny, editors, *MeCBIC*, volume 40 of *EPTCS*, pages 70–84, 2010.

[13] L. Cardelli and P. Gardner. Processes in spaces. *Programs, Proofs, Processes, LNCS 6158*, pages 78–87, 2010.

[14] M. Cardona, M. A. Colomer, M. J. Perez-Jimenez, D. Sanuy, and A. Margalida. A P System modeling an ecosystem related to the bearded vulture. *Proceedings of Sixth Brainstowming Week on Membrane Computing*, pages 52 – 66, 2008.

[15] P. Cazzaniga, D. Pescini, F. J. Romero-campero, D. Besozzi, and G. Mauri. Stochastic approaches in p systems for simulating biological systems. In *Proceedings of the Fourth Brainstorming Week on Membrane Computing Sevilla, RGNC REPORT 02/2006*, pages 145–164. Fnix Editora, 2006.

[16] A. Cerone and M. Scotti. Research challenges in modelling ecosystems. 2014.

[17] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.

[18] L. Dematte, C. Priami, and A. Romanel. The BlenX language: a tutorial. *Formal Methods for Computational Systems Biology, Springer, LNCS 5016*, pages 313–365, 2008.

[19] V. Durier, P. Graham, and T. S. Collett. Snapshot memories and landmark guidance in wood ants. *Current Biology, Elsevier Sience Ltd.*, 13:1614–1618, 2003.

[20] D. A. Focks, E. Daniels, D. G. Haile, and J. E. Keesling. A simulation model of the epidemiology of urban dengue fever: Literature analysis, model development, preliminary validation, and samples of simulation results. *The American Journal of Tropical Medicine and Hygiene, 53*, pages 489–506, 1995.

[21] V. Galpin. Towards a spatial stochastic process algebra. In *Proceedings of the 7th Workshop on Process Algebra and Stochastically Timed Activities (PASTA), Edinburgh*, 2008.

[22] M. Gardner. Mathematical games: The fantatic combination of john conway's new solitaire game 'life'. *Scientific American 223(4)*, pages 120–123, 1970.

[23] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computation Physics, 22*, pages 403–434, 1976.

[24] C. C. Guet, A. Gupta, T. A. Henzinger, M. Mateescu, and A. Sezgin. Delayed continuous-time markov chains for genetic regulatory circuits. In *Computer Aided Verification*, pages 294–309. Springer, 2012.

[25] W. A. Hawley. The biology of *aedes albopictus. Journal of American Mosquito Control Association*, pages 1–39, 1988.

[26] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[27] A. G. Hoekstra, J.-L. Falcone, A. Caiazzo, and B. Chopard. Multi-scale modeling with cellular automata: The complex automata approach. In H. Umeo, S. Morishita, K. Nishinari, T. Komatsuzaki, and S. Bandini, editors, *ACRI*, volume 5191 of *Lecture Notes in Computer Science*, pages 192–199. Springer, 2008.

[28] A. G. Hoekstra, E. Lorenz, J.-L. Falcone, and B. Chopard. Towards a complex automata framework for multi-scale modeling: Formalism and the scale separation map. In Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, editors, *International Conference on Computational Science (1)*, volume 4487 of *Lecture Notes in Computer Science*, pages 922–930. Springer, 2007.

[29] R. M. Holdo, R. D. Holt, and J. M. Fryxell. Opposing rainfall and plant nutritional gradients best explain the wildebeest migration in the serengeti. *The American Naturalist*, 173(4):431–445, 2009.

[30] M. John, R. Ewald, and A. M. Uhrmacher. A spatial extension to the π-calculus. *Electronic Notes in Theoretical Computer Science, 194*, pages 133–148, 2009.

[31] O. Kahramanoğullari, F. Jordan, and J. Lynch. CoSBiLab LIME: A language interface for stochastic dynamical modelling in ecology. *Environmental Modelling and Software 26*, pages 685–687, 2011.

[32] I. Karafyllidis and A. Thanailakis. A model for predicting forest fire spreading using cellular automata. *Ecological Modelling*, 99(1):87–97, 1997.

[33] U. B. Kaupp, N. D. Kashikar, and I. Weyand. Mechanism of sperm chemotaxis. *Annual Review of Physiology*, 70:93–117, 2008.

[34] K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. Molecule interaction maps of bioregularity networks: A general rubric for systems biology. *Molecular Biology of a Cell, 17*, pages 1–13, 2005.

[35] K. J. Lohmann, N. F. Putman, and C. M. F. Lohmann. Geomagnetic imprinting: A unifying hypothesis of long-distance natal homing in salmon and sea turtles. *Proceedings of the National Academy of Sciences*, 105(49):19096–19101, 2008.

[36] J. Martin. *Introduction to Languages and the Theory of Computation.* McGraw-Hill Series in Computer Science. MCGRAW HILL BOOK Company, 2003.

[37] M.-E.-C. Mateescu. Propagation models for biochemical reaction networks. 2011.

[38] P. Milazzo. *Qualitative and Quantitative Formal Modeling of Biological Systems.* PhD thesis, Università di Pisa, 2007.

[39] R. Milner. Communication and mobile systems: the π-calculus. *Proceeding of the Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[40] D. H. R. Moreno, P. Federico, and G. A. Canziani. Population dynamics models base on cellular automata that includes habitat quality indices defined through remote sensing. *ISRSE RM*, 2001.

[41] J. Murray. *Mathematical Biology: I. An Introduction.* Interdisciplinary Applied Mathematics. Springer, 2002.

[42] J. Murray. *Mathematical Biology II: Spatial Models and Biomedical Applications.* Intercisciplinary Applied Mathematics: Mathematical Biology. Springer, 2003.

[43] D. Noble. *The music of life: Biology beyond the genome.* Oxford University Press, 2006.

[44] J. Ogutu, H.-P. Piepho, H. Dublin, N. Bhola, and R. Reid. Dynamics of mara–serengeti ungulates in relation to land use changes. *Journal of Zoology*, 278(1):1–14, 2009.

[45] G. Pardini. *Formal Modelling and Simulation of Biological Systems with Spatiality.* PhD thesis, Università di Pisa, 2011.

[46] D. Pescini, D. Besozzi, C. Zandron, and G. Mauri. Analysis and simulation of dynamics in probabilistic p systems. In A. Carbone and N. Pierce, editors, *DNA Computing*, volume 3892 of *Lecture Notes in Computer Science*, pages 236–247. Springer Berlin Heidelberg, 2006.

[47] A. Philippou and M. Toro. Process ordering in a process calculus for spatially-explicit ecological models. *MoKMaSD 2013, Madrid, Spain*, 2013.

[48] C. Priami, A. Regev, W. Silverman, and E. Y. Shapiro. Application of a stochastic name-passing calculus to representation and simulation a molecular processes. *Information Processing Letters, 80*, pages 25–31, 2001.

[49] G. Păun. Computing with membranes. *Journal of Computer and System Sciences, 61*, pages 108–143, 2000.

[50] A. Regev, W. Silverman, and E. Y. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. *Proceeding of the Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[51] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics, Volume 21, Number 4*, 1987.

[52] U. Sauer and N. Z. Matthias Meinemann. GENETICS: Getting closer to the whole picture. *Science*, pages 550–551, 2007.

[53] S. Setiawan and A. Cerone. Stochastic modelling of seasonal migration using rewriting systems with spatiality. *MoKMaSD 2013, Madrid, Spain*, 2013.

[54] D. Tilman and P. K. ed. *Spatial Ecology. The Role of Space in Population Dynamics and Interspecific Interactions.* Princeton University Press, 1997.