Boolean networks and their dynamics: the impact of updates

Loïc Paulevé [1,*] and Sylvain Sené [2,†]

[1] Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400, Talence, France
[2] Université publique, Marseille, France

Abstract

Abstract Boolean networks are a mathematical model which has been widely used since decades in the context of biological regulation networks qualitative modelling. They consist in collections of entities, each having two possible local states (1 – active, and 0 – inactive), which interact with each other over discrete time. The simplicity of their setting together with their high abstraction level are especially convenient to focus on foundations of information transmission in genetic regulation, and on mathematical explanation and prediction of phenomenological observations. This chapter aims to present the Boolean modelling framework, by developing its theoretical bases and emphasising its usefulness for capturing biological regulation phenomena. But it goes beyond that by covering their ability to capture the information transmission and its consequences depending on the ways the entities update their local state over time.

Keywords: Boolean modelling, Boolean networks, updating modes, dynamics and complexity.

1 Introduction

The term *bioinformatics* was originally introduced by Hesper and Hogeweg and defined as "the study of informatic processes in biotic systems" [74, 77, 76]. Whilst bioinformatics raised in the 1970's, we did not have to wait for the appearing of the term to witness studies on biological information processes. Let us simply take a look at the beginning of modern computer science and its development in the first half of the 20th century. Admittedly, the science of computation has its roots in the purely theoretical works in the domain of discrete mathematics by Herbrand and Gödel on recursive functions, by [25, 26] on λ-calculus, and by [86, 87] at the frontiers of both. Nevertheless, early major progresses in this discipline have been made thanks to inspirations from natural phenomenology. The first example is undoubtedly Turing machines. In his seminal papers, Turing built a mechanical comprehension of computation by means of successive deconstructions of the human calculation process [143]. Another classical and in no way less important family of examples, in particular in the context of biological modelling, is automata networks. Automata networks were introduced distinctly in the 1940's by McCulloch and Pitts with artificial neural networks, and by von Neumann with cellular automata [89, 144]. In both of these works, the idea was twofold:

- to abstract the logical structure of life and thus obtain more inclined mathematical models to capture natural phenomena;
- to understand the computational power of these abstractions.

The way to achieve that is rather "simple": it consists in designing networks of entities (called automata) which influence each other over time. As suggested by the information and computability theories, those entities evolve locally inside a finite set, and time is discrete.

*loic.pauleve@labri.fr
†sylvain.sene@lis-lab.fr

Just as biology inspired and keeps inspiring computer science (DNA computing and molecular programming), computer science and discrete mathematics, through the automata network model, have become central in qualitative modelling in molecular biology since the end of the 1960's. Indeed, automata networks provide an appropriate setting to model the structural and dynamical features of biological complex systems at the level of the cell such as gene or protein networks. Besides, the understanding of regulation processes is a key problem that biology cannot solve by itself. It is a fundamental observation made in parallel by Kauffman and Thomas [81, 80, 139] in the framework of genetic regulation, with the support of Delbrück's works emphasising analogies between differentiated cellular types and attractors of network theoretical models [40]. The experimental nature of the common techniques in biology cannot address the whole problem since they are conceived to deal with specific matters. In other terms, experimentations are not adapted for this problem. Their repetition allows to acquire sharp knowledge on biological elements (subject to human interpretation, observation mistakes, statistical biases...) but, for complexity reasons notably, they prevent from putting this knowledge in perspective and have hindsight enough to comprehend the causalities and the effect of combinations of regulations. Kauffman and Thomas early mentioned that biology needs to rest on more theoretical sciences to develop more general and systematic approaches of the living underlying issues. Kauffman made explicit in 1971 "the urgent need for theories about the ways in which integrated genetic control systems might function" [82]. Thomas wrote in 1973: "the mere description of a situation as it is seen at a given state of research has often become heavy and tedious, requiring long sentences which are easily ambiguous or misleading. I have felt for some years an increasing necessity for a formalisation of the concepts in the field" [139]. Following these lines, both of them introduced first models of gene regulation networks by using a peculiar restriction of automata networks, Boolean networks, i.e. automata networks in which elements can be either active or inactive, so that they are in fact a generalisation of McCulloch & Pitts' neural networks. This choice comes from the idealisation consisting in ignoring the activities of mRNAs and proteins. Informally, a Boolean network can be viewed as a directed graph composed of Boolean automata (taking values in the set $\{0, 1\}$) that influence each other through local Boolean functions which govern their evolution over time. Despite this abstraction which can be considered simplistic, Boolean networks have very relevant characteristics discussed further in the chapter which make this mathematical model the most used and impacting in the framework of regulation network modelling to this day.

Whilst Kauffman and Thomas worked intrinsically on the same mathematical objects, their approaches were not equivalent. The main differences lie in network structures and the ways automata update their local state over time. When Kauffman chose to formalise gene regulation through an approach inheriting from statistical mechanics with randomly constructed regular graphs whose nodes represent genes which update their state synchronously, Thomas used an approach closer to combinatorics and theoretical computer science with general graphs whose nodes evolve in a totally non-deterministic asynchronous manner, arguing the absence of genetic synchronicity. Of course, further studies based on these seminal works have relaxed the hypotheses mentioned above. On the one hand, works on Kauffman's model called *random Boolean networks* have re-examined the assumptions of regular graph structures and synchronicity. Probabilistic asynchronicity was notably considered [73, 60], as it has been widely made in the context of cellular automata [54]. On the other hand, the asynchronicity of Thomas' model has been questioned. Some studies focused on variations reinforcing it by adding delays [84, 142, 3], other ones relaxed it by allowing synchronous events [103]. However, whatever the choices made about synchronicity, a fundamental point is that Boolean networks have been and keep being widely used in the framework of biological modelling, for formal analysis and prediction purposes. Examples of this are the modelling of the control of cellular differentiation processes of T cells [92, 98, 1] and blood cells [31], the modelling of cancer development [30] and drug resistance [147], the sharp understanding of the floral morphogenesis of *Arabidopsis thaliana* [93, 94], and the analysis of the dynamics of the genetic control of budding yeast *Saccharomyces cerevisiae* [83], fission yeast [38], and *Candida albicans* yeast [146], to name but a few.

More generally, this issue about (a)synchronicity and the chosen method to compute the new local states of automata in Boolean networks has very strong relations with the concept of *discrete time* (and qualitative modelling), and is thus fundamental, on at least three grounds. First, from the biological standpoint, the passage of time and its consequences are certainly a major issue at the different scales of the living. For instance, if we consider genetic regulations, the question of how gene expressions occur over time is central. Some elements have recently been proposed in the research undertaken on chromatin dynamics [17, 90] but they are only preliminary and do not allow to pinpoint a fine spectrum of biologically plausible updates. Second, from the mathematical standpoint, understanding the consequences of choosing this or that specific updating mode, and the sensitivities of a network to this choice, is one of the fundamental current issues in the field, following in the discrete world some lines given by René Thom in his essay on structural stability [136]. Third, in a modelling framework, from the theoretical computer science point of view, the non-countable infinite number of possible updating ways and their consequences on the dynamics of Boolean networks, along with the underlying intrinsic exponential complexity between static and dynamical views of the latter, require sampling relevant updating modes, and understanding (even predicting) their impact.

As we will emphasise in this chapter, the updating mode is a crucial part of the dynamical modelling, and consequently of the validation of models. In the scope of biology, the validation of a Boolean model has multiple facets. On the one hand, Boolean networks can serve as conceptual and phenomenological models: they can demonstrate sufficient or necessary mechanisms leading to complex emergent dynamical properties, inspired by the observation of natural systems, e.g. the emergence of patterns, rhythms, resilience. The validation of such models builds primarily on the justification of the different parameters, including the updating mode, and how they can relate to the current knowledge and hypotheses on the biological system. On the other hand, Boolean networks are also employed as "mechanistic" models of biological systems: the automata model the activity of identified biological entities, which can be directly or indirectly measured in laboratory. Boolean networks then describe the logic of activation and inactivation of biological entities over time. In many such applications, such as to gene regulatory networks, the Boolean modelling of the dynamics of gene activity is viewed as a simplification of a quantitative system, making the assumption of non-linear influences and abstracting away influence thresholds based on copy-number or concentration of the transcription factors. The validation of these models then relies on their capability to reproduce observed behaviours. Such a modelling approach can be qualified as "top-down", with a direct specification of an abstract model to derive predictions on a concrete quantitative system, in opposition to "bottom-up" abstractions. Bottom-up abstractions typically start from a precise quantitative model and derive mathematically abstractions of it, which formally ensure that some properties will be preserved in the abstract model [34, 53, 2]. However, in practice, the knowledge of the system is often insufficient to build such a precise quantitative model, even putting aside its parameterisation. It turns out that the Boolean network modelling framework fits with the current granularity of the knowledge of biological processes. Moreover, modellers often take advantage of the abstraction level of Boolean networks to account for automata relating to different biological concepts: for instance mixing automata modelling activity of genes, of proteins, and more abstract concepts such as phenotypes. The fact that they enable transforming static information on known and putative influences into a dynamical *in silico* model which can be easily simulated is probably one of the reasons for their current growing adoption in theoretical and experimental biology research groups.

In summary, since the seminal works of Kauffman and Thomas, Boolean networks have been widely employed, and certainly are the most used model in biological regulation network modelling as a matter of fact. Nevertheless, most of the works which have followed their lines have been conducted principally in the direction of applications. There currently exists a deep gap between the applicative and fundamental aspects of Boolean networks in favour of applications. As a consequence, the intrinsic properties of Boolean networks, such as the way that information is transmitted along the entities, the ability to produce this or that global dynamical behaviour depending on local interactions, are still not well understood. Of course, results have already been obtained but they are far from being sufficient. In some sense, this lack of fundamental knowledge tends to limit the innovating power of applications.

It can be explained by the fact that, at present, as soon as a question is addressed from the application standpoint, scientists provide ad hoc solutions to them, which prevents to develop a unified, federative and general framework. This actually is paradoxical because without more theoretical fundamental knowledge, applications cannot evolve deeply and become more impacting. In light of this, keeping in mind what Kauffman and Thomas said (see above), we made the choice in this chapter to address biological modelling and more precisely gene regulation network modelling with a rather theoretical but illustrated standpoint, through the prism of updating modes and dynamics.

To this end, this chapter outline is as follows:

- After a short briefing on general notations, Section 2 presents the Boolean network framework and gives the main definitions and notations specific to it.
- Section 3 puts the emphasis on some relevant case studies from the literature in biological modelling which show some important features of Boolean modelling.
- Section 4 focuses on deep fundamental known results on Boolean networks coming from discrete mathematics and computer science which establish very important links with qualitative knowledge on genetic regulations.
- Finally, Section 5 concludes this chapter with a discussion about time, and current research directions and links to software tools for analysing Boolean networks with the different updating modes presented here.

General notations and definitions

The Boolean domain $\{0, 1\}$ is denoted by $\mathbb{B}$. Given a finite set S (resp. a vector V), $|S|$ (resp. $|V|$) denotes its cardinality (resp. its dimension). The set $\{1, \dots, n\}$ is denoted by $[\![n]\!]$. The empty set $\{\}$ is denoted by $\varnothing$. The empty vector $()$ is denoted by $\vec{\varnothing}$. Given two vectors $V = (1, 2)$ and $V' = (2, 3)$, their concatenation is denoted by $V \parallel V' = (1, 2, 2, 3)$. Given two Boolean vectors $x, y \in \mathbb{B}^n$, the set of their components having a different value is denoted by $\Delta(x, y) = \{i \in [\![n]\!] \mid x_i \neq y_i\}$. Given a Boolean vector $x \in \mathbb{B}^n$ and $i \in [\![n]\!]$, $x^{\overline{i}}$ is the Boolean vector identical to x except on the i-th component which is inverted, i.e. $\Delta(x, x^{\overline{i}}) = \{i\}$. This naturally extends to the subsets of $[\![n]\!]$. Formally, we also use, for any $x = (x_1, \dots, x_n) \in \mathbb{B}^n$:

- $\forall W = W' \uplus \{i\} \subseteq [\![n]\!]$, $x^{\overline{W}} = (x^{\overline{i}})^{\overline{W'}} = (x^{\overline{W'}})^{\overline{i}}$, where $\uplus$ represents the union of two disjoint sets, i.e. $A = B \uplus C \iff A = B \cup C$ and $B \cap C = \varnothing$;
- $\overline{x} = x^{\overline{[\![n]\!]}} = (\neg x_1, \dots, \neg x_n)$, with $\neg a = 1 - a$.

Notice that a Boolean vector $x = (x_1, \dots, x_n) \in \mathbb{B}^n$ is sometimes written as the binary word $x = x_1 \dots x_n$ for the sake of clarity in some contexts (notably in the figures).

Given a directed graph $G = (V, E)$ and two vertices i and j of V, the set $V^-(i) = \{j \mid (j, i) \in E\}$ (resp. $V^+(i) = \{j \mid (i, j) \in E\}$) denotes the *in-neighbourhood* (resp the *out-neighbourhood*) of i. A *path* is a finite or infinite series of edges which joins a sequence of vertices. Consider the relation of being strongly connected of such a graph as the fact that there is a path between every ordered pair of its vertices. This relation is an equivalence relation and the induced subgraphs by its equivalence classes are the *strongly connected components* of G. A strongly connected component is said to be *terminal* if and only if there is no path from it to another strongly connected component.

The classical Boolean operators used in this chapter are: $\neg$ denotes the unary negation operator (NOT) such that $\neg a = 0$ if and only if $a = 1$ and *vice versa*; $\vee$ denotes the binary disjunction operator (OR) such that $a \vee b = 1$ if and only if at least one of the operands equals 1; $\wedge$ denotes the binary conjunction operator (AND) such that $a \wedge b = 1$ if and only if both the operands equal 1; and $\veebar$ denotes the binary exclusive disjunction operator (XOR) such that $a \veebar b = (a \wedge \neg b) \vee (\neg a \wedge b)$, i.e. either a equals 1 or b equals 1.

2 The Boolean network framework

This section aims to present the mathematical and computational model of Boolean networks. After a discussion on the motivations for choosing such a framework in the context of modelling, we put the emphasis on classical formal definitions related to these networks, their updating modes and their dynamics.

2.1 On the simplicity of Boolean networks

From a general standpoint, automata networks can be used to model any real system which satisfies the following three properties:

- it is a real system made up of distinct entities which interact with each other;
- each entity is characterised by a variable quantity, that is precisely intended to be expressed formally in terms of states of automata in the model;
- the events undergone by the real system, just like the mechanisms which are responsible for them, are not directly and fully observable with certainty: only the consequences of these events, namely completely accomplished changes, are.

These three properties impose very few restrictions on the set of systems which can be modelled by automata networks. These theoretical objects are therefore generic models for a very large variety of real systems.

Let us come back to the "variable quantity" of entities mentioned in the second point above. To be translated in terms of automaton states, it calls for a formalisation. This consists in choosing whether what interests us in the variation of this quantity is of Boolean, discrete or continuous nature. As an illustration, consider the example of genetic regulations and choose the action of a gene as the so-called "variable quantity":

- If, in the action of this gene, what interests us is its expression or non-expression, then we fall directly into the Boolean case.
- If what interests us are the different ways in which the gene acts on other elements of the system, then we can match a state to each of them. We then fall into the discrete case which can be encoded in Boolean (an automaton state k can simply be encoded by $\log_2(k)$ automata).
- Finally, if we measure the action of the gene through the concentration of proteins it produces, then we fall into the continuous case. This concentration is usually presented as a sigmoidal function. To deal with this case, three common methods exist. The first one is to stay in a continuous formalism. The second one consists in approximating the sigmoid by cutting it into intervals to which states are made to correspond so as to fall back into the discrete formalism. The third one consists in considering the extremal concentrations of the sigmoid so as to fall in the Boolean case.

This shows that we can give different statuses to the Boolean framework depending on whether we see it as a direct modelling of reality or as an approximation or an encoding of an intrinsically continuous or discrete modelling. Note that the direct Boolean modelling is consistent with the choice to focus on the state changes of the automata rather than on their states themselves. As an illustration, if we compare automata to internal combustion engines, the interest is more in the fact that an engine, taken separately, goes from the "off" state to the "on" state and vice versa than on the quantity of electricity supplied by the battery to start it or on that released by the spark plugs to cause the explosion and initiate movement. Under this assumption, the Boolean abstraction is necessary and sufficient. In addition, the discourse of biologists is generally imbued with syntactic elements of propositional logic and it is not uncommon to hear sentences such as: "in the absence of repressor α, gene β is expressed" or further "if the products of genes α and β form a complex, the latter promotes the expression of gene γ whereas these genes tend to inhibit its expression when they are in monomeric form." This again agrees with a direct modelling of reality in Boolean formalism.

In addition, Boolean networks derive other interesting benefits from their simplicity. In particular, they provide a framework with clearly defined contours, ideal for tackling fundamental problems around the modelling of complex interacting systems. Some of these problems are presented in this chapter. Given the variety of their nature, ranging from structural sensitivity analyses to dynamical behaviour characterisations, and the current state of our knowledge, such problems could not currently benefit from significantly more elaborate frameworks. This would inevitably lead to diluting the primary questions and de-structuring the problems posed by drawing attention to ancillary questions induced by the set of parameters to be considered and not intrinsically included in the initial problem. For these questions, on the contrary, Boolean networks offer just what is needed and facilitate the manipulation of a minimal concept of causality, which is rooted in the notion of state change. Their merit therefore lies in the reliability of the information they potentially provide, and the simplicity of their setting (see below) associated with their ability to capture most of the heterogeneities and intricacies carried by the modelled systems.

2.2 Boolean network specification

A *Boolean network* of dimension n is specified by a function $f : \mathbb{B}^n \to \mathbb{B}^n$ mapping Boolean vectors of dimension n to Boolean vectors of dimension n. For every $i \in [\![n]\!]$, $f_i : \mathbb{B}^n \to \mathbb{B}$ is the i-th component of this function, that we call the *local function* of automaton i. Classically, the 2^n Boolean vectors of $\mathbb{B}^n$ are called the *configurations* of the Boolean network. A configuration denoted by x can be also viewed as a one-to-one function from $[\![n]\!]$ to $\mathbb{B}$. With this notation, we say that x_i is the local state, abbreviated by *state* in the sequel, of automaton i.

The local functions may only depend on a subset of automata of the network, and these dependencies may even be monotone. This information can be summarised by a directed and signed graph, called the *influence graph* (often called interaction graph in the literature), (V, E) with its vertices $V = [\![n]\!]$ and edges $E \subseteq [\![n]\!] \times \{+, -\} \times [\![n]\!]$. Whenever $(i, +, j) \in E$, we say that i is an *activator* of j; whenever $(i, -, j) \in E$, i is an *inhibitor* of j. An influence graph is *simple* if there is at most one edge from one vertex to another, i.e. there is no $(i, j) \in [\![n]\!]^2$ with $\{(i, +, j), (i, -, j)\} \subseteq E$. An influence graph $([\![n]\!], E)$ is *compatible* with an influence graph $([\![n]\!], E')$ whenever $E \subseteq E'$.

The influence graph of f has a positive (resp. negative) edge from i to j if and only if one can assign a state to each automaton other than i so that the local function of j becomes equal (resp. different) to the state of i. It is formally denoted by $\mathscr{G}_f = ([\![n]\!], E_f)$ where

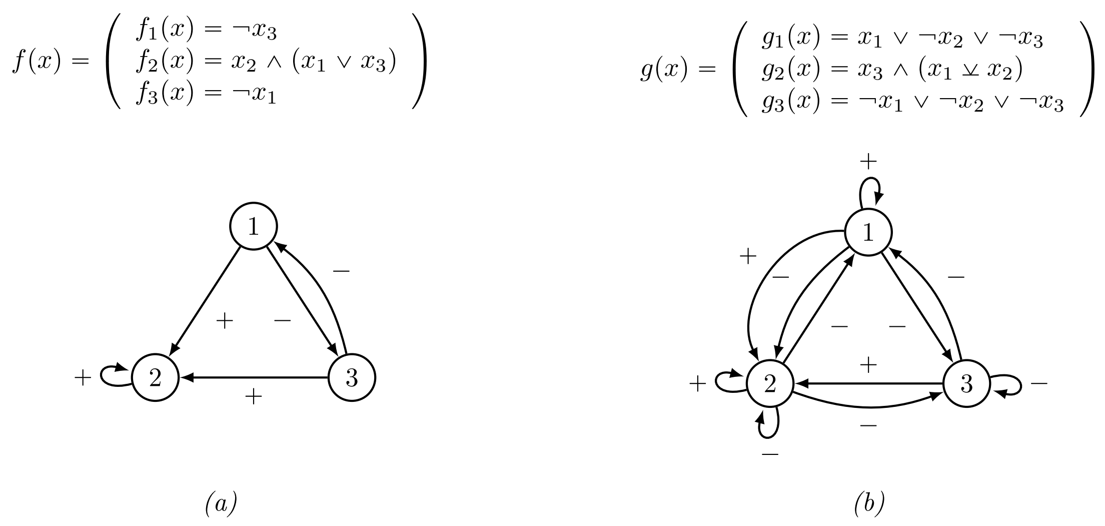
- $(i, +, j) \in E_f$ if and only if there exists $x, y \in \mathbb{B}^n$ with $\Delta(x, y) = \{i\}$ and $x_i = 0$ such that $f_j(x) = 0$ and $f_j(y) = 1$;
- $(i, -, j) \in E_f$ if and only if there exists $x, y \in \mathbb{B}^n$ with $\Delta(x, y) = \{i\}$ and $x_i = 0$ such that $f_j(x) = 1$ and $f_j(y) = 0$.

By forgetting the signs on the edges, a weaker way to relate the local functions to the edges of the influence graph is given by the following formula:

$$\forall i \in [\![n]\!], \exists x \in \mathbb{B}^n,\ f_j(x) \neq f_j(x^{\bar{i}}) \iff (i, j) \in E_f.$$

Such a relation implies notably that $\mathscr{G}_f$ is minimal, i.e. any of its edges is supposed to represent an effective influence based on essential Boolean variables [36]. In other terms, given f and a configuration $x \in \mathbb{B}^n$, the set of edges of $\mathscr{G}_f$ which operate an influence on x equals $E_f(x) = \{(i, s \in \{+, -\}, j) \mid f_j(x) \neq f_j(x^{\bar{i}})\}$.

A Boolean network f is *locally monotone* if and only if its influence graph $\mathscr{G}_f$ is simple. In such a case, for each local function f_j, for each i so that $(i, +, j) \in E_f$, the sole change of state of automaton i from 0 to 1 cannot cause f_j to switch from 1 to 0; for each i so that $(i, -, j) \in E_f$, the sole change



$f(x) = \begin{pmatrix} f_1(x) = \neg x_3 \\ f_2(x) = x_2 \wedge (x_1 \vee x_3) \\ f_3(x) = \neg x_1 \end{pmatrix}$
$g(x) = \begin{pmatrix} g_1(x) = x_1 \vee \neg x_2 \vee \neg x_3 \\ g_2(x) = x_3 \wedge (x_1 \veebar x_2) \\ g_3(x) = \neg x_1 \vee \neg x_2 \vee \neg x_3 \end{pmatrix}$
1
2
3
+
−
(a)
(b)

Figure 1: The two Boolean networks presented in Example 1: *(a)* Boolean network f and its associated influence graph $\mathscr{G}_f$; *(b)* Boolean network g and its associated influence graph $\mathscr{G}_g$.

of state of automaton i from 0 to 1 cannot cause f_j to switch from 0 to 1. In other words, the local functions f_j are monotone according to a component-wise ordering of Boolean vectors which depends on j, where activators are ordered increasingly ($\leqslant$), and inhibitors decreasingly ($\geqslant$). Remark that f is monotone only if it is locally monotone. Formally, a Boolean network f is locally monotone if and only if, $\forall j \in V$, $\forall i \in V^-(j)$, f_j either satisfies:

$$\forall x \in \mathbb{B}^n, x_i = 0 \implies f_j(x) \leqslant f_j(x^{\bar{i}}), \text{ and thus } (i, +, j) \in E_f,$$

or

$$\forall x \in \mathbb{B}^n, x_i = 0 \implies f_j(x) \geqslant f_j(x^{\bar{i}}), \text{ and thus } (i, -, j) \in E_f.$$

Example 1. Let us consider the two following distinct Boolean networks f and g of dimension $n = 3$ defined so that they respect the minimality constraint of their influence graph:

$$f(x) = \begin{pmatrix} f_1(x) = \neg x_3 \\ f_2(x) = x_2 \wedge (x_1 \vee x_3) \\ f_3(x) = \neg x_1 \end{pmatrix} \quad \text{and} \quad g(x) = \begin{pmatrix} g_1(x) = x_1 \vee \neg x_2 \vee \neg x_3 \\ g_2(x) = x_3 \wedge (x_1 \veebar x_2) \\ g_3(x) = \neg x_1 \vee \neg x_2 \vee \neg x_3 \end{pmatrix}$$

Given these Boolean networks, it is easy to construct their associated influence graphs $\mathscr{G}_f = (\llbracket 3 \rrbracket, E_f)$ and $\mathscr{G}_g = (\llbracket 3 \rrbracket, E_g)$. The general idea is to look at every local function f_i, written in conjunctive or dijunctive normal form. If there is a literal x_j (resp. its negation $\neg x_j$) in its definition, with $j \in \llbracket n \rrbracket$, then $(j, +, i) \in E_f$ (resp. $(j, -, i) \in E_f$). Consider Boolean network f. By local function f_1, we deduce that automaton 1 of f is influenced negatively by automaton 3; definition of f_2 shows that automaton 2 is influenced positively by automata 1 and 3, and itself; for automaton 3, we deduce that it is influenced negatively by automaton 1. Now, for Boolean network g, we have: by local function g_1, automaton 1 is influenced positively by itself and negatively by the others; by expanding the XOR operator, g_2 can be rewritten into $g_2(x) = x_3 \wedge ((x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2))$, which shows that automaton 2 is influenced positively by automaton 3, and both positively and negatively by automaton 1 and itself. This leads to obtain $\mathscr{G}_f$ and $\mathscr{G}_g$ depicted in Figure 1. From local function definitions as well as influence graph constructions, it appears that f is locally monotone, which is not the case for g. Indeed, function g_2 has a XOR operator which is non-monotone, which leads $\mathscr{G}_g$ not to be simple; indeed, $\{(1, -, 2), (1, +, 2), (2, -, 2), (2, +, 2)\} \subseteq E_g$.

2.3 Boolean network dynamics

A dynamical system describes the temporal evolution of the set of its *configurations*, being $\mathbb{B}^n$ in the case of Boolean networks. A Boolean network f can lead to several distinct ways of computing the possible evolutions of a configuration $x \in \mathbb{B}^n$, depending on how the latter gets updated according to $f(x)$. Thus, specifying this *updating mode* is a compulsory part of Boolean network analysis and modelling which asks for introducing several concepts.

2.3.1 Updates

In a configuration, events update the state of one or more automata. Suppose that $x \in \mathbb{B}^n$ is the current configuration of a network f of dimension n whose influence graph is $\mathscr{G}_f = (\llbracket n \rrbracket, E_f)$. We say that automaton $i \in \llbracket n \rrbracket$ is *updated* if its state x_i becomes $f_i(x)$. If $f_i(x) = x_i$, then the update of i is not effective in x. Such a local event (even ineffective) leads to what is called an *update* of configuration at the global level which is described by the *updating function* $\phi_i : \mathbb{B}^n \to \mathbb{B}^n$ of automaton i such that:

$$\forall x \in \mathbb{B}^n,\ \phi_i(x) = (x_0, \ldots, x_{i-1}, f_i(x), x_{i+1}, \ldots, x_n).$$

Do not confuse $f_i(x)$, $\phi_i(x)$ and $f(x)_i$. The first notation refers to the state of automaton i after the execution of its local function on x, the second one refers to the configuration obtained after the execution of the updating function ϕ_i on x, and the third one refers to the state of automaton i after the execution of (global) function f on x.

More generally, the updating function extends to subsets of $\llbracket n \rrbracket$. Notice that, abusing notations here for the sake of clarity, we have $\phi_i(x) = \phi_{\{i\}}(x)$. Formally, given $W \subseteq \llbracket n \rrbracket$, the updating function $\phi_W : \mathbb{B}^n \to \mathbb{B}^n$ represents the configuration change caused by the update of all the automata belonging to W, and is such that:

$$\forall x \in \mathbb{B}^n, \forall i \in \llbracket n \rrbracket,\ \phi_W(x)_i = \begin{cases} f_i(x) & \text{if } i \in W, \\ x_i & \text{otherwise.} \end{cases}$$

As an illustration, consider the network f defined in Figure 1 and let us focus on four distinct updates on its configurations. The first update is ineffective and consists in changing nothing. The second update changes the state of automaton 1 by application of ϕ_1, the third one changes the states of both automata 2 and 3 by application of $\phi_{\{2,3\}}$, and the fourth one changes the state of every automaton by application of $\phi_{\llbracket n \rrbracket}$. Notice that applying $\phi_{\llbracket n \rrbracket}$ to a configuration is equivalent to applying directly f on this configuration. Table 1 presents the results of these updates.

Until now, we have defined the classical framework of updates, in which updates correspond to executions of one or more local functions simultaneously. We will see later (when we introduce the most permissive updating mode) that we can extend this framework and thus consider more complex ways of updating a Boolean network.

2.3.2 Transitions and trajectories

Informally speaking, a transition is a couple $(x, y) \in \mathbb{B}^n \times \mathbb{B}^n$ which represents the change of configuration x into configuration y operated by a series of events (possibly composed of only one event). The transitions which come from a unique update are said to be *elementary*. Conversely, those which come from a series of several updates are said to be *non-elementary*.

More formally, given a Boolean network f, an *elementary transition* $(x, y) \in \mathbb{B}^n \times \mathbb{B}^n$ of f corresponds to the update in x of any subset $W \neq \emptyset \subseteq \llbracket n \rrbracket$ of automata, *i.e* such that $y = \phi_W(x)$. By convention, we denote them by $x \to_f y$ or $x \xrightarrow{W}_f y$. There exist two kinds of elementary transitions: the *asynchronous transitions* are such that $W \neq \emptyset \subsetneq \llbracket n \rrbracket$; the *synchronous transitions* are such that

Table 1: Configurations, local functions ($(f_i)_{i \in [\![3]\!]}$) and four updating functions ($\phi_\varnothing$, ϕ_1, $\phi_{\{2,3\}}$, and $\phi_{[\![3]\!]}$) of Boolean network f presented in Example 1 and depicted in Figure 1.a.

$x = (x_1, x_2, x_3)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$\phi_\varnothing(x)$	$\phi_1(x)$	$\phi_{\{2,3\}}(x)$	$\phi_{[\![3]\!]}(x) \equiv f(x)$
$(0,0,0)$	1	0	1	$(0,0,0)$	$(1,0,0)$	$(0,0,1)$	$(1,0,1)$
$(0,0,1)$	0	0	1	$(0,0,1)$	$(0,0,1)$	$(0,0,1)$	$(0,0,1)$
$(0,1,0)$	1	0	1	$(0,1,0)$	$(1,1,0)$	$(0,0,1)$	$(1,0,1)$
$(0,1,1)$	0	1	1	$(0,1,1)$	$(0,1,1)$	$(0,1,1)$	$(0,1,1)$
$(1,0,0)$	1	0	0	$(1,0,0)$	$(1,0,0)$	$(1,0,0)$	$(1,0,0)$
$(1,0,1)$	0	0	0	$(1,0,1)$	$(0,0,1)$	$(1,0,0)$	$(0,0,0)$
$(1,1,0)$	1	1	0	$(1,1,0)$	$(1,1,0)$	$(1,1,0)$	$(1,1,0)$
$(1,1,1)$	0	1	0	$(1,1,1)$	$(0,1,1)$	$(1,1,0)$	$(0,1,0)$

$W = [\![n]\!]$. The reflexive and transitive closure of $\rightarrow_f$ is denoted by $\rightarrow_f^*$ and is defined as: given two configurations $x, y \in \mathbb{B}^n$, $x \rightarrow_f^*$ if and only if there exists a series of transitions which changes x into y.

Consider now the non-elementary transitions of the form $(x, y) \in \mathbb{B}^n \times \mathbb{B}^n$ of f denoted by $x \rightarrow_f^* y$. They correspond to series of several elementary transitions such that: $x \rightarrow_f^* y \iff \exists p \geqslant 2, \exists x^1, \ldots, x^{p-1} \in \mathbb{B}^n,\ x \rightarrow_f x^1 \rightarrow_f \ldots \rightarrow_f x^{p-1} \rightarrow_f y$. In other terms, any *non-elementary transition* $x \rightarrow_f y$ is a vector of sets $(W_k)_{1 \leqslant k \leqslant p}$ such that $y = \phi_{W_p} \circ \cdots \circ \phi_{W_1}(x)$, and can be represented by $x \xrightarrow{W_1, \ldots, W_p}_f y$.

Trajectories are vectors of (elementary or not) transitions $((x^0, x^1), (x^1, x^2), \ldots, (x^{p-1}, x^p))$ such that $x^0 \rightarrow_f x^1 \rightarrow_f x^2 \rightarrow_f \ldots \rightarrow_f x^{p-1} \rightarrow_f x^p$.

Let us illustrate theoretically these definitions and notations by the following trajectory of an arbitrary Boolean network f of dimension $n \geqslant 3$, given $i, j \in [\![n]\!]$ and $W_1 = \{1, \ldots, \lfloor \frac{n}{2} \rfloor\}, W_2 = \{\lceil \frac{n}{2} \rceil, \ldots, n\} \subsetneq [\![n]\!]$:

$$x^0 \xrightarrow{[\![n]\!]}_f \left(x^1 = f(x^0)\right) \xrightarrow{W_1, W_2}_f \left(x^2 = \phi_{W_2} \circ \phi_{W_1}(x^1)\right) \xrightarrow{\{i,j\}}_f \left(x^3 = \phi_{\{i,j\}}(x^2)\right).$$

This trajectory is composed of three transitions among which:

- two are elementary: $x^0 \xrightarrow{[\![n]\!]}_f x^1$ is a synchronous transition and $x^2 \xrightarrow{\{i,j\}}_f x^3$ is an asynchronous one;
- one is non-elementary, $x^1 \xrightarrow{W_1, W_2}_f x^2$, and could be decomposed into $x^1 \xrightarrow{W_1}_f y \xrightarrow{W_2}_f x^2$, namely a series of two elementary transitions.

Now, for the sake of clarity, consider configuration $(1, 1, 1)$ of Boolean network f defined in Figure 1. We know that $W_1 = \{1, \ldots, \lfloor \frac{n}{2} \rfloor\} = \{1\}$ and $W_2 = \{\lceil \frac{n}{2} \rceil, \ldots, n\} = \{2, 3\}$; suppose that $i = 1$ and $j = 3$. With Table 1, it is easy to compute the previous trajectory on it:

$$x^0 = (1,1,1) \xrightarrow{[\![n]\!]}_f x^1 = (0,1,0) \xrightarrow{W_1, W_2}_f x^2 = (1,1,0) \xrightarrow{\{1,3\}}_f x^3 = (1,1,0).$$

2.3.3 Updating mode and transition graph

A dynamical system associated with a Boolean network f is the combination of f with an updating mode. In other words, the dynamics of f depends on the chosen way to proceed with automata updates, from which of course the trajectories to be considered depend. From a general standpoint, the updating modes define restrictions on the set of imaginable updates (and thus of the set of underlying transitions) allowed in the set of configurations of a network. Thus, given a Boolean network, choosing an updating mode gives formally a framework to the study of its dynamics. Here, we provide the main definitions related to Boolean network dynamics by paying particular attention to updating modes and transition graphs. Observe that, given f of dimension n and an updating mode μ, the dynamical system (f, μ) defines a binary *transition relation* between configurations of $\mathbb{B}^n$ denoted by $\rightarrow_{(f,\mu)} \subseteq \mathbb{B}^n \times \mathbb{B}^n$.

Main concepts related to dynamical systems Let (f, μ) be the dynamical system associated with Boolean network f of dimension n and updating mode μ. This dynamical system can be represented by a directed graph $\mathscr{D}_{(f,\mu)} = (\mathbb{B}^n, \rightarrow_{(f,\mu)})$, where $\rightarrow_{(f,\mu)}$ represents the set of realisable transitions. This graph is usually called the *transition graph* of (f, μ). Now, as for transitions above, we can define the reflexive and transitive closure of relation $\rightarrow_{(f,\mu)}$, denoted by $\rightarrow^*_{(f,\mu)}$ as follows: given two configurations $x, y \in \mathbb{B}^n$, we have $x \rightarrow^*_{(f,\mu)} y$ if and only if there exists a path from x to y in $\mathscr{D}_{(f,\mu)}$. Configuration $x \in \mathbb{B}^n$ is said to be *transient* if there exists a configuration y such that $(x, y) \in \rightarrow^*_{(f,\mu)}$ and $(y, x) \notin \rightarrow^*_{(f,\mu)}$. Configurations which are not transient are called *limit configurations*. Because n is finite, these configurations induce the terminal strongly connected components of $\mathscr{D}$, called the *limit sets* of (f, μ). If there exists at least one trajectory from a transient configuration to a limit set, then this limit set is called an *attractor* of (f, μ) [33, 95]. The *basin of attraction* of an attractor $\mathcal{A}$ of (f, μ), denoted by $\mathcal{B}(\mathcal{A})$, is the sub-graph of $\mathscr{D}_{(f,\mu)}$ induced by the set of transient configurations x such that, for any limit configuration y belonging to $\mathcal{A}$, $(x, y) \in \rightarrow^*_{(f,\mu)}$. A limit set of cardinal 1, i.e. composed of a unique limit configuration x which only admits $(x, x) \in \rightarrow_{(f,\mu)}$ as possible outgoing transitions, is called a *fixed point* (or stable configuration) of (f, μ). A limit set of cardinal greater than 1, i.e. composed of configurations which can reach each other without reaching the others, is called a *limit cycle* (or sustained oscillation) of (f, μ).

Now that the main general concepts on dynamical systems are clarified, some classical updating modes are going to be defined. Let us start by specifying that, considering Boolean networks as mathematical objects evolving over discrete time, it is common to denote by x^t (resp. x_i^t) an image (or the image, depending on the context) of configuration $x = x^0$ (resp. the state of the automaton $i \in [\![n]\!]$) at step (of time) $t \in \mathbb{N}$. It is essential to understand that it is impossible to present all updating modes, simply because of a combinatorial argument. If we consider a Boolean network f of dimension n and only the deterministic updating modes based on updates, such as those which have been formally presented until now, the most general way to define an updating mode is as an infinite vector of subsets of $[\![n]\!]$. From this, with Cantor's diagonal argument, it is easy to derive that the set of such deterministic updating modes is an uncountable set. Therefore, in order to avoid confusions and misleadings, our following presentation separates voluntarily deterministic updating modes from non-deterministic ones. Deterministic updating modes are such that each configuration has a unique image, i.e. given a configuration, only one outgoing transition is possible. Non-deterministic updating modes are such that each configuration may admit several images, i.e. given a configuration, distinct outgoing transitions are possible.

2.3.4 Deterministic updating modes

Block-sequential updating modes Since the works of Robert [121, 122], the community working on Boolean networks, and more generally on automata networks, has paid particular attention to a specific family of deterministic and periodic updating modes, classically called *block-sequential updating modes* in the literature [41, 44, 9, 65] and introduced as series-parallel updating modes by Robert. Informally, given a Boolean network f of dimension n, the idea of such modes is to partition automata of $[\![n]\!]$ into disjoint blocks (or subsets), and to make automata of one block execute their updating functions synchronously (or in parallel) while blocks are iterated in series and periodically. Let E be an arbitrary finite set. A vector E' composed of subsets of E is an *ordered partition* of E if all the elements of E' are non-empty, pairwise disjoint and if their union equals E. Such an updating mode

$\mu = \mathsf{bs}$ admissible for Boolean network f of dimension n is defined as an ordered partition $(W_1, \ldots, W_p)$ of $[\![n]\!]$, with $p \leqslant |[\![n]\!]|$.

Definition 1. *Let f be a Boolean network of dimension n and $\mathsf{bs} = (W_1, \ldots, W_p)$ an ordered partition of $[\![n]\!]$. Dynamical system (f, bs) is defined by the transition graph $\mathscr{D}_{(f,\mathsf{bs})} = (\mathbb{B}^n, \rightarrow_{(f,\mathsf{bs})} \subseteq \mathbb{B}^n \times \mathbb{B}^n)$ where:*

$$\forall x, y \in \mathbb{B}^n,\ x \rightarrow_{(f,\mathsf{bs})} y \iff y = \phi_{W_p} \circ \cdots \circ \phi_{W_1}(x).$$

The transitions operated in such dynamical systems are non-elementary, except in one case, when $\mathsf{bs} = [\![n]\!]$, i.e. when all the transitions are synchronous. This latter case is certainly the most classical from the theoretical standpoint, because the underlying dynamical system $(f, ([\![n]\!]))$ is directly defined by f such that for every $x \in \mathbb{B}^n$, its image is $\phi(x) = f(x)$. This is called the *synchronous*, or *parallel* updating mode. Notice also that the number of possible block-sequential updating modes is exponential in the number of automata n, as it is given by the Fubini number whose recurrence formula is: $\#_{\mathsf{bs}}(n) = \sum_{i=0}^{n-1} \binom{n}{i} \#_{\mathsf{bs}}(i)$, with $\#_{\mathsf{bs}}(0) = 1$.

As an illustration, consider Boolean network f of dimension 3 defined in Figure 1. Since it is composed of 3 automata, it admits thirteen distinct block-sequential updating modes. If we compute the thirteen underlying dynamical systems, it appears that six amongst them are different. Their respective transition graphs are depicted in Figure 2. Let us focus now on $\mathscr{D}_{(f,(\{1,3\},\{2\}))}$ pictured on the up right corner of that figure. We can see that dynamical system $(f, (\{1,3\}, \{2\}))$ has five different limit sets, four fixed points ($\mathsf{001} \circlearrowleft$, $\mathsf{011} \circlearrowleft$, $\mathsf{100} \circlearrowleft$, and $\mathsf{110} \circlearrowleft$) which are not attractors, and one limit cycle of length 2 ($\mathsf{000} \rightleftarrows \mathsf{101}$) which is an attractor whose basin of attraction is $\{\mathsf{010}, \mathsf{111}\}$. The whole figure shows another interesting feature: the fixed points are preserved, whatever the block-sequential updating mode, which is not the case for the limit cycle. We will formally speak of this feature in Section 4.1.

Block-sequential updating modes are definitely relevant from the mathematical standpoint. Their study has emphasised strong properties about the convergence of dynamical systems towards limit sets [122, 64], about their influence on these limit sets [9, 52, 67, 8, 10], and we are still far from having elucidated each of their underlying dynamical consequences. They also have brought interesting knowledge in the framework of genetic regulation network modelling [93, 11, 42, 124]. Nevertheless, the fact they are defined by means of ordered partitions of the set of automata is a strong constraint. First, they are periodic. This is a quite convenient mathematical restriction since it allows to leave the uncountable universe of deterministic updating modes and enter into a countable one as soon as period $p \in \mathbb{N}$ is determined and finite. Second, they make every automaton update its state exactly once during an updating period. The fact that every automaton updates its state at least once during a period is actually a good property, because what would be the point of an automaton which never updates its state? Last but not least, the fact that it does only once per period prevents from observing peculiar phenomena. For instance, given a network, this makes it impossible for one of its subnetworks to act with its own clock, distinct from that of the rest of the network. Considering such biological phenomenology asks for conceiving more expressive and more relaxed deterministic updating modes in the framework of Boolean networks. Notice that more expressive deterministic periodic updating modes have never been studied in depth *per se*, even if they are evoked in [65, 47].

Block-parallel updating modes A natural way to design new deterministic, periodic updating modes rests on the property above which says that each automaton needs to update its state at least once during a period. This leads to focus on periodic updating modes of arbitrary period $p \in \mathbb{N}$ defined as infinite periodic vectors $(W_1, \ldots, W_p, W_1, \ldots, W_p, \ldots)$ such that $\forall i \in \{1, \ldots, p\}, W_i \subseteq [\![n]\!]$ and $\bigcup_{i=1}^{p} W_i = [\![n]\!]$. For the sake of clarity, such updating modes are usually denoted by means of finite vectors such as $(W_i)_{i \in \mathbb{N}/p\mathbb{N}}$. From now on, given a dynamical system (f, μ), let us distinguish time steps from elementary time steps. There is one *time step* between configuration $x \in [\![n]\!]$ and

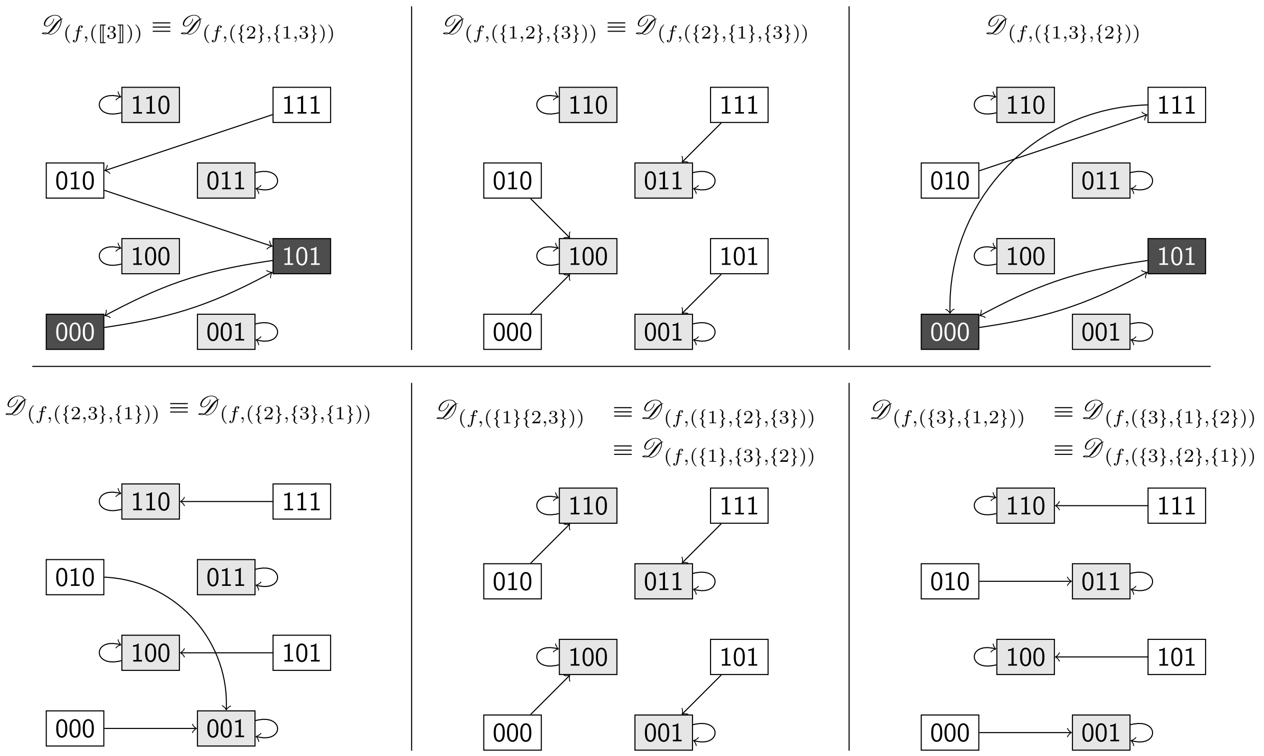

$\mathscr{D}_{(f,(\llbracket 3\rrbracket))} \equiv \mathscr{D}_{(f,(\{2\},\{1,3\}))}$
$\mathscr{D}_{(f,(\{1,2\},\{3\}))} \equiv \mathscr{D}_{(f,(\{2\},\{1\},\{3\}))}$
$\mathscr{D}_{(f,(\{1,3\},\{2\}))}$
$\mathscr{D}_{(f,(\{2,3\},\{1\}))} \equiv \mathscr{D}_{(f,(\{2\},\{3\},\{1\}))}$
$\mathscr{D}_{(f,(\{1\}\{2,3\}))} \equiv \mathscr{D}_{(f,(\{1\},\{2\},\{3\}))} \equiv \mathscr{D}_{(f,(\{1\},\{3\},\{2\}))}$
$\mathscr{D}_{(f,(\{3\},\{1,2\}))} \equiv \mathscr{D}_{(f,(\{3\},\{1\},\{2\}))} \equiv \mathscr{D}_{(f,(\{3\},\{2\},\{1\}))}$
110
111
010
011
100
101
000
001

Figure 2: The 6 possible distinct block-sequential dynamics of Boolean network f defined in Figure 1 represented by their associated transition graphs $\mathscr{D}_{(f,\mathsf{bs})}$, where bs is amongst the 13 possible ordered partitions of $\llbracket 3\rrbracket$.

another $y \in \llbracket n\rrbracket$ if $x \rightarrow_{(f,\mu)} y$. If this transition is non-elementary, as explained above, it can be decomposed into elementary ones and we say that there are p *elementary time steps* between x and y if $x \rightarrow_{(f,\mu)} y \iff x = x^0 \rightarrow_{\phi_{W_1}} x^1 \rightarrow_{\phi_{W_2}} \dots \rightarrow_{\phi_{W_{p-1}}} x^{p-1} \rightarrow_{\phi_{W_p}} x^p = y$. These periodic updating modes are called *fair periodic updating modes* because for any elementary time step $t \in \mathbb{N}$, there exists $k \in \{1,\dots,p\}$ such that every automaton of $\llbracket n\rrbracket$ is updated in the elementary time interval $[t; t+k]$. Nevertheless, whilst the set of fair updating modes is indeed countable, it is too much big to hope studying it in depth. Indeed, for a network of dimension n and given a period p, let us argue that it is greater than the number of coverings of dimension p of $\llbracket n\rrbracket$, which means greater than $\sum_{k=0}^{n}(-1)^k\binom{n}{k}\binom{2^{n-k}-1}{p}$ [32], because of the order on subsets and the subset repetition availability.

To constrain this set of possible periodic updating modes, consider the dual of the set of block-sequential updating modes, that is the set of block-parallel updating modes. Informally, the idea of such modes is to partition automata of $\llbracket n\rrbracket$ into disjoint blocks, and to make automata of one block execute their updating functions sequentially while blocks are iterated in parallel. Formally, a *partitioned order of* $\llbracket n\rrbracket$ is a set $\{S_k\}_{1\leqslant k\leqslant s}$, with $1 \leqslant s \leqslant |\llbracket n\rrbracket|$, such that:

- $\forall k \in \{1,\dots,s\}$, $S_k \neq \vec{\varnothing}$ is a vector of automata of $\llbracket n\rrbracket$ without repetitions;
- $\forall i \in \llbracket n\rrbracket,\ i \in S_k \implies \forall \ell \in \{1,\dots,s\}\setminus\{k\},\ i \notin S_\ell$;
- $\|_{k=1}^{s} S_k$ covers $\llbracket n\rrbracket$.

From this, we derive that a *block-parallel updating mode* $\mu = \mathsf{bp}$ admissible for a Boolean network f of dimension n is a partitioned order $\{S_1,\dots,S_s\}$ of $\llbracket n\rrbracket$, with $s \leqslant |\llbracket n\rrbracket|$. Notice that the number of partitioned orders of a set equals that of its ordered partitions.

Proposition 1. *Given a Boolean network f of dimension n, block-parallel updating modes admissible*

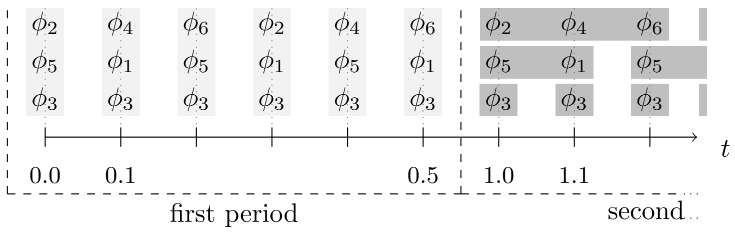


Figure 3: Insight of the rewriting of block-parallel updating mode $\mathsf{bp} = \{(2,5,6),(5,1),(3)\}$ into a classical periodic updating mode form. This corresponds to the elementary time diagram of updating function executions. Two periods are pictured, the first one entirely, the second one partially. In the first period, the rectangles filled in light gray come from a reading in columns of the diagram and represent the subsets W_ℓ of three automata evolving in parallel. In the second period, the rectangles filled in gray come from a reading in row of the diagram and represent the sub-vectors S_k of automata evolving sequentially.

for f are fair periodic updating modes.

Proof. Given f and n, any block-parallel updating mode defined as a partitioned order $\{S_k\}_{1 \leqslant k \leqslant s}$ of $[\![n]\!]$ can be rewritten into a periodic vector $(W_\ell)_{1 \leqslant \ell \leqslant p}$, with period $p = \operatorname{lcm}(|S_k|_{1 \leqslant k \leqslant s})$, of subsets of $[\![n]\!]$ of cardinal s, such that:

$$\forall \ell \in \{1, \dots p\},\ W_\ell = \{i \in [\![n]\!] \mid \exists k, \exists j \in \{1, \dots, |S_k|\}, \exists m \in \mathbb{N},\ (S_k)_j = i \text{ and } \ell = j + m|S_k|\}.$$

Let us denote this rewriting by the non injective function rw such that $\operatorname{rw}(\{S_k\}_{1 \leqslant k \leqslant s}) = (W_\ell)_{1 \leqslant \ell \leqslant p}$, and let us denote its right inverse by $\operatorname{rw}^{-1}_{\mathsf{bp}}$. The construction of rw implies notably that $\forall i \in [\![n]\!], \exists \ell \in \{1, \dots, p\},\ i \in W_\ell$. □ □

Definition 2. *Let f be a Boolean network of dimension n and $\mathsf{bp} = \{S_1, \dots, S_s\}$ a partitioned order of $[\![n]\!]$. Dynamical system (f, bp) is defined by the transition graph $\mathscr{D}_{(f,\mathsf{bp})} = (\mathbb{B}^n, \rightarrow_{(f,\mathsf{bp})} \in \mathbb{B}^n \times \mathbb{B}^n)$ such that:*

$$\forall x, y \in \mathbb{B}^n,\ x \rightarrow_{(f,\mathsf{bp})} y \iff y = \phi_{W_p} \circ \dots \circ \phi_{W_1}(x),$$

where $(W_\ell)_{1 \leqslant \ell \leqslant p} = \operatorname{rw}(\mathsf{bp})$ and $p = \operatorname{lcm}(|S_k|_{1 \leqslant k \leqslant s})$.

As a first illustration, consider a Boolean network f of dimension 6 and the block-parallel updating mode $\mathsf{bp} = \{(2,4,6),(5,1),(3)\}$. One can derive from the latter that automata 2, 4, and 6 are updated every three elementary time steps, automata 5 and 1 every two elementary time steps, and automaton 2 at each elementary time step. More precisely, automaton 2 (resp. 4 and 6) is updated at each elementary time step $t \in \mathbb{N}$ such that $t \equiv 0 \mod 3$ (resp. $t \equiv 1 \mod 3$, and $t \equiv 2 \mod 3$), automaton 5 (resp. 1) is updated at each elementary time step $t \in \mathbb{N}$ such that $t \equiv 0 \mod 2$ (resp. $t \equiv 1 \mod 2$). This corresponds exactly to the following periodic updating mode: $(\{2,3,5\},\{1,3,4\},\{3,5,6\},\{1,2,3\},\{3,4,5\},\{1,3,6\}) = \operatorname{rw}(\mathsf{bp})$. An insight of this rewriting is given in Figure 3. Now, given a configuration $x = (x_1, x_2, x_3, x_4, x_5, x_6)$, its global updating according to bp gives the following transition:

$$x^0 = (x_1, x_2, x_3, x_4, x_5, x_6) \rightarrow_{(f,\mathsf{bp})} x^1 = \phi_{\{1,3,6\}} \circ \phi_{\{3,4,5\}} \circ \phi_{\{1,2,3\}} \circ \phi_{\{3,5,6\}} \circ \phi_{\{1,3,4\}} \circ \phi_{\{2,3,5\}}(x^0).$$

As a second illustration, consider again Boolean network f of dimension 3 given in Figure 1. There exist thirteen block-parallel updating modes admissible for it. Seven underlying dynamical systems are different which are depicted in Figure 4. Notice that five of them correspond to dynamical systems emerging from block-sequential updating modes given in Figure 2. What is interesting at this stage is

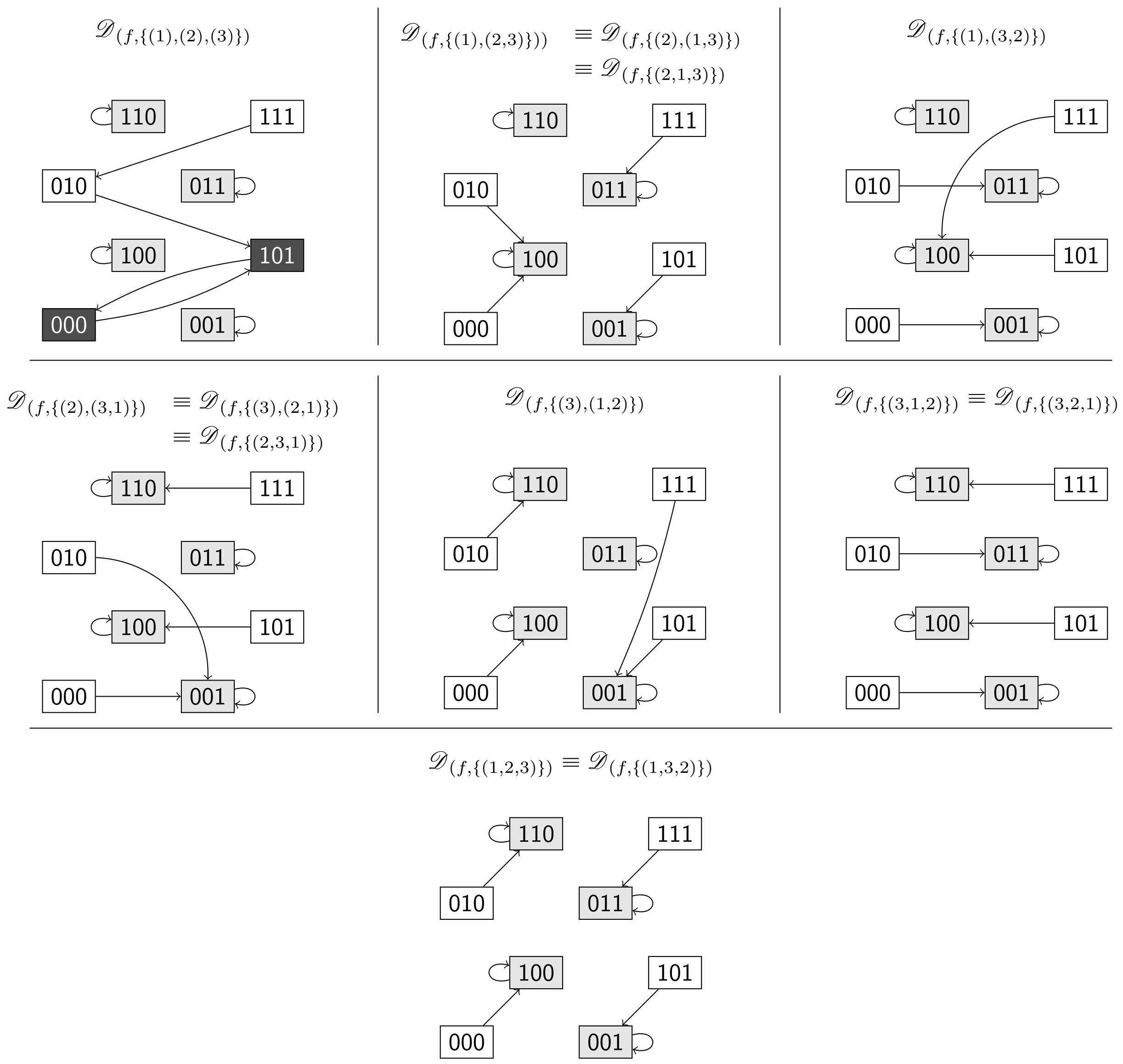

$\mathscr{D}_{(f,\{(1),(2),(3)\})}$
$\mathscr{D}_{(f,\{(1),(2,3)\})} \equiv \mathscr{D}_{(f,\{(2),(1,3)\})} \equiv \mathscr{D}_{(f,\{(2,1,3)\})}$
$\mathscr{D}_{(f,\{(1),(3,2)\})}$
$\mathscr{D}_{(f,\{(2),(3,1)\})} \equiv \mathscr{D}_{(f,\{(3),(2,1)\})} \equiv \mathscr{D}_{(f,\{(2,3,1)\})}$
$\mathscr{D}_{(f,\{(3),(1,2)\})}$
$\mathscr{D}_{(f,\{(3,1,2)\})} \equiv \mathscr{D}_{(f,\{(3,2,1)\})}$
$\mathscr{D}_{(f,\{(1,2,3)\})} \equiv \mathscr{D}_{(f,\{(1,3,2)\})}$
110
111
010
011
100
101
000
001

Figure 4: The 7 possible distinct block-parallel dynamics of Boolean network f defined in Figure 1 represented by their associated transition graphs $\mathscr{D}_{(f,\mathsf{bp})}$, where bp is amongst the 13 possible partitioned orders of $[\![3]\!]$.

that two of them are not: the one in the up right frame, and the one in the middle central frame. This shows that block-parallel updating modes can lead to dynamical systems not observable by means of block-sequential updating modes.

Let us denote the set of block-parallel updating modes defined on $[\![n]\!]$ viewed as periodic modes by $\{\mathrm{rw}(\mathsf{bp}([\![n]\!]))\}$, and the set of block-parallel updating modes defined on $[\![n]\!]$ by $\{\mathsf{bs}([\![n]\!])\}$. The following properties hold:

- $|\{\mathrm{rw}(\mathsf{bp}([\![n]\!]))\}| \leqslant |\{\mathsf{bs}([\![n]\!])\}|$. The number of partitioned orders is upper-bounded by the number of ordered partitions since rewritting function rw is not injective. Indeed, for $n \geqslant 4$, there exist specific partitioned orders in which subvectors of automata lead to the same rewriting into vector of automata subsets. For instance, $\mathrm{rw}(\{(1,2),(3,4)\}) = \mathrm{rw}(\{(1,4),(3,2)\}) = (\{1,3\},\{2,4\})$;

- $\forall n \geqslant 3, \{\mathrm{rw}(\mathsf{bp}(\llbracket n \rrbracket))\} \neq \{\mathsf{bs}(\llbracket n \rrbracket)\}$, since updating repetitions in a period are allowed by block-parallel updating modes. Considering Boolean networks (and *a fortiori* automata networks), a consequence of this is that, given a network f, the set of its underlying block-parallel dynamical systems is not necessarily the set of its underlying block-sequential dynamical systems.
- $\{\mathrm{rw}(\mathsf{bp}(\llbracket n \rrbracket))\} \cap \{\mathsf{bs}(\llbracket n \rrbracket)\} \neq \emptyset$. Indeed, the parallel and the $n!$ sequential updating modes, as well as other specific modes, are both block-parallel and block-sequential updating modes.
- If bp is a partitioned order $\{S_k\}_{1 \leqslant k \leqslant s}$ such that $\forall k, \ell \in \{1,\dots,s\},\ |S_k| = |S_\ell| = c$, then $\mathrm{rw}(\mathsf{bp})$ is a block-sequential updating mode whose subset cardinals equal s and whose period is c. Conversely, if bs is an ordered partition $(W_k)_{1 \leqslant k \leqslant p}$ such that $\forall k, \ell \in \{1,\dots,p\},\ |W_k| = |W_\ell| = c$, then $\mathrm{rw}^{-1}_{\mathsf{bp}}(\mathsf{bs})$ is a block-parallel updating mode of cardinal c whose sub-vector dimensions equal p. In other terms, rw is a bijection for such block-parallel and block-sequential updating modes sets and $\mathrm{rw}^{-1}_{\mathsf{bp}}$ is the (two-sided) inverse of rw in this case.

These properties are interesting *per se* since they emphasise that the block-parallel and block-sequential families of periodic updating modes are different whilst they share some properties. However, it would be of better interest to have families of updating modes which integrate both of them. To this end, let us introduce a generalisation of block-parallel updating modes.

Towards local-clocks updating modes

Block-parallel generalisation The general idea here consists in restarting from block-parallel updating modes and to extend them by allowing automata of distinct blocks to synchronise their updatings, thanks to the concept of *waiting delay*. To this end, consider the set $\llbracket n \rrbracket \cup \{0\}$, where element 0 is a fictitious automaton with no local function which will serve as a kind of waiting delay. Formally, a *partitioned order of $\llbracket n \rrbracket$ with delay* is a set $\{S_k\}_{1 \leqslant k \leqslant s}$, with $1 \leqslant s \leqslant n$, for which the following properties hold:

- $\forall k \in \{1,\dots,s\},\ S_k \neq \vec{\emptyset}$ is a vector of automata of $\llbracket n \rrbracket \cup \{0\}$ without repetitions of elements of $\llbracket n \rrbracket$, and such that $\exists i \in \llbracket n \rrbracket,\ i \in S_k$;
- $\forall i \in \llbracket n \rrbracket,\ i \in S_k \implies \forall \ell \in \{1,\dots,s\} \setminus \{k\},\ i \notin S_\ell$;
- $\|_{k=1}^{s} S_k$ covers $\llbracket n \rrbracket$;

From this, we derive that a *general block-parallel updating mode* $\mu = \mathsf{gbp}$ admissible for a Boolean network f of dimension n is a partitioned order $\{S_1,\dots,S_s\}$ of $\llbracket n \rrbracket$ with delay, with $s \leqslant n$.

This leads to the following proposition whose proof is similar to that of Proposition 1. In this case, however, the left inverse of rw is distinct from $\mathrm{rw}^{-1}_{\mathsf{bp}}$ since it needs to generate 0s in sub-vectors, and is denoted by $\mathrm{rw}^{-1}_{\mathsf{gbp}}$.

Proposition 2. *Given a Boolean network f of dimension n, general block-parallel updating modes admissible for f are fair periodic updating modes.*

Definition 3. *Let f be a Boolean network of dimension n and $\mathsf{gbp} = \{S_1,\dots,S_s\}$ a partitioned order of $\llbracket n \rrbracket$ with delay. Dynamical system (f,gbp) is defined by the transition graph $\mathscr{D}_{(f,\mathsf{gbp})} = (\mathbb{B}^n, \to_{(f,\mathsf{gbp})} \subseteq \mathbb{B}^n \times \mathbb{B}^n)$ such that:*

$$\forall x, y \in \mathbb{B}^n,\ x \to_{(f,\mathsf{lc})} y \iff y = \phi_{W_p} \circ \cdots \circ \phi_{W_1}(x),$$

where $(W_\ell)_{1 \leqslant \ell \leqslant p} = \mathrm{rw}(\mathsf{gbp})$ and $p = \mathrm{lcm}(|S_k|_{1 \leqslant k \leqslant s})$.

As a first illustration, consider a Boolean network f of dimension 4 and the general block-parallel updating mode $\mathsf{gbp} = \{(1,0),(0,2,0),(3),(0,4)\}$. One can derive from it that automaton 1 (resp. 2, 3, and 4) executes its updating function at elementary time step $t = 0$ (resp. $t = 1$, $t = 0$, and $t = 2$), and then every two (resp. three, one, two) elementary time steps. This corresponds exactly to the

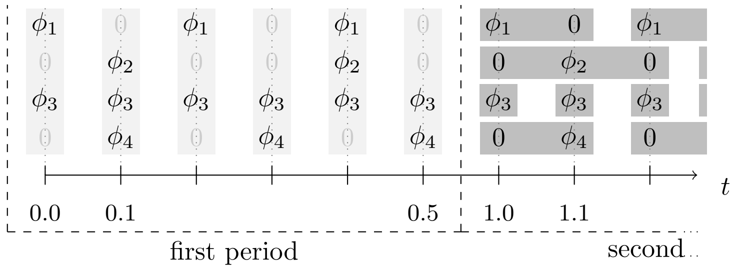


Figure 5: Insight of the rewriting of general block-parallel updating mode $\mathsf{gbp} = \{(1,0),(0,2,0),(3),(0,4)\}$ into a classical periodic updating mode form. This corresponds to the elementary time diagram of updating functions executions. Two periods are pictured, the first one entirely, the second one partially. In the first period, the rectangles filled in light gray come from a reading in columns of the diagram and represent the subsets W_ℓ of automata of $[\![4]\!]$ evolving in parallel. In the second period, the rectangles filled in gray come from a reading in row of the diagram and represent the sub-vectors S_k of automata evolving sequentially.

following periodic updating mode: $(\{1,3\},\{2,3,4\},\{1,3\},\{3,4\},\{1,2,3\},\{3,4\})$. In other words, we have:

$$\begin{aligned} &\mathrm{rw}(\mathsf{gbp}) = (\{1,3\},\{2,3,4\},\{1,3\},\{3,4\},\{1,2,3\},\{3,4\}),\\ \text{and}\quad &\mathrm{rw}^{-1}_{\mathsf{gbp}}\big((\{1,3\},\{2,3,4\},\{1,3\},\{3,4\},\{1,2,3\},\{3,4\})\big) = \mathsf{gbp}\,. \end{aligned}$$

An insight of this rewriting is presented in Figure 5.

As a second illustration, consider once again Boolean network f of dimension 3 given in Figure 1, and the three following general block-parallel updating modes admissible for it:

- $\mathsf{gbp}_1 = \{(1,0),(2),(3,0,0)\}$ is an updating mode of period 6 which makes automaton 1 (resp. 2, and 3) update its state at each two (resp. one, and three) elementary time steps from $t=0$;
- $\mathsf{gbp}_2 = \{(1,0,2),(3,0)\}$ is an updating mode of period 6 which makes automaton 1 (resp. 2, and 3) update its state at each three (resp. three, and two) elementary time steps from $t=0$ (resp. $t=2$, and $t=0$);
- $\mathsf{gbp}_3 = \{(1,2),(3,0)\}$ is an updating mode of period 2 which makes automata 1 and 3 update their states at each two elementary time steps from $t=0$, and automaton 2 update its state at each two elementary time steps from $t=1$.

Their associated dynamical systems are depicted in Figure 6. First, remark that $\mathscr{D}_{(f,\{(1,0),(2),(3,0,0)\})} = \mathscr{D}_{(f,(\{1,2,3\},\{2\},\{1,2\},\{2,3\},\{1,2\},\{2\}))}$ is neither a block-sequential updating mode nor a block-parallel one. This emphasises that general block-parallel updating modes can generate dynamics which are not observable thanks to block-sequential and block-parallel updating modes. Second, remark that $\mathscr{D}_{(f,\{(1,0,2),(3,0)\}))} = \mathscr{D}_{(f,(\{1,3\},\varnothing,\{2,3\},\{1\},\{3\},\{2\}))}$ equals $\mathscr{D}_{(f,\{(3),(1,2)\}))} = \mathscr{D}_{(f,(\{1,3\},\{1,2\})))}$. This emphasises two distinct relevant features:

- The periodical form of a general block-parallel updating mode can be partially composed of empty subsets which correspond to elementary time steps where nothing happens. In this example, considering configuration $x = (x_1,x_2,x_3)$, its global updating according to gbp_2 gives the following transition:

$$\begin{aligned} & x^0 = (x_1,x_2,x_3) \quad \to_{(f,\mathsf{gbp}_2)} \quad \phi_{\{2\}} \circ \phi_{\{3\}} \circ \phi_{\{1\}} \circ \phi_{\{2,3\}} \circ \phi_{\varnothing} \circ \phi_{\{1,3\}}\\ \iff\ & x^0 = (x_1,x_2,x_3) \quad \to_{(f,\mu)} \quad \phi_{\{2\}} \circ \phi_{\{3\}} \circ \phi_{\{1\}} \circ \phi_{\{2,3\}} \circ \phi_{\{1,3\}}. \end{aligned}$$

- general block-parallel updating modes which are not block-parallel can generate the same dynamics directly observable thanks to block-parallel updating modes. Notice that it is also the case for block-sequential updating modes.

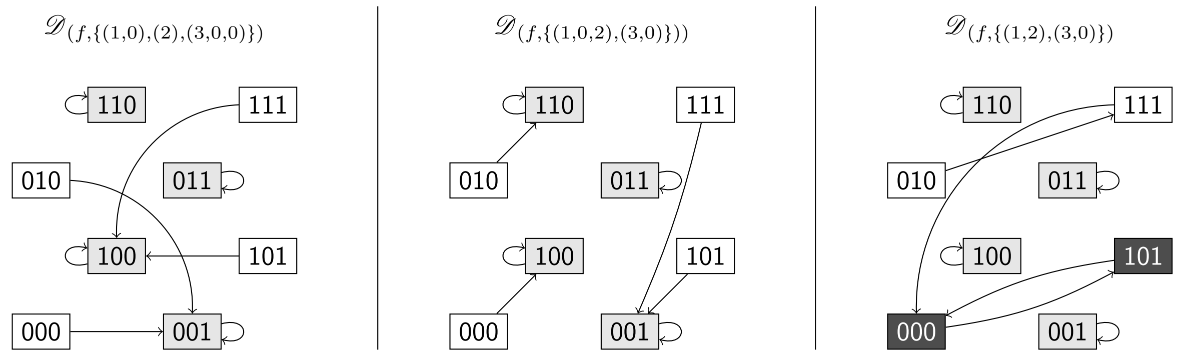

$\mathscr{D}_{(f,\{(1,0),(2),(3,0,0)\})}$
110
111
010
011
100
101
000
001
$\mathscr{D}_{(f,\{(1,0,2),(3,0)\}))}$
110
111
010
011
100
101
000
001
$\mathscr{D}_{(f,\{(1,2),(3,0)\})}$
110
111
010
011
100
101
000
001

Figure 6: Three possible distinct general block-parallel dynamics of Boolean network f defined in Figure 1 represented by their associated transition graphs.

Third, remark that $\mathsf{gbp}_3 = \{(1,2),(3,0)\}$ is a block-sequential updating mode by definition since $\mathrm{rw}(\mathsf{gbp}_3) = (\{1,3\},\{2\})$. Actually, the general block-parallel updating modes of equal sub-vectors dimensions are either block-sequential updating modes or updating modes whose underlying dynamics equals a block-sequential dynamics (they can have empty subsets in their periodical form).

Local-clocks Actually, the three previous classes of periodic updating modes "by blocks" are particular sub-classes of *local clocks updating modes* [119, 120].

The general idea underlying the local clocks updating modes rests on a local view, i.e. at the level of the automata of a network, not at the level of the network, of the scheduling of the updates over time. More formally, considering the time scale decomposed according to elementary transitions, given a Boolean network f of dimension n and a periodic updating mode μ, we say that μ is a local clocks updating mode if and only if, for each automaton $i \in [\![n]\!]$, there exist a local period p_i and an initial shift $\rho_i \in \{0,\ldots,m\}$ such that updating function ϕ_i is executed at each time step $t_i \in \mathbb{N}$ such that, for all $k \in \mathbb{N}$, $t_i = \rho_i + kp_i$.

In other terms, each automaton has its own initial shift at which it enters in its own local updating period. From this, it is easy to see that block-sequential and (general) block-parallel updating modes respect this property. Indeed, for block-sequential ones of period p, all the automata execute their updating functions once per period, and thus, all have an updating period equal to p. Moreover, each automaton is initially shifted of a number of time steps equal to the number minus 1 of the subset it belongs to in the ordered partition. For (general) block-parallel ones, the local updating period of each i is the cardinal of the sub-vector it belongs to, and its initial shift corresponds to its position minus 1 in the sub-vector. Proposition 3 below derives directly from the definition of local clocks updating modes and Proposition 2.

Proposition 3. *Given a Boolean network f of dimension n, local clocks updating modes admissible for f are fair periodic updating modes.*

Nevertheless, it is important to notice that local clocks updating modes can lead to elementary dynamics which cannot be captured by the previous ones. This comes from the initial shifts which can force an updating transitory phase before entering into an updating periodic phase. As a consequence, they cannot be directly taken into account in general block-parallel updating modes since the latter do not allow any transitory phase. Indeed, by definition, any initial local shift generated by putting 0s at the beginning of sub-vectors of the partitioned order leads to increase the corresponding local period. As an illustration, consider any Boolean network f composed of 3 automata and the following local clocks updating mode defined by means of the two following vectors: $\rho = (3,1,0)$ and $p = (2,3,2)$, the initial local shift vector and the local period vector respectively (see Figure 7). This updating mode

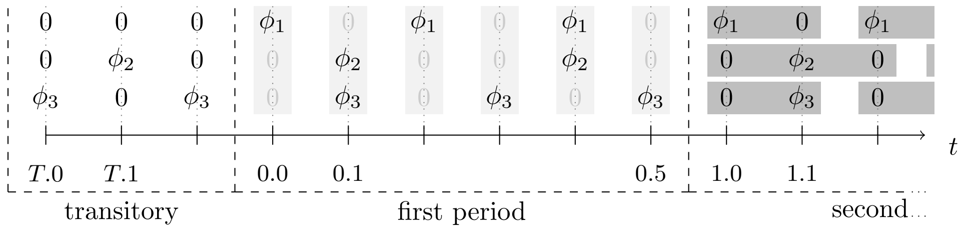


Figure 7: Elementary time diagram of updating functions executions under local clocks updating mode $\mathsf{lc} = (\rho = (3, 1, 0), p = (2, 3, 2))$. The transitory phase and two periods are pictured, the first one entirely, the second one partially. In the first (resp. second) period, the rectangles filled in light gray (resp. gray) give the periodical reading (resp. the general block-parallel view) of the periodic phase of the updating mode.

generates a transitory phase of 3 time steps which cannot be interpreted in the framework of general block-parallel updating modes, and a periodic phase of periods of length 6 which can be. Indeed, the mode $\{(1, 0), (0, 2, 0), (0, 3)\} \equiv (\{1\}, \{2, 3\}, \{1\}, \{3\}, \{1, 2\}, \{3\})$ produces it. More generally, it can be observed that, when there exists at least one initial local shift greater than 1, local clocks updating modes induce a transitory phase of $\max_{i \in [\![n]\!]}(\rho_i) - 1$ time steps which cannot be captured by general block-parallel updating modes.

Updating modes induced by Boolean networks with memory All the periodic updating modes discussed above, and more generally all the periodic updating modes leading to define dynamical systems whose transitions are the results of a global transition function which is a composition of updating functions, have a particularity with respect to Boolean networks and their possible dynamics. They are in some sense *context free.* Indeed, given a Boolean network f of dimension n and such an updating mode μ, configurations of $\mathbb{B}^n$, and thus automata of $[\![n]\!]$, evolve over time according to the schedule determined by μ, without taking into account anything else. The underlying dynamical systems are consequently memoryless. To go even further on deterministic updating modes, one could consider updating modes accounting for some context. The literature on Boolean networks recently put the emphasis on such modes through the gene protein Boolean networks model, introduced by Graudenzi and Serra [69, 70, 71]. The general initial idea of their work rests on:

- considering bipartite Boolean networks with two kinds of automata: a half of them models genes, the other half models their associated one-to-one proteins;
- considering that each protein has its own decay time which defines the number of time steps during which it remains present in the cell after having been produced by the punctual expression of its associated gene.

Without entering into the details of this formalism, the study presented in [63] shows that this new model of genetic regulation networks can be viewed as *Boolean networks with memory*, i.e. Boolean networks with an added delay vector so that an activated automaton remains activated during at least a number of time steps equal to its own delay. Formally, a *Boolean network with memory* f of dimension n is defined as a Boolean network f and a delay vector $d \in \mathbb{N}^n$.

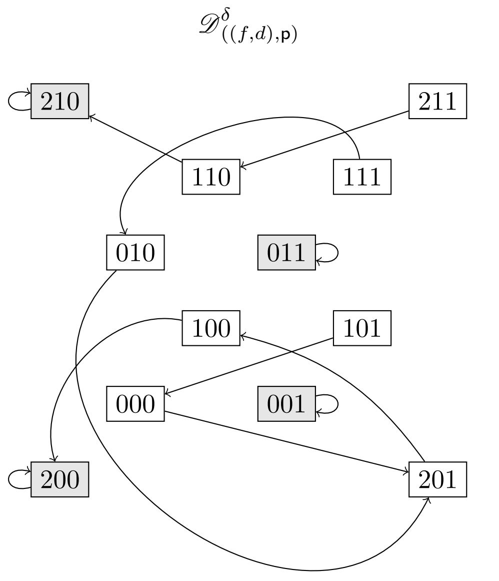
Definition 4. *Let f be a Boolean network of dimension n and $d = (d_1, \ldots, d_n) \in (\mathbb{N} \setminus \{0\})^n$ be a delay vector. The set of its configurations is defined as $X_{(f,d)} = \{(x, \delta) \in \mathbb{B}^n \times \mathbb{N}^n \mid \forall i \in [\![n]\!],\ \delta_i \in \{0, \ldots, d_i\},\ x_i = 0 \iff \delta_i = 0$ and $x_i = 1 \iff \delta_i \in \{1, \ldots, d_i\}\}$. Dynamical system $((f, d), \mathsf{p})$ is defined by the transition graph $\mathscr{D}_{((f,d),\mathsf{p})}$, where p represents the parallel updating mode such that $\mathsf{p} = ([\![n]\!])$, made of transitions based on updating function $\phi^\star : X_{(f,d)} \to X_{(f,d)}$ depending on the delays such that:*

$$\forall (x, \delta), (y, \delta') \in X_{(f,d)},\ (x, \delta) \to_{((f,d),\mathsf{p})} (y, \delta') \iff (y, \delta') = \phi^\star_{[\![n]\!]}(x, \delta),$$

δ	$\phi^\star_{[\![n]\!]}(\delta)$
000	201
001	001
010	201
011	011

δ^t	$\phi^\star_{[\![n]\!]}(\delta)$
100	200
101	000
110	210
111	010

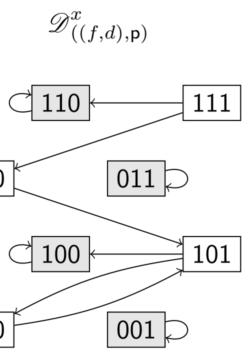
δ^t	$\phi^\star_{[\![n]\!]}(\delta)$
200	200
201	100
210	210
211	110



(a)

x^t	x^{t+1}
000	101
001	001
010	101
011	011

x^t	x^{t+1}
100	100
101	000
	100
110	110
111	010
	110



(b)

Figure 8: *(a)* Dynamics of Boolean network f with memory $d = (2,1,1)$ represented by its deterministic transition table on delay configurations and the associated transition graph $\mathscr{D}^{\delta}_{((f,d),\mathsf{p})}$; *(b)* Projection of the dynamics of (f,d) on the underlying Boolean configurations, for any time step t in $\mathbb{N}$, and its associated transition graph $\mathscr{D}^{x}_{((f,d),\mathsf{p})}$.

where:

$$\forall i \in [\![n]\!],\ \phi^\star_{[\![n]\!]}(x,\delta)_i = (y_i, \delta'_i),$$

with

$$\delta'_i = \begin{cases} 0 & \textit{if } f_i(x) = 0 \textit{ and } \delta_i = 0, \\ \delta_i - 1 & \textit{if } f_i(x) = 0 \textit{ and } \delta_i > 0, \\ d_i & \textit{if } f_i(x) = 1, \end{cases}$$

and

$$y_i = \begin{cases} 1 & \textit{if } \delta'_i \geqslant 1, \\ f_i(x) & \textit{if } \delta'_i = 0. \end{cases}$$

The previous definition shows that, whilst Boolean networks with memory evolve globally according to the parallel updating mode, automata desynchronisation is made possible locally thanks to the delays whose evolution depends on the local context. In other terms, Boolean networks with memory are Boolean networks whose global dynamics depend on the local context of their automata at each time step. Nevertheless, notice that when $d = (1,\ldots,1)$, the dynamics of Boolean network with firing memory (f,d) is nothing else but $\mathscr{D}_{(f,\mathsf{p})}$.

Let us take once more Boolean network f of dimension 3 defined in Figure 1 as an example to illustrate peculiarities and intricacies of the contextual updating modes at stake in Boolean networks with memory. To this end, let us combine f with delay vector $d = (2, 1, 1)$. For convenience and the sake of clarity, let us add a notation based on the specification of the set of configurations of (f, d): rather than considering configurations in $X_{(f,d)}$ which decouple Boolean configurations of f from their possible associated delay configurations of $\mathbb{N}^n$, let us consider only the latter. So, for instance, delay configuration $\delta = (2, 0, 1)$ corresponds actually to configuration $(x = (1, 0, 1), d = (2, 0, 1))$ of (f, d). From this, let us abuse notation and consider that the updating function $\phi^{\star}_{[\![n]\!]}$ maps the set of delay configurations to itself. Figure 8.a depicts the deterministic dynamics of this Boolean network with memory (f, d).

However, an interesting point is that, when the deterministic dynamics is projected on the Boolean configurations of f by getting rid of local delays, then the obtained transition graph is not deterministic anymore, as pictured in Figure 8.b. Indeed, for instance, configuration $(\mathsf{1}, \mathsf{1}, \mathsf{1})$ can reach both configurations $(\mathsf{0}, \mathsf{1}, \mathsf{0})$ and $(\mathsf{1}, \mathsf{1}, \mathsf{0})$. So, generally speaking, in some sense, working on an arbitrary Boolean network with memory (f, d) of dimension n by considering the projection of its dynamics on its partial configurations of $\mathbb{B}^n$ may involve considering the dynamics of f evolving according to a specific non-deterministic updating mode.

2.3.5 Non-deterministic updating modes

Asynchronous updating modes The updating modes defined in the previous section enables specifying which automata should get updated simultaneously, possibly in a given sequence. The *asynchronous* updating mode considers *any* combination of automata to be updated simultaneously: it corresponds to the non-empty elementary transitions introduced in Section 2.3.2.

Definition 5. *Let f be a Boolean network of dimension n. Dynamical system (f, a) is defined by the transition graph $\mathscr{D}_{(f,\mathsf{a})} = (\mathbb{B}^n, \rightarrow_{(f,\mathsf{a})} \subseteq \mathbb{B}^n \times \mathbb{B}^n)$ where:*

$$\forall x, y \in \mathbb{B}^n,\ x \rightarrow_{(f,\mathsf{a})} y \iff \exists W \subseteq [\![n]\!], W \neq \emptyset, y = \phi_W(x).$$

The obtained dynamics are then non-deterministic. Remark that whenever k automata can change of state ($k = |\Delta(x, f(x))|$) then, $2^k - 1$ transitions from x are generated by the asynchronous updating mode.

From a modelling standpoint, the asynchronous updating mode aims at accounting for state changes which occur at different speed and whenever it is not possible to determine local clocks, due for instance to insufficient knowledge on the system, or to an intrinsic stochasticity of the system. As we will detail later in this section, whereas the asynchronous updating mode captures all different time scales of state changes in Boolean automata, it is no longer complete when considering Boolean networks as abstraction of quantitative systems.

Interestingly, this asynchronous updating mode has been barely considered so far when modelling biological systems with Boolean networks (where it is usually referred to as the *general asynchronous* updating mode). Indeed, in the modelling framework of René Thomas, one and only automaton can be updated in a transition, leading to the so-called *fully asynchronous* updating mode (usually referred to as asynchronous in the biological systems modelling community). Nevertheless, some applications consider updating modes that mix parallel and fully asynchronous transitions, thus giving (particular) trajectories of the asynchronous updating mode [55, 37].

Definition 6. *Let f be a Boolean network of dimension n. Dynamical system (f, fa) is defined by the transition graph $\mathscr{D}_{(f,\mathsf{fa})} = (\mathbb{B}^n, \rightarrow_{(f,\mathsf{fa})} \subseteq \mathbb{B}^n \times \mathbb{B}^n)$ where:*

$$\forall x, y \in \mathbb{B}^n,\ x \rightarrow_{(f,\mathsf{fa})} y \iff \exists i \in [\![n]\!], y = \phi_i(x).$$

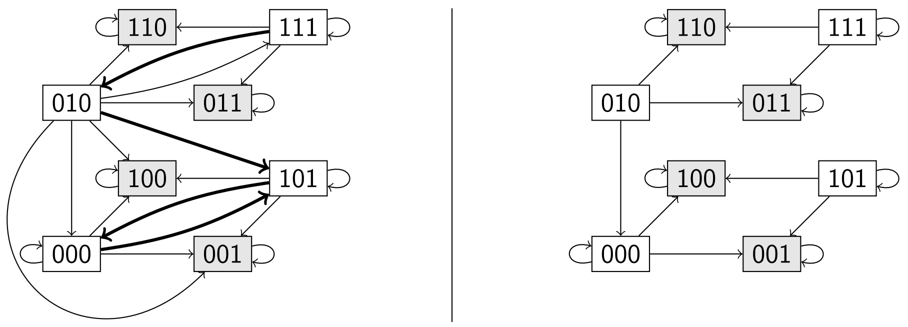


Figure 9: Asynchronous (left) and fully asynchronous (right) dynamics of Boolean network f defined in Figure 1 represented by their associated transition graphs $\mathscr{D}_{(f,\mathsf{a})}$ and $\mathscr{D}_{(f,\mathsf{fa})}$. Non-loop transitions of the parallel updating mode are drawn in bold.

In [140], Thomas justifies this modelling choice in the scope of gene regulatory networks with respect to practical observations on the delays for state changes of genes: "*There is absolutely no reason why the time delays (...) should be equal. As a matter of fact, they are often very unequal. (...) This leads us to a fully asynchronous description, in which all the time delays are different in the absence of an accidental coincidence*". Another practical advantage of the fully asynchronous updating mode is that there are at most n transitions from a same configuration, simplifying greatly the representation of their dynamics.

Figure 9 pictures the asynchronous and fully asynchronous dynamics of Boolean network f from Example 1.

Non-deterministic updates The updating modes considered so far correspond to the application and composition of elementary *deterministic updates* of configurations $\phi : \mathbb{B}^n \to \mathbb{B}^n$. As we have seen above, deterministic updates can generate non-deterministic updating modes, by allowing different updates to be applied on a same configuration. The main exception is the case of Boolean networks with memory, where the dynamics has been defined at the end of Section 2.3.4 as the projection of a discrete parallel dynamics. Other updating modes have been defined in the literature which cannot be expressed as the application nor the composition of elementary updates, such as the Interval updating mode [24, 23] and the most permissive updating mode [109]. Indeed, these latter can generate transitions which are neither elementary nor non-elementary transitions.

In [110], we introduced an extension towards *non-deterministic updates*, with a unifying framework for expressing complex updating modes, including those mentioned above. Non-deterministic updates are functions mapping *sets of configurations*, i.e. of the form $\Phi : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$. We define Φ as a map from sets of configurations to sets of configurations for enabling iterations and compositions of non-deterministic updates. Nevertheless, we assume that for any $X \subseteq \mathbb{B}^n$, $\Phi(X) = \bigcup_{x \in X} \Phi(\{x\})$: one can define Φ only from all singleton configuration sets. This restriction ensures that, for any $X \subseteq \mathbb{B}^n$, each configuration in the image set $y \in \Phi(X)$ can be computed from a singleton set $\{x\}$ for some $x \in \mathbb{B}^n$. In the following, we call such updates *set updates*.

Starting from a singleton configuration set $\{x\}$, the iteration of set updates delineates the domains of configurations the system can evolve to. Thus, set updates naturally define transition relations between configurations:

Definition 7. *Given a set update function Φ for Boolean networks of dimension n, the generated transition relation is given by $\varphi : (2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}) \to 2^{\mathbb{B}^n \times \mathbb{B}^n}$ with $\varphi(\Phi) = \{(x, y) \mid x \in \mathbb{B}^n, y \in \Phi(\{x\})\}$.*

In contrast with deterministic updates, non-deterministic updating modes can be characterised directly by set updates. Indeed, non-deterministic updating modes allow "superposing" alternative updates to generate different transitions from a single configuration x, although each of them is computed with a deterministic update. An example of this is one update ϕ where $\phi(x) = y$ and another update ϕ' where $\phi'(x) = y' \neq y$. Now, let us imagine an updating mode superposing two set updates, Φ and Φ' where, for some configurations $x \in \mathbb{B}^n$, $\Phi(\{x\}) \setminus \Phi'(\{x\}) \neq \varnothing$. One can then build a single set update Φ^* such that $\Phi^*(X) = \Phi(X) \cup \Phi'(X)$. It results that $\varphi(\Phi^*) = \varphi(\Phi) \cup \varphi(\Phi')$, thus the updating mode can be assimilated to Φ^*. Remark that limit sets of the generated dynamics $\varphi(\Phi)$ can be characterised as the $\subseteq$-smallest sets of configurations $X \subseteq \mathbb{B}^n$ such that $\Phi(X) = X$.

As a first illustration of set updates and how they can characterise updating modes, consider the following set update for Boolean networks of dimension n:

$$\Phi_{\mathsf{a}}(X) = \{\phi_W(x) \mid x \in X, \varnothing \neq W \subseteq [\![n]\!]\}.$$

This set update generates exactly all the non-empty elementary transitions: $\varphi(\Phi_{\mathsf{a}}) = \to_{\mathsf{a}}$. Thus, Φ_{a} characterises the asynchronous updating mode.

Similarly, let us now consider the following set update:

$$\Phi_{\mathsf{fa}}(X) = \{\phi_i(x) \mid x \in X, i \in [\![n]\!]\}.$$

Remark that $\varphi(\Phi_{\mathsf{fa}}) = \to_{\mathsf{fa}}$, i.e. Φ_{fa} characterises the fully asynchronous updating mode.

Context-dependent updating modes Set updates enable specifying elementary updates which may depend on the state of automata in a given configuration. As an illustration, we show here an explicit characterisation of the Boolean dynamics of Boolean networks with memory mentioned in the previous section.

Recall from Definition 4 that a Boolean network with memory f of dimension n is parameterised with a delay vector $d \in \mathbb{N}^n$. Its configurations are the couples of binary and discrete configurations $X_{(f,d)}$ such that each automaton i having state 0 in the binary part has state 0 in the discrete part, and each automaton i having state 1 in the binary part has a state between 1 and d_i in the discrete part. First, let us define as $\mathsf{mem}(x)$ the set of all memory configurations which can be associated with binary configuration $x \in \mathbb{B}^n$, and conversely, let us denote by $\mathsf{bin}(\delta)$ the binary configuration corresponding to memory configuration $\delta \in \mathbb{N}^n$. Formally:

$$\begin{aligned} \forall x \in \mathbb{B}^n, \quad & \mathsf{mem}(x) = \{\delta \in \mathbb{N}^n \mid x_i = 0 \iff \delta_i = 0, x_i = 1 \iff \delta_i \in [\![d_i]\!]\}, \\ \forall i \in [\![n]\!], \quad & \mathsf{bin}(\delta)_i = \min\{\delta_i, 1\}, \\ \forall x \in \mathbb{B}^n, \forall \delta \in \mathsf{mem}(x), \quad & \mathsf{bin}(\delta) = x. \end{aligned}$$

It appears that $X_{(f,d)} = \{(\mathsf{bin}(\delta), \delta) \mid \delta \in \mathbb{N}^n, \forall i \in [\![n]\!], \delta_i \in \{0, \dots, d_i\}\}$. Thus one can reformulate the original definition by considering the deterministic parallel update of memory configurations $\delta \in \mathbb{N}^n$, and replacing x with $\mathsf{bin}(\delta)$: an automaton $i \in [\![n]\!]$ is set to state d_i whenever its local function f_i is evaluated to 1 on the corresponding binary configuration $\mathsf{bin}(\delta)$; otherwise, its state is decreased by one, unless it is already 0. In particular, one can define the deterministic memory update $\phi^*_d : \mathbb{N}^n \to \mathbb{N}^n$ such that, for each $i \in [\![n]\!]$,

$$\phi^*_d(\delta)_i = \begin{cases} 0 & \text{if } f_i(\mathsf{bin}(\delta)) = \mathsf{0} \text{ and } \delta_i = 0, \\ \delta_i - 1 & \text{if } f_i(\mathsf{bin}(\delta)) = \mathsf{0} \text{ and } \delta_i \geqslant 0, \\ d_i & \text{if } f_i(\mathsf{bin}(\delta)) = \mathsf{1}. \end{cases}$$

Let us now extend the above definitions to sets:

- $\forall X \subseteq \mathbb{B}^n$, $\textsf{Mem}(X) = \bigcup_{x \in X} \mathsf{mem}(x)$;
- $\forall D \subseteq \mathbb{N}^n$, $\textsf{Bin}(D) = \{\mathsf{bin}(\delta) \mid \delta \in D\}$; and
- $\forall D \subseteq \mathbb{N}^n$, $\Phi^*_d(D) = \{\phi^*_d(\delta) \mid \delta \in D\}$.

The memory set update can then be defined for any set of configurations $X \subseteq \mathbb{B}^n$ by first generating the set of corresponding memory configurations, then applying the deterministic update on them, and finally converting them back to binary configurations:

$$\Phi_d(X) \;=\; \textsf{Bin} \;\circ\; \Phi^*_d \;\circ\; \textsf{Mem}(X).$$

With this formulation, one can see that the memory updating mode, being the projection of configurations of Boolean networks with memory on their binary part, leads to non-deterministic dynamics. Indeed, whenever a configuration gets mapped to several possible memory configurations, and whenever for two of these configurations δ and δ', there is an automaton $i \in [\![n]\!]$ where $\phi_d(\delta)_i = 0$ and $\phi_d(\delta')_i \geqslant 1$. This can occur if and only if $d_i \geqslant 2$, $x_i = 1$, and $f_i(x) = 0$. Thus, the memory updating mode of Boolean networks can equivalently be parameterised by a set of automata $\bar{d} = \{i \in [\![n]\!] \mid d_i \geqslant 2\}$ and defined as the following set update:

$$\Phi_{\bar{d}}(X) = \left\{\phi_W(x) \mid x \in X, W \subseteq [\![n]\!], W \supseteq \{i \in [\![n]\!] \mid i \notin \bar{d} \text{ or } f_i(x) = 1\}\right\}.$$

Remark that this definition no longer relies on delay configurations in $\mathbb{N}^n$. Overall, the memory updating mode of Boolean networks can be understood as a particular set of elementary transitions which depends on the configurations: the state changes from $\mathsf{1}$ to $\mathsf{0}$ of automata in $\bar{d}$ are applied asynchronously, whereas the other state changes are applied in parallel.

Beyond elementary and non-elementary updates Boolean networks are often employed as abstractions of a more detailed *concrete* model which would account for non-Boolean states, e.g. make them correspond to concentrations or copy numbers of biological molecules, or speed and delay of state changes of automata. Here, a Boolean model has the advantage of offering a simplified coarse-grained view of the concrete dynamics, and requiring much less parameters.

However, then, the question arises of how faithful are Boolean dynamics with respect to the quantitative dynamics, from a formal standpoint. It has been recently underlined in [24, 109] that the elementary and non-elementary transitions of Boolean networks are not complete enough to capture particular quantitative trajectories. With a fixed logic, and starting from similar configurations, the quantitative system shows that an automaton can eventually get activated, whereas none of the elementary and non-elementary dynamics of the Boolean network can reproduce this behaviour.

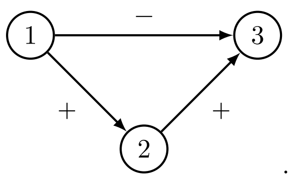
Thus, recently, several updating modes generating transitions which are neither elementary nor non-elementary have been introduced, enabling to capture delays in the change of automata states [24, 23, 109, 110]. They result in set updates Φ where, for some Boolean networks of dimension n and for some configurations $x \in \mathbb{B}^n$, there is $k \in \mathbb{N}$ such that there exists $y \in \Phi^k(\{x\})$ whereas $x \not\to^*_{\mathsf{e}} y$.

In this chapter, we focus on the *most permissive* updating mode of Boolean networks. "Permissive" refers to the transition relation it generates. It has been proven in [109] that the most permissive updating mode captures *all* trajectories which can be achieved by any quantitative model in agreement with the Boolean network specification. We will come back more formally to this notion in Section 3.4. In [110], we also detail set update formalisation of the *Interval* updating mode [24, 23] of Boolean networks which accounts for a specific type of delay in the state updates which generate transitions being neither elementary nor non-elementary.

Most permissive updating mode Consider the case whenever the state of an automaton i is used to compute the state of two distinct automata j and k, and assume that i is increasing from 0. During its increase, there are times when i may be high enough to trigger a state change of j but not (yet) high enough for k. This can be illustrated on a concrete biological example, the so-called *Incoherent Feed-Forward Loop of type 3* (I3-FFL) [88]: a Boolean network f of dimension 3 with

$$f_1(x) = 1 \qquad f_2(x) = x_1 \qquad f_3(x) = \neg x_1 \wedge x_2.$$

Its influence graph $G(f)$ is as follows:



Starting from configuration $(0,0,0)$, the asynchronous updating mode predicts only the following non-reflexive transitions: $(0,0,0) \to_{\mathsf{a}} (1,0,0) \to_{\mathsf{a}} (1,1,0)$. However, it has been observed experimentally [125] and in quantitative models [78, 123] that depending on reaction kinetics, one can actually activate transiently automaton 3. Essentially, the idea is that during the increase of the state of automaton 1, there is a period of time where automaton 1 is high enough so that automaton 2 can consider it active, i.e. $x_1 = 1$, but automaton 3 still considers it inactive, i.e. $x_1 = 0$. Then, the state of automaton 2 can increase, and so does the state of automaton 3. Leaving the Boolean network framework, one could model such a behaviour using 3 ordered states for automaton 1: 0, $\frac{1}{2}$, 1. Then, automaton 2 can be updated to state 1 whenever $x_1 \geqslant \frac{1}{2}$ and automaton 3 whenever $(x_1 < 1) \wedge (x_2 = 1)$. In other words, when automaton 1 is mild active, automaton 2 can become active, and then automaton 3 as well... until automaton 1 becomes fully active, inhibiting automaton 3. But, this activation of automaton 3 cannot be predicted at the Boolean level by combining the deterministic updates defined so far, whereas the logic encoded by f is correct.

Without introducing any parameter, the most permissive updating mode captures these additional dynamics by accounting for *all* possible threshold orderings, and all updates which can happen over a switch of a Boolean state. In some sense, the most permissive updating mode abstracts both the quantitative domain of automata and the duration of state changes. Their original definition [109] is based on the introduction of pseudo *dynamical* states, namely increasing and decreasing. An automaton can change from 0 to increasing whenever it can interpret the state of the other automata so that its local function is satisfied. Once in increasing state, it can change to the state 1 without any condition, or to the decreasing state whenever it can interpret the state of other automata so that its local function is not satisfied. Whenever an automaton is in a dynamical state, the automata can freely interpret its state as either 0 or 1.

The most permissive updating mode can equivalently be expressed in a more standard way by the means of composition of set updates [110]. This definition relies on the notion of sub-hypercubes. A sub-hypercube of dimension n can be specified by a vector $h \in \{0,1,*\}^n$ where components of the vectors having value $*$ are *free*: the vertices of the sub-hypercube are all the binary vectors $x \in \mathbb{B}^n$ such that for each dimension $i \in [\![n]\!]$, either $h_i = *$ or $h_i = x_i$. Thus, if we project a sub-hypercube over the d dimensions which are free, we obtain the hypercube of dimension d. For instance, the sub-hypercube for $h = (0,*,*)$ has 4 vertices: $(0,0,0)$, $(0,0,1)$, $(0,1,0)$, and $(0,1,1)$.

Given a configuration x, the computation of the next configurations according to the most permissive updating mode is done in two stages. A first stage consists in computing all the elementary updates of a single automaton and then widening the resulting set. The *widening* is defined using function $\nabla : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ which computes the vertices of the smallest sub-hypercube containing the given set of configurations. For instance, $\nabla(\{(0,0,1),(0,1,1)\}) = \{(0,0,1),(0,1,1)\}$, and $\nabla(\{(0,0,1),(0,1,0)\}) = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1)\}$. Given a set of automata W, the *widening set update* $\Phi_{W,\nabla} : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ applies this operator on the results of the elementary set update, or equivalently with the fully-asynchronous set update, on the automata of W. This widening is re-iterated until a fixed point is reached. Then, a *narrowing* $\Lambda_W : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ filters the computed configurations X to retain only those where the states of automata in W can be computed with f from X.

Definition 8. *The* most permissive set update Φ_{MP} *of a Boolean network of dimension n is given by*

$$\Phi_{\mathsf{MP}}(X) = \bigcup_{W \subseteq [\![n]\!]} \Lambda_W \circ \Phi^{\omega}_{W,\nabla}(X),$$

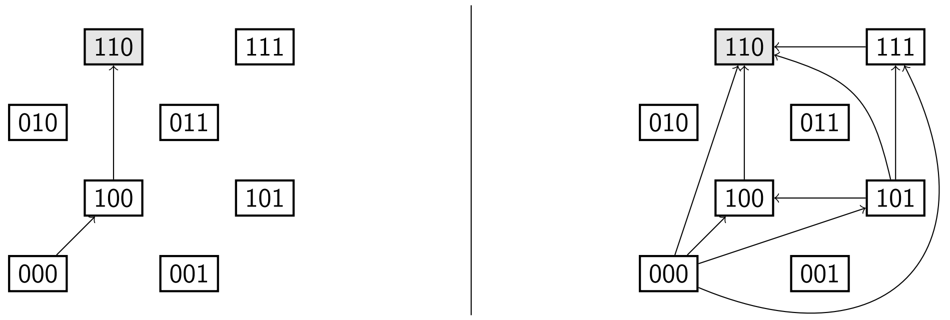


Figure 10: (left) Asynchronous, and (right) most permissive dynamics *reachable from the configuration* $(0, 0, 0)$ for the Boolean network f of the I3-FFL defined page 23. For the sake of readability, loops are omitted. In the two examples, there are loops on each configuration involved in the displayed transitions.

where, for any $X \subseteq \mathbb{B}^n$ *and any* $W \subseteq [\![n]\!]$*:*

$$\begin{aligned} \nabla(X) &= \textstyle\prod_{i=1}^{n}\{x_i \mid x \in X\} \\ \Phi_{W,\nabla}(X) &= \nabla(X \cup \{\phi_i(x) \mid x \in X, i \in W\}), \\ \Lambda_W(X) &= \{x \in X \mid \forall i \in W, \exists y \in X,\ x_i = f_i(y)\}, \end{aligned}$$

and where $\Phi^{\omega}_{W,\nabla}(X)$ *denotes the iteration of* $\Phi_{W,\nabla}$ *until reaching a fixed point (remark that for any* $k \in \mathbb{N}$, $\Phi^{k}_{W,\nabla}(X) \subseteq \Phi^{k+1}_{W,\nabla}(X) \subseteq \mathbb{B}^n$*).*

Example 2. Figure 10 shows the dynamics generated from the configuration 000 by the asynchronous and most permissive updating mode on the Boolean network of the I3-FFL defined page 23. By instantiating Definition 8, we obtain:

$$\begin{aligned} \Phi_{\{1,2,3\},\nabla}(\{(0,0,0)\}) &= \nabla(\{(0,0,0),(1,0,0)\}) = \{(0,0,0),(1,0,0)\}, \\ \Phi^2_{\{1,2,3\},\nabla}(\{(0,0,0)\}) &= \nabla(\{(0,0,0),(1,0,0)\} \cup \{(1,1,0)\}) = \{(0,0,0),(1,0,0),(0,1,0),(1,1,0)\}, \\ \Phi^3_{\{1,2,3\},\nabla}(\{(0,0,0)\}) &= \nabla(\{(0,0,0),(1,0,0),(0,1,0),(1,1,0)\} \cup \{(0,1,1)\}) = \mathbb{B}^n, \\ \Lambda_{\{1,2,3\}}(\mathbb{B}^n) &= \{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}. \end{aligned}$$

Thus, $(1,1,1) \in \Phi_{\mathsf{MP}}(\{(0,0,0)\})$, whereas $(0,0,0) \not\to^*_{\mathsf{e}} (1,1,1)$, i.e. configuration $(1,1,1)$ is reachable from $(0,0,0)$ using the most permissive updating mode, whereas there are no elementary (and non-elementary) paths between these two configurations.

Finally, let us conclude by listing some basic properties of the most permissive updating mode:

1. It preserves the fixed points of f: for any configuration $x \in \mathbb{B}^n$, $f(x) = x$ if and only if $\Phi_{\mathsf{MP}}(x) = \{x\}$.
2. It subsumes elementary transitions: $\to_{\mathsf{e}} \subseteq \varphi(\Phi_{\mathsf{MP}})$.
3. Its transition relation is reflexive and transitive: $\Phi_{\mathsf{MP}} = \Phi^2_{\mathsf{MP}}$.
4. (by 2 and 3) Its transition relation subsumes non-elementary transitions: $\to^*_{\mathsf{e}} \subseteq \varphi(\Phi_{\mathsf{MP}})$.
5. (by 4 and the example) There exist Boolean networks such that the most permissive transition relation is strictly larger than non-elementary transitions, i.e. there exist $x, y \in \mathbb{B}^n$ such that $y \in \Phi_{\mathsf{MP}}(\{x\})$ but $x \not\to^*_{\mathsf{e}} y$.

3 Biological case studies

As discussed in the introduction of this chapter and in Section 2.1, Boolean network features have been particularly appreciated to apprehend regulation network modelling from both theoretical and applied standpoints. As explained, this comes notably from their setting simplicity and their abstraction level which allow to focus on the natural computations operated by the modelled systems and thus deduce or predict properties of the latter.

In general, computational models of biological systems are appealing for bringing potential explanations to observed phenomena, and for performing experiments *in silico*, which may help prioritising wet lab experiments [51]. Biological processes involve a myriad of features, such as the shape of cells, the shape of molecules, their location, their movement, the way they interact, they way they fold, etc. Thus, to be manageable and keep some interpretability of the results, a model of a biological process is first a harsh selection of the biological features. A model which is able to reproduce an observed phenomenon thus supports an hypothesis according to which the selected features and the granularity of their representation are sufficient to explain the phenomenon. With this perspective, Boolean networks offer a very high abstraction of the system, focusing on the structure of the causal influences between the components of the network, and without considering explicitly quantitative features such as the chronological time and quantities of molecules. The Boolean network setting specifies the logic of automata state changes, whereas the updating mode specifies the logic of the orchestration of state changes over time, potentially abstracting features related to time and quantities. Moreover, because of their discrete and finite nature, one can obtain an exhaustive assessment of their trajectories and limit sets which are very often theoretical representatives of real observable physiological characteristics. This provides a powerful tool to formally demonstrate that some behaviours are impossible to reproduce with a given Boolean network and updating mode.

In this section, we summarise a selection of case studies for which the updating mode plays a central role:

- the robustness of predictions of limit sets and trajectories leading to them when varying the updating mode with the modelling of the floral morphogenesis of *Arabidopsis thaliana* (Section 3.1);
- the definition of custom updating modes to reflect time constraints and accurately reproduce sequences of state changes within the cell cycle (Section 3.2) and for biological rhythms synchronisation (Section 3.3); and
- the formal analysis of the absence of trajectories whenever the Boolean network is considered as an abstraction of a quantitative system, by relying on the most permissive updating mode and illustrated on an incoherent feed-forward loop system (Section 3.4).

3.1 Floral morphogenesis of *Arabidopsis thaliana*

The floral development of *Arabidopsis thaliana* is among the model organisms the most studied in vegetal biology since the early 1900s, and the first plant whose nuclear genome was sequenced [134].

The first model based on Boolean networks introduced in the literature for the genetic network of the floral development of this plant comes from [93]. It is based on a threshold Boolean network, namely a Boolean network whose local functions are threshold functions, composed of 12 automata, each one corresponding to a particular gene or gene complex acting on the control of the floral morphogenesis. In this seminal paper, the dynamics of this network is studied according to a specific block-sequential updating mode. This mode was chosen arguing its pertinence according to the activation time of the genes in the flowering and the flower morphogenesis processes. One year later, together with Thieffry, the same authors transformed this model into a classical Boolean network, i.e. with no threshold local functions, and used the method developed by [140] to study its fully asynchronous dynamics [94].

These two distinct choices, for both the mathematical model at stake and the updating mode, led the authors to highlight that the underlying dynamical systems converge towards six fixed points. More interestingly, among these six fixed points, four correspond to the four floral organs of the *Arabidopsis thaliana*: sepals, petals, carpels and stamens; the two remaining fixed points correspond to *(i)* inflorescence cells of the plant which do not belong to its flowers, and to *(ii)* an expression pattern never observed in wild-type plants until now which could be produced experimentally.

These works emphasised that qualitative modelling with Boolean networks is helpful, in the sense that their limit sets may capture real biological structures, such as organs, tissues or cellular types, as evoked by [40]. However, from a more theoretical standpoint, further work need to be done to achieve a deep understanding of the network itself. [42], basing themselves on the initial threshold Boolean network model, wanted to understand the network in depth. First, they highlighted that there exist updating modes, e.g. the parallel updating mode, for which the underlying dynamical systems admit not only the former fixed points but also limit cycles. Then, an analysis of the local functions led them to construct a simplified / compressed network with the same asymptotic dynamics, and to show that the asymptotic richness depends mainly on five genes which are parts of 2 distinct symmetric strongly connected components which can actually be considered as the dynamical complexity engines of the network. This actually explained the possible existence of limit cycles, by applying a result of [66]. The natural question then was: now we can state the phenomenological sense of the six fixed points, what is the biological likeliness of the limit cycles? The answer to this question was given by the asynchronous dynamics, i.e. all the possible elementary transitions, of the model which highlights their unlikeliness, by showing that the probabilities to enter and stay into an asymptotic oscillation is very weak. Eventually, these previous results were complemented and refined in 2020 by an analysis of the dynamics of this network when modelled by a Boolean network with memory [63].

Finally, we can notice that after two decades, the scientific community eventually gets a sharp understanding of this simple network modelling the floral development of *Arabidopsis thaliana* from both theoretical and applied standpoints, thanks notably to a particular attention paid to updating modes. This highlights the importance of considering a consistent updating mode depending on the question addressed, and that any updating mode may be *a priori* relevant.

3.2 Cell cycle

Together with the modelling of cell differentiation and reprogramming processes which occupy a vast part of Boolean network models in biology, the modelling of cellular rhythms and oscillators is another prominent application of Boolean networks. In such systems, we expect that the sustained oscillatory behaviours are captured by the Boolean network as non-singleton limit sets. The system is usually assumed to be in a configuration of this limit set.

One of the most studied biological oscillator is the cell cycle. The cell cycle refers to the successive divisions of a cell, which goes through a specific and well characterised sequence of events, including the duplication of its genome (so-called S phase, for Synthesis) and its mitosis (M phase). Boolean network models of cell cycles usually focus on reproducing accurately this sequence of events which can be observed by the activity of specific proteins, as well as its control and coupling with other biological processes.

The cell cycle involves numerous genes and proteins and is a tightly regulated process. It is also expected that the chronological time, notably related to the speed of different interactions, plays a crucial role in this regulation. With this latter consideration, an accurate modelling using Boolean networks seems challenging.

The Boolean model of [55] is a nice demonstration of how updating modes enable abstracting time adequately and obtain a realistic dynamical model. Their model gathers well-known influences between key regulators of the cell cycle, which have been extensively studied with quantitative models. These regulators involve different types of proteins (including cyclins, transcription factors, complexes, inhibitors), the activity of some of them being well known markers for the different phases. In total, the model comprises 10 automata.

In their study, Fauré first compared the limit cycles obtained with the parallel and fully asynchronous updating modes when the input automaton of the network (*CycD*) is activated. The parallel updating mode leads to one limit cycle between 7 configurations, which matches with available data and previous quantitative models. However, many state changes are compressed into a single transition, making difficult a finer analysis on the possible sequences of events during the cell cycle. The fully asynchronous updating mode then enables results in a single limit cycle of 112 configurations which contains well detailed descriptions of the sequences of events of the cell cycle. However, the limit cycle contains many spurious trajectories, notably with shortcuts avoiding key checkpoints of the cell cycle. In order to refine the Boolean dynamics, the authors proposed a custom updating mode which can be seen related to the Memory Boolean networks and block-sequential updating modes studied in this chapter. The main principle is to group the state changes of automata into different priority classes, and further split these priority classes into automata which are updated according to the fully-asynchronous updating mode and other ones which are updated with the parallel updating mode. The priority classes allow treating more accurately the fast and slow biological processes. These classes are then refined further by identifying state changes which are controlled by the same regulatory process, and thus should happen simultaneously. The obtained dynamics consists then in a selection of elementary transitions which depend on the state changes of automata. In the scope of the cell cycle model, this translates into the prediction of a limit cycle of 18 configurations with a finer description of sequences of cell cycle events, and for which the different trajectories cannot be discriminated with the current knowledge of the biological process.

3.3 Vegetal and animal zeitgebers

Despite the fundamental role of updating modes discussed so far in this section related to biological case studies, whatever they are block-sequential, fully asynchronous, or even asynchronous, and the large spectrum of questions they can help us answer, it turns out that they do not necessarily own the relevant properties to capture some biological phenomenological intricacies. Indeed, whilst they allow in some sense to modulate the internal clocks of a regulation process modelled by a network, they do it by considering the network as a whole, with no distinction between its own components.

In [47], the authors focused on Zeitgebers. Zeitgebers are classically defined in biology and medecine as exogenous cues which synchronise endogenous biological rhythms. In the framework of genetic regulation network modelling, considering Zeitgebers as sorts of timers of genetic or physiological origin, somehow external to the very functional components of the network, they highlight that the latter cannot be modelled by means of the until now classically studied updating modes mentioned above, for the reasons evoked. Among the best known examples of a Zeitgeber here, namely a subnetwork having its own clock which synchronises the dynamics of the whole network which contains it, is certainly the role played by genes PER and TIM which work together to control the circadian rhythm of *Drosophila melanogaster* [72, 126, 62].

However, using threshold Boolean networks both as a genetic network model and as a neurophysiologic network model, [47] showed that some deterministic and periodic updating modes belonging to the local clocks family, namely the block-parallel updating modes, are good candidates to capture synchronisation phenomena, in the sense that they are expressive enough, which is essential for the biological modelling standpoint, without being too expressive, which is important for the mathematical and computational standpoint in order to stay away from a combinatorial explosion argument. More precisely, they emphasised schematic models taking into account the existence of specific genetic regulation subnetworks acting as Zeitgebers in two distinct contexts:

- the genetic control of plant growth — they proposed a model of vegetal growth by considering the regulatory relations between the circadian clocks genes CCA1 and TOC1 and abstract genes associated with auxin flows corresponding to the localised expressions of auxin [135, 35, 16]. In this model, the subnetwork composed of CCA1 and TOC1 plays a timer role which synchronises the regulation of the other component acting as the functional auxin part of the network serving the plant growth, thanks to a specific block-parallel updating mode. The dynamical role of this timer is to produce a regular scheme of growth, which seems to coincide pretty much with the quasi-perfect morphogenesis of plants like *Araucaria araucana*.

- the cardio-respiratory pace — they proposed a model of the clock governing the cardio-respiratory regulation at the neuro-physiological level. Four components are considered which are derived from [15, 49, 96, 43]: inspiratory and expiratory neurons which play the role of a timer synchronising the sinoatrial node and the baroreceptor of the heart. Together with an adapted block-parallel updating mode, the underlying network behaves in two phases as it can be observed in nature. Nevertheless, even if the timer modelling seems to comply with the biological assumptions and observations, it is not exactly the case of the cardiac activity modelling, which paves the way to further refinements.

3.4 Abstraction of quantitative models

In many applications, the Boolean modelling of biological systems abstracts both the quantitative time of the interactions and the quantitative state of the interacting automata. This is notably the case when we consider gene regulatory networks, where the activation of a gene is generally assumed to depend on a sufficient amount of transcription factors. Whereas this dependency is often assumed to be non-linear and modelled as a threshold function, it is nevertheless a quantitative process. Indeed, differences of thresholds between the different gene activations may play an important role in several biological processes. The Boolean abstraction may also hinder the validation of the model with respect to observations of the biological system. The current experimental observations mostly rely on counting the population of different molecules either cell-per-cell (single-cell measurement technologies), or in a mixture of cells. Actually, the activity of genes is often linked to the amount of their transcripts, and is thus a quantitative feature. A qualitative interpretation of such data then relies again on delineating thresholds to determine the Boolean state of automata.

As illustrated by the previous case studies, the updating mode of Boolean networks are effectively employed to model features related to the time or speed of different Boolean processes. In these case studies, the updating modes actually select specific elementary paths of the Boolean network dynamics, avoiding predicting spurious transitions. In a way, they assume that the asynchronous dynamics (generating elementary transitions) gives a boundary on the admissible paths, or equivalently on the admissible non-elementary transitions: if there is no elementary path from a configuration x to a configuration y, then, none of the updating modes considered in the above case studies can generate transitions which would connect these two configurations.

Given the abstraction level imposed by Boolean networks for modelling biological systems, one may wonder whether the assumption that the elementary and non-elementary transitions capture its state changes holds in general, i.e. if there exist cases of systems for which the asynchronous Boolean network misses transitions. The answer to this question heavily depends on the type of systems we want to model. In this section, we focus on quantitative systems, such as multivalued networks [141] or ordinary differential equations [61, 39]. From a formal standpoint, this question relates to the correctness of the abstract interpretation of a quantitative system by a Boolean network.

The incoherent feed-forward loop system I3-FFL developed page 23 is a simple counter-example showing the incompleteness of elementary and non-elementary transitions of Boolean networks. As discussed above, several quantitative models and experimental studies show that the output automaton of the system can be transiently activated from the configuration where all automata are inactive. However, the asynchronous dynamics shows no such paths. From a model validation perspective, this is critical, as the Boolean network would likely be considered as incoherent with the data, whereas the logic it encodes is correct. And indeed, the observed behaviour can be recovered using the most permissive updating mode without any additional parameter.

[109] have demonstrated that the most permissive updating mode leads to a complete and minimal abstraction of state changes of any quantitative model being a *refinement* of the Boolean network. In other words, if there is no path between two configurations in the most permissive dynamics, then no quantitative refinement of the model can create this path. The definition of a *refinement* is obviously key. Let us first give a general definition of a quantitative model: a quantitative model F can be

defined as a function mapping discrete or continuous configurations to the derivative of the state of automata. Then, F is a refinement of f if and only if the derivative of automaton i is strictly positive (resp. negative) in a given quantitative configuration z only if there is a binarisation $\tilde{z}$ of z so that $f_i(\tilde{z}) = \mathtt{1}$ (resp. $\mathtt{0}$). The binarisation has to map the quantitative 0 to the Boolean $\mathtt{0}$, and the maximal quantitative state, e.g. with multivalued networks, to the Boolean $\mathtt{1}$; otherwise any Boolean value can be considered.

One may remark that the updating mode which enables the transition from any configuration to any other ones verifies the completeness criteria. However, in addition to being non-minimal, such dynamics would make impossible any interesting prediction from the model. As we mentioned several times, Boolean networks are employed to model differentiation processes, where the system can have multiple limit behaviours. Predicting which attractors the system may reach, and how to control this reachability is a prominent application. However, these predictions rely on the absence of paths between certain configurations of the Boolean network. Thus, an updating mode generating too many spurious path between configurations would largely hinder such predictions. [109] have shown on several case studies that the most permissive updating mode predicts the same reachable attractors and control as identified in previous studies with the fully asynchronous updating mode. Thus, the most permissive updating mode brings a formal abstraction of quantitative systems, with strong guarantees of capturing possible stage changes, while being stringent enough to capture differentiation processes. Moreover, as we well detail in Section 4.2, the analysis of the most permissive dynamics requires a much lower computational cost than elementary and non-elementary updating modes, enabling addressing genome-scale models with several thousands of automata [109].

4 Fundamental knowledge

Beyond their wide use in the context of qualitative modelling of biological networks which has led to numerous significant advances about genetic regulation, Boolean networks keep being studied *per se* from the theoretical (mathematical and computational) standpoint, notably because we are still far from understanding in depth their underlying properties. For instance, despite real progresses realised these last two decades, the way that information is transmitted along the automata as well as the ability to produce this or that global dynamical behaviour depending on local interactions, are still not well understood. Trying to answer this kind of questions needs to focus on distinct discrete structures (*e.g* influence graphs, sets of local functions, transition graphs, updating modes, etc.) and to relate them by means of methods and tools coming from different areas of mathematics and computer science like combinatorics, dynamical system theory, algorithms, computational complexity theory, and computability theory. In this section, in order to give an insight on some relevant fundamental problems on Boolean networks (and more generally on automata networks), some seminal results obtained are presented following two research lines: the links between static (syntactic) and dynamical (semantic) properties of Boolean networks, and the inherent complexity of some classical problems underlying Boolean network studies.

4.1 Structural properties and attractors

Given an influence graph or a Boolean network, knowing the underlying possible asymptotical dynamics is a problem which is known to be complex from the computational standpoint (see Section 4.2 below). Nevertheless, past studies gave very general and important results on the subject. This subsection aims to present some of these results, while emphasising links with the role of updating modes.

4.1.1 Fixed points stability

The result presented hereafter is certainly the most classical of discrete and finite dynamical systems. It concerns fixed points and their relative stability depending on updating modes, and holds for any kind of finite dynamical system. More precisely, Theorem 1 below states that, considering any Boolean network f, the parallel fixed points of f are necessarily fixed points of (f, μ), for any updating mode μ which governs the evolution of f by organizing the executions of associated updating functions over time.

Theorem 1 (Folklore)**.** *Let $f : X \to X$ be an automata network whose configurations are elements of X. Let π be the parallel updating mode. Let $\mathrm{FP}(f) = \mathrm{FP}(f, \pi)$ be the set of fixed points of dynamical system (f, π). Then we have, for any μ defining an organisation of updating functions over time:*

$$\mathrm{FP}(f) \subseteq \mathrm{FP}(f, \mu).$$

Proof. Consider a Boolean network f of dimension n, and $x \in \mathbb{B}^n$ a fixed point of (f, π). Since $x = (x_1, \ldots, x_n)$ is a fixed point of (f, π), $x_1 = f_1(x)$ (resp. $x_2 = f_2(x), \ldots, x_n = f_n(x)$) is a fixed point of local function f_1 (resp. $f_2, \ldots, f_n$). So, since x is actually a vector composed of the fixed points of the n local functions, whatever the way an updating mode μ updates states from x, it cannot change them, by definition. In other words, $\forall W \subseteq [\![n]\!], \phi_W(x) = x$, which is the expected result. □ □

A first remark is that the reciprocal of this theorem does not hold. Indeed, there exists an automata network $f : X \to X$ and pertinent updating mode μ such that $\mathrm{FP}(f, \mu) \nsubseteq \mathrm{FP}(f)$. For instance, consider the Boolean network of dimension 3 defined as $f(x) = (f_1(x) = x_3, f_2(x) = x_1, f_3(x) = x_2)$ whose associated influence graph is a cycle of length 3 whose every arc is positively signed. If this Boolean network evolves according to the parallel updating modes, then it admits two fixed points such that $\mathrm{FP}(f) = \{(0, 0, 0), (1, 1, 1)\}$. Now, let $\mu = (\{2, 3\}, \{1, 3\}, \{1, 2\})$ be a pertinent periodic updating mode. Notice that μ is not a local clocks updating mode. Then, it is easy to compute the dynamics of (f, μ) and to conclude that this dynamical system has four fixed points and is such that $\mathrm{FP}(f, \mu) = \{(0, 0, 0), (0, 1, 0), (1, 0, 1), (1, 1, 1)\}$. As a consequence, even if playing with updating modes cannot lead to destroy fixed points according to Theorem 1, it can lead to create fixed points. Although it has never been addressed in depth, this fixed point generation by updating modes is very interesting and certainly deserves further analyses.

A second remark is that we could imagine other kinds of updating modes creating transitions which are not the result of local function executions, like transitions imposed by some kind of external events / behaviours to the underlying network itself. In that case, there would be no guaranty to conserve the "canonical" fixed points of f.

4.1.2 Feedback cycles as engines of dynamical complexity

The previous result related to fixed points stability emphasises a strong intrinsic dynamical property of finite dynamical systems, and thus of Boolean networks. Of course, such a property is very important and pertinent. However, we have to keep in mind that, in the context of modelling by means of Boolean networks, in most situations, the inputs of the problems that are to be addressed are either Boolean networks, i.e. their definitions as collections of local functions, or influence graphs (or even communication graphs in which the interactions are not necessarily effective). As a consequence, it is natural to ask for finding / characterising structural properties on these latter objects which induce relevant properties about their underlying possible dynamical systems.

The literature offers us several results relating structural and dynamical properties of Boolean networks. Among the most classical ones is the theorem of [121] stating the crucial role of feedback cycles in the influence graphs of automata networks for them to have a non-trivial dynamics. Theorem 2 below is the adaptation to Boolean networks of the seminal theorem of Robert, and states that the presence of a feedback cycle in the influence graph of a Boolean network f is a necessary condition for f to admit several limit sets.

From a purely theoretical standpoint, any updating mode which is mathematically correct is reasonable but, if we consider updates in a context of modelling, some constraints need to be taken in account. In this framework, consider the following definition: an updating mode is called *reasonable* if its application leads every local function to be executed infinitely often[1].

Theorem 2 ([121]). *Let f be a Boolean network of dimension n and $\mathscr{G}_f = (\llbracket n \rrbracket, E_f)$ its associated influence graph. If $\mathscr{G}_f$ is acyclic, then for any reasonable updating mode μ, every configuration of dynamical system (f, μ) converges towards a unique attractor $y \in \mathbb{B}^n$ which is a fixed point.*

Proof. If $\mathscr{G}_f = (\llbracket n \rrbracket, E_f)$ is acyclic, then it is a directed acyclic graph (DAG) and, as a consequence, *(i)* at least one of its vertices is a source, i.e. a vertex with indegree 0, and *(ii)* it can be structured into layers defined recursively such that the layer of depth 0 contains all the sources, and the layer of depth k contains all the vertices which have at least one in-neighbour belonging to the layer of depth $k-1$. Notice that as soon as a vertex i is a source, i has no incoming influence. Thus, the state of its corresponding automaton does not depend on the state of any automaton in $\llbracket n \rrbracket$, which means that its local function is necessarily constant, namely $f_i(x) = 0$ or $f_i(x) = 1$. Moreover, since $\mathscr{G}_f$ is a DAG, it has at least one topological ordering, i.e. a total ordering of its vertices such that for each arc $(i, j) \in E_f$, i comes before j in the ordering. Let L be the topological ordering of $\mathscr{G}_f$ obtained by applying a variation of the Kahn's algorithm [79], where the initial set of sources is structured into a queue where vertices are ordered both increasingly depending on the depths of reachable layers and lexicographically according to $\llbracket n \rrbracket$, and reordered similarly when a vertex is removed from the queue. In other terms, L is the unique topological ordering which takes into account both the layered structure of $\mathscr{G}_f$ and the lexicographical order on $\llbracket n \rrbracket$. Consider now that L is organised according to the layers of $\mathscr{G}_f$ such that $L = (L_0, ..., L_\ell)$, where L_0 is the restriction of L to the sources of $\mathscr{G}_f$, and L_k is the restriction of L to the automata whose states only depend on automata which belong to layers between L_0 and L_{k-1}.

Now, let us consider any initial configuration $x \in \mathbb{B}^n$, and time step $t_0 \in \mathbb{N}$ after which all the automata of L_0 have updated their state for the first time. Since the updating mode μ is reasonable, t_0 exists. Since every automaton of L_0 is a source in $\mathscr{G}_f$, we have: for all $i \in L_0$, $x_i^{t_0} = f_i(x)$ and will remain fixed, i.e. $\forall t_k \geqslant t_0 \in \mathbb{N}, x_i^{t_k} = 0$ or $x_i^{t_k} = 1$. Consider now time step $t_1 > t_0$ after which all the automata of L_1 have updated their state for the first time. Again, for the same reason, t_1 exists. By definition, these automata only depend on automata of L_0, whose states are fixed since $t_1 > t_0$. As a result, from time step t_1, the state of every automaton of L_1 will remain fixed because it has updated its state by executing its local function whose variables are fixed. From this, by an induction on the layer depth of L and by applying the same reasoning on every $(L_k)_{2 \geqslant k \geqslant \ell}$, there exists time step t_ℓ after which the states of all automata of $\llbracket n \rrbracket$ remain fixed. Let $x' = f^{t_\ell}(x) \in \mathbb{B}^n$. Since x' is the image of any configuration after t_ℓ time steps, it is the unique fixed point of (f, μ). □ □

Despite the apparent simplicity of its statement and of its proof which can easily be extended to finite dynamical systems, Theorem 2 is certainly one of the most important theorems of dynamical system theory. (Remark that generalisations of it can be found in [128, 118].) Indeed, it emphasises that feedback cycles are actually sorts of fundamental engines to create asymptotic diversity. To go beyond, it is a formalisation and a concretisation of the key principle of cybernetics [145], the *circular causality.*

4.1.3 About signed feedback cycles

The previous result of [121] and its extensions highlight the major role played by feedback cycles on the asymptotic behavioural complexity of Boolean networks. At the same period, Thomas mentioned the importance of distinguishing cycles according to their nature.

[1]All the updating modes discussed in this chapter are reasonable. Notice that, obviously, the best example of a non-pertinent updating mode is $(\varnothing)$ which does not update any state. Any Boolean network evolving according to this updating mode admits 2^n fixed points because each of its configurations is stable by definition.

Let $\mathscr{C} = (\llbracket n \rrbracket, E)$ be a (signed) cycle of length n, where $E \subseteq \llbracket n \rrbracket \times \{-,+\} \times \llbracket n \rrbracket$. With no loss of generality, consider that the arcs of $\mathscr{C}$ are ordered according to the lexicographical order on $\llbracket n \rrbracket$ such that $E = \{(1,2),(2,3),\ldots,(n,1)\}$. Now, let us consider that $\mathscr{C}$ is the influence graph of a Boolean network f defined on $\llbracket n \rrbracket$ and rename it as $\mathscr{C}_f$. Obviously, since $\mathscr{C}_f$ is a cycle, the arity of every local function of f equals 1, so that:

$$\forall i \in \llbracket n \rrbracket \backslash \{1\},\ f_i(x) = \begin{cases} \neg x_{i-1} \\ x_{i-1} \end{cases} \quad \text{and} \quad f_1(x) = \begin{cases} \neg x_n \\ x_n \end{cases} .$$

The definition below discriminates Boolean network cycles depending on the signs of their arcs.

Definition 9. *Let $\mathscr{C}_f = (\llbracket n \rrbracket, E_f)$ be a cycle which represents the influence graph of a Boolean network f. $\mathscr{C}_f$ is said to be a* positive cycle *(resp. a* negative cycle*) if the number of negative arcs which compose it is even (resp. odd).*

Let us give an insight to understand the trajectory of a Boolean cycle configuration of dimension n: consider one automaton of the Boolean cycle as special in the sense that it is the unique observation point from which an observer can see the evolution of the dynamics of the cycle, and denote this automaton by $i \in \llbracket n \rrbracket$; now, imagine that at time step $t \in \mathbb{N}$, the state of i is $x_i^t = b$, and imagine that the cycle evolves according to the parallel updating mode. Then, information b will travel along the cycle as follows: at time step $t+1$, b (resp. $\neg b$) becomes the state of automaton $i+1$, except of course in the case where $i = n$ in which b becomes the state of automaton 1, if arc $(i, i+1)$ (or if arc $(n, 1)$ when relevant) is positive (resp. negative); at each following time step, the information keeps traveling along the cycle, until time step $t+n$ is reached. At this time step, if the observer evaluates the state of i, then:

$$x_i^{t+n} = \begin{cases} b & \text{if the cycle is positive} \\ \neg b & \text{if the cycle is negative} \end{cases} .$$

Indeed, only a negative cycle can negate the initial information after one complete round, thanks to the odd number of its negative arcs.

At the beginning of the 1980s, Thomas proposed two general rules related to these two types of cycles [137]:

1. The presence of a positive cycle in the influence graph of a dynamical system is necessary for this system to have several fixed points.
2. The presence of a negative cycle in the influence graph of a dynamical system is necessary for this system to have a limit cycle.

These two rules led to numerous studies aiming to prove their validity in different contexts. In particular, they gave rise to several theorems in both discrete and continuous settings. Whilst they are not the purpose of this chapter, we invite the readers to see [130, 68, 28, 131, 132] which deal with these theorems in the continuous case.

Let us come back now to the discrete case on which our focus is. Notice that the first rule was proven in more or less general frameworks, see:
- [7, 6] for the parallel Boolean case;
- [113] for the fully asynchronous Boolean case; and
- [117, 115] for the fully asynchronous discrete case.

Following these lines while focusing on the Boolean case, in [103, 127], the authors generalised the existing results to the asynchronous updating mode. Remind that, given a Boolean network, the asynchronous updating mode is a non-deterministic mode which authorises every possible transition inducing subsets of automata of this network, i.e. all the transitions made possible by any deterministic updating modes as well as the fully asynchronoous updating mode. This extension to the asynchronous case is in consequence quite pertinent. Indeed, it induces the validity of the first Thomas' rule for a very large set of updating modes (namely an infinite one actually).

Theorem 3 ([103]). *Let f be a Boolean network of dimension n, $\mathscr{G}_f = (\llbracket n \rrbracket, E_f)$ its associated influence graph, and let a be the asynchronous updating mode. If (f, a) has several fixed points, then there is a positive cycle in $\mathscr{G}_f$.*

Idea of the proof. The general idea of the reasoning rests on a proof by contradiction. Indeed, consider that $\mathscr{G}_f$ does not contain any positive cycle. Then, $\mathscr{G}_f$ is either acyclic or it admits at least one negative cycle so that, if it admits several ones, these cannot induce a positive cycle by hypothesis. Let us admit that $\mathscr{G}_f$ is acyclic. In this case, Theorem 2 applies and (f, a) has only one fixed point. Now, let us consider the other case. Without entering into the details here, the idea of the related proof is to show that a negative cycle Boolean network can never suppress all the local instabilities, an automaton i being unstable in a configuration x if and only if $f_i(x) \neq x_i$, and a local instability in x being the number of unstable automata of x. In particular, if a configuration has zero local instability, then it is a fixed point. Summarising, given any configuration $x \in \mathbb{B}^n$, there exists no asynchronous trajectory $x \to^*_{(f,\mathsf{a})} y$ such that y is a fixed point. This can be proven by focusing on the structure of the transition graph $\mathscr{D}_{(f',\mathsf{a})}$ of a cyclic negative Boolean network f' of dimension n'. Indeed, $\mathscr{D}_{(f',\mathsf{a})}$ is a layered graph in which each layer contains configurations of equivalent instability level, which allows to show that a configuration instability can be an odd number between 1 and n'. In other words, this induces that negative cycles cannot generate fixed points. As a consequence, the presence of a positive cycle in $\mathscr{G}_f$ is necessary for (f, a) to admit several fixed points. □

Concerning now the second rule about negative cycles, notice that it has been proven in [116] for the discrete fully asynchronous case. Whilst we are not going to enter into details here, an interesting fact concerning this rule is that it has also been generalised to the asynchronous case by [103] and [127]. However, contrary to the first rule developed above, the fact that it holds under the asynchronous updating mode does not induce that it does for all deterministic modes, because of the very nature of limit cycles and their high sensitivity to synchronism [67, 46, 105]. Notably, the dynamics in parallel of any positive cycle of length greater than 1 admits limit cycles.

Thomas' rules, together with Robert's theorem, put the highlight on the remarkable role of feedback cycles. Nevertheless, the results presented until now do not explain in depth the dynamics of positive and negative cycles *per se*. That is in particular why studies dedicated to cycles have been led. [112] proved that, under the fully asynchronous updating mode hypothesis, positive (resp. negative) cycles of length n admit exactly two fixed points x and $\overline{x}$ (resp. one limit cycle of length $2n$) as attractors. [46] gave then a complete combinatorial and dynamical characterisation of cycle dynamics in parallel which extends to every block-sequential updating mode by the main result of [65] that shows that computing a block-sequential cycle dynamics can be reduced to computing the parallel dynamics of a smaller cycle of same sign. A synthesis of these results on cycles can be found in [45]. Then, works focused on cycle tangential intersections to make a first step in the direction of comprehending how combinations of cycles operate in more complex networks [102, 103, 4].

The previous results succeeded in giving deep knowledge on Boolean cycles and their tangential intersections. These results suggested also that cycle intersections would be key structural elements to reduce drastically the degrees of freedom of Boolean networks and their ability to have too many attractors, which seems not to be likely empirically in biology for instance. Nevertheless, whilst acquiring knowledge (even complete) about these specific patterns forms a mandatory first step before going further, it is far from being sufficient to comprehend globally those of more complex networks, possibly composed of numerous intersections of cycles of different kinds. In this framework, the second part of Hilbert's XVIth problem [75] asking for the maximal number of limit cycles and their relative sizes in polynomial vector fields gives rise to addressing problems which are discrete variations of it: perform deep analyses of the structure of finite Boolean dynamical systems and their asymptotic combinatorics. The question of counting the maximal number of fixed points was addressed in [12] through the following question: "Given an influence graph $G = (V, E)$, with $|V| = n$, what is $\text{maxFP}(G) = \max(\{\text{card}(\text{FP}(G, f)) \mid f : \mathbb{B}^n \to \mathbb{B}^n\})$?", where $\text{maxFP}(G)$ is the maximum number of fixed points that a Boolean network f admitting G as influence graph can have. For the sake of clarity here, notice that the input of this problem is not a Boolean network but an influence graph which, by definition, can admit several distinct Boolean networks. Furthermore, the authors chose the parallel updating mode as the reference updating mode in their study, resting on the fixed points stability theorem (see Theorem 1), in order to consider only the very fixed points of f and those which could be generated by specific modes, as if the latter were some kinds of side effects. They obtain, among other results, the following theorem.

Theorem 4 ([12])**.** *Let $G = (V, E)$ be an influence graph, $\nu(G)$ be the maximum number of disjoint cycles of G (also known as its packing number), and $\tau^+(G)$ be the minimum number of vertices meeting every positive cycle of G (also known as its positive feedback vertex set). Then:*

$$\nu(G) + 1 \ \leqslant\ \text{maxFP}(G) \ \leqslant\ 2^{\tau^+(G)}.$$

This result paves the way to refinements, in particular of the upper bound. Indeed, here, we can see that only positive cycles are considered in this latter bound. To refine it, we could for instance take into account the role of negative cycles, in the sense that [46] showed that their intersections with positive cycles tends to maintain the convergence towards fixed points, while reducing it. However, whilst this seems to be a natural and promising research track, the fact remains that obtaining such refinements will be highly difficult since the impact of negative cycles is still very weakly understood because of their versatility.

Eventually, remark that similar questions could be tackled about limit cycles. But these limit sets are peculiar in the sense that, contrary to fixed points which are rather stable depending on updating modes, limit cycles are very sensitive to updating modes. So, for the sake of generality, such studies would necessarily ask to be parameterised with updating modes, which obviously adds to the difficulty of dealing with limit cycles counting. An approach which could lead to make progress on this would consist for instance in changing the viewing angle, by focusing more on analysing the complexity of counting through pertinent decision problems.

4.2 Computational complexity

Given a Boolean network f of dimension n and an updating mode μ, we focus on the computational complexity of determining basic dynamical properties related to fixed points, reachability between configurations, and limit sets. All these properties reduce to simple graph properties of transition graph $\mathscr{D}_{(f,\mu)}$ as discussed in Section 2.3.3. However, this graph is of exponential size with n, both in terms of number of nodes (configurations) and arcs (transitions). Moreover, recall that updating modes like the asynchronous one can generate an exponential number of out-going transitions from a single configuration, leading thus to a doubly-exponential number of arcs in the transition graph. However, this is only an upper bound on the actual complexity of computing the desired dynamical properties.

In what follows, we give an overview of the complexity of deciding dynamical properties of Boolean networks. As usual, these properties are expressed as decision problems, i.e. expecting a yes or no answer. The complexity will be expressed in function of the number of dimensions n, the size of Boolean network f and additional inputs of the problem, typically configurations, when relevant. Boolean network f is assumed to be given in a symbolic form, for instance expressed using propositional logic as done along this chapter (in contrast with a truth table). Moreover, we assume that given a configuration x, evaluating $f(x)$ is linear in the size of f.

Let us first recall the bases of computational complexity classes [106, 13]: the P class (resp. NP class, PSPACE class) is formed by the decision problems which can be solved by algorithms running in polynomial time (resp. in polynomial time with non- deterministic choices, in polynomial space) according to the size of its inputs. We know that P $\subseteq$ NP $\subseteq$ PSPACE, where "$\subseteq$" can be understood as "simpler". A problem is harder than another problem if the latter can be "reduced" to the former (intuitively, it is a particular case of the former problem). In general, different types of reductions can be considered. In this chapter, we essentially rely on polynomial-time reductions. A problem is hard for a given complexity class if it is among the hardest problems of this class. A problem is complete for a given complexity class if it belongs to and is among the hardest problems of this class. The famous SAT problem of determining if a formula expressed in propositional logic (essentially Boolean variables and logic connectors) has a satisfying solution is NP-complete. It is not known yet if NP = PSPACE, but in practice, NP-complete problems are much more tractable than PSPACE-complete ones, by several orders of magnitude. Hereafter, we also refer to the coNP class, delimiting the problems for which finding a counter-example is in NP, and to the P^{NP} and $\mathrm{coNP}^{\mathrm{coNP}}$ classes, where A^{B} denotes the problems which can be solved with complexity A assuming problems of class B can be solved in one instruction (oracle); remark that P^{P}=P and NP^{P}=NP. These complexity classes belong to the polynomial hierarchy, and are subject to the following properties: NP $\subseteq \mathrm{P}^{\mathrm{NP}}$ and coNP $\subseteq \mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{coNP}^{\mathrm{coNP}} \subseteq$ PSPACE.

Together with the details of the complexity results, we also give pointers to software tools which implement the corresponding dynamical analyses. However, before entering into the details, let us present one very recent result which is certainly the most general in the framework of automata networks, and thus of Boolean networks, because it emphasises that all relevant problems we may want to address in the context of Boolean networks are actually complex. In the 1950's, [114] showed that *any non-trivial property of the function computed by a Turing machine is undecidable*, namely no algorithm leading to a correct positive or negative answer can be constructed. This result is a major result in computer science and more precisely in computability theory since it generalises for instance the undecidability of the halting problem [143]. In [58], among others, the authors suggested a meta-theorem for automata networks stating that *any non-trivial property of the function computed by an automata network admits a high computational complexity*, namely is (co)NP-hard at least. This means notably that we are currently not able to conceive an efficient algorithm to decide if such properties are true or false.

4.2.1 Existence of a fixed point

The fixed points are a particular type of limit sets consisting of a single configuration. They form one of the most basic and important characteristics of Boolean networks. Most modelling studies rely on the computation of the fixed points to offer a first validation of the model with respect to expected stable behaviours of the system.

Proposition 4. *Given a Boolean network f of dimension n, deciding if there exists $x \in \mathbb{B}^n$ such that $f(x) = x$ is NP-complete.*

The belonging of this problem to NP comes directly from the complexity of evaluating $f(x)$, assumed to be in polynomial time. The completeness can be proven by reductions from different NP-complete problems, such as the SAT problem or the PARTITION problem [5, 56, 57].

Notice that the problem is formulated independently of the updating mode: the fixed points considered here are fixed points of the n-dimensional Boolean function f. By the folklore theorem (Theorem 1), the fixed points of f are fixed points of the dynamical system (f, μ). For many of the defined updating modes, including parallel, asynchronous, fully asynchronous, block-sequential, block-parallel, and most permissive, the fixed points of f correspond exactly to the fixed point of the dynamical system. As pointed out in Section 4.1.1, there exist updating modes which can generate fixed points which are not fixed points for f. In that case, the complexity of deciding the existence of the fixed point may be different, and will likely depend on the definition of the updating mode. However, in the case whenever computing a transition is in polynomial time, we can already settle that it is at most in NP. Then, determining the existence of a fixed point being either of f or of the dynamical system is NP-complete.

Following the link between the influence graph of a Boolean network and its fixed points discussed in Section 4.1.3, the complexity of deciding the existence of a Boolean network having a fixed given influence graph and a given minimal number of fixed point has been characterised by [21]. Notably, deciding whether there exists a Boolean network having a given influence graph and having at least one fixed point is in P; if having $k \geqslant 2$ fixed points is NP-complete. The former case comes from the result that one can build a Boolean network having at least one fixed point if and only if each non-singleton strongly connected component of the input influence graph contains at least one positive cycle, which can be decided in polynomial time. The latter result is done by reduction from the 3-SAT problem.

Software tools The resolution of the decision problem of the existence of a fixed point can be directly encoded as a SAT problem. When a fixed point exists, the resolution of the SAT problem comes with an instance of configuration x so that $f(x) = x$. Then, one can re-iterate with the decision of the existence of a fixed point different than x, and so on, in order to obtain an explicit listing of the fixed points of the Boolean network, as implemented in the software Pint [107]. In practice, the identification of a single fixed point by SAT solving is scalable to networks with several hundreds of thousands of automata. The number of iterations of SAT solving being linear with the number of fixed points, the complete and explicit listing of fixed points can only be obtained when there is a limited number of them; otherwise we usually perform a partial enumeration. Another approach for listing fixed points relies on data structures such as decision diagrams as done in the software GINsim [101, 99]. There, the set of fixed points is encoded symbolically, and their enumeration is done by enumerating particular paths within the diagram structure.

4.2.2 Reachability between configurations

The reachability problem consists in deciding whether there exists a trajectory leading from a given configuration x to a given configuration y with a given updating mode μ. Such properties are employed to predict future states which may be observed in the future in the modelled system.

Let us first focus on the classical parallel, asynchronous, and fully asynchronous updating modes. As there is at most 2^n configurations to explore, the problem can be solved in a polynomial space: consider a non-deterministic simulation algorithm which iteratively applies a transition modifying the current configuration of the network. If there exists a trajectory from x to y, then there is at least one execution of the algorithm which will encounter the configuration y after at most $2^n - 1$ transitions. Thus, it is sufficient to store the current configuration of the network and a counter which is incremented after each transition. Considering a binary encoding, this counter requires n bits. Thus, the non- deterministic simulation algorithm requires a linear space, and is thus in PSPACE. With the parallel updating mode, the PSPACE-hardness derives by reduction from the reachability problem in reaction systems, a subclass of synchronous Boolean networks [48]. With fully asynchronous and asynchronous updating modes, the PSPACE-hardness derives by reduction from the reachability problem in synchronous Boolean networks. Indeed, similarly to cellular automata [97], one can define a Boolean network g so that asynchronous and fully asynchronous dynamics give reachability relations which are equivalent with the synchronous dynamics of f [109].

Proposition 5. *Given a Boolean network f of dimension n, an updating mode μ, and two configurations $x, y \in \mathbb{B}^n$, deciding if $x \to^*_{(f,\mu)} y$ is PSPACE-complete with $\mu \in \{\mathsf{p}, \mathsf{fa}, \mathsf{a}\}$.*

Regarding the sequential updating modes, it is interesting to remark that one can encode the parallel dynamics of a Boolean network f of dimension n as a sequential dynamics of a Boolean network g in at most polynomial time. One naive approach is to double the number of automata, the first set storing the next state of each original automaton, and the second set storing its state before the update. The first set is updated first, and then the second set copies the states of the first set of automata: define

g with $2n$ dimensions such that for all $x, x' \in \mathbb{B}^n$, for each $i \in [\![n]\!]$, $g_i(xx') = f_i(x')$ and $g_{n+i}(xx') = x_i$. Then, for all $x, y \in \mathbb{B}^n$, $x \rightarrow_{(f,\mathsf{p})} y$ if and only if $xx \rightarrow_{(f,(1,\cdots,2n))} yy$. Note that [22, 19, 20] proposed a more efficient encoding requiring less than $2n$ automata. Thus, the reachability problem with the sequential updating mode is harder than with the parallel updating mode. Because it lies in PSPACE, it is therefore PSPACE-complete as well. Finally, because the parallel updating mode is a particular case of the block-sequential and block-parallel, the complexity result applies for these updating modes as well, and any other generalisation of them.

As demonstrated in [109], the case of the permissive updating mode is quite different. Following Definition 8, the computation of the configurations succeeding configuration x is done by a two-steps process repeated for each subset of automata W. First it will compute the smallest sub-hypercube h which contains x and such that for each dimension $i \in W$, either $h_i = *$ or for each vertex z of the sub-hypercube, $f_i(z) = h_i = x_i$. This computation can be performed by the following algorithm, where $c(h)$ denotes the set of vertices of the sub-hypercube h:

```
h := $x$
repeat $|W|$ times
    for each $i \in W$ such that h$_i \neq *$:
        if $\exists z \in c($h$)$ such that $f_i(z) \neq x_i$:
            h$_i$ := $*$
```

The second step consists in filtering the obtained sub-hypercube by removing states of automata in W which cannot be computed from any vertex of h. From the computation above, these states correspond to the states of automata of W in x:

```
for each $i \in W$ such that h$_i = *$:
    if not ($\exists z \in c($h$)$ such that $f_i(z) = x_i$):
        h$_i$ := 1 - $x_i$
```

Then, y is reachable from x if and only if y is a vertex of such an obtained sub-hypercube, for at least one subset W.

Let us analyse the complexity of this algorithm. First, with a fixed W, remark that the computation of h relies on several tests of the form "$\exists z \in c(h) : f_i(z) = b$", with $b \in \mathbb{B}$. This is exactly the SAT problem. Thus, in the general case, such decisions are NP-complete, and in the case whenever f is locally monotone (thus f_i is monotone), such decisions are in P. Then, given the configuration y, it is key to remark that there is no need to compute the sub-hypercubes for all the possible subsets W. Indeed, let us focus on the case whenever for a fixed W, y is not a vertex of the computed sub-hypercube. Two cases arise: *(i)* y is not a vertex of the sub-hypercube before the filtering: then, it is also the case for all the subsets of W; *(ii)* y is no longer a vertex of the sub-hypercube after the filtering: it means that there is a subset $D \subseteq W$ of automata which have been filtered and such that $\forall i \in D$, $x_i = y_i$. Then, y is not a vertex of all the sub-hypercubes computed with W minus a strict subset of D. Thus, the overall procedure starts with $W = [\![n]\!]$, and repeatedly removes the subset D until either y is a vertex of the sub-hypercube (and it is thus reachable from x), or is not a vertex of the sub-hypercube before filtering (and is thus not reachable from x).

Proposition 6. *Given a Boolean network f of dimension n, and two configurations $x, y \in \mathbb{B}^n$, deciding if $x \rightarrow^*_{(f,\mathsf{MP})} y$ is in P if f is locally monotone, otherwise it is in P^{NP}.*

Software tools For the synchronous, fully asynchronous and asynchronous updating modes, the verification of reachability properties is usually tackled by generic tools related to the model checking of discrete and finite dynamical systems: the reachability property being expressed in temporal logics such as CTL [29]. Tools like GINsim [99] enable exporting Boolean networks to files in suitable formats for model checkers like NuSMV [27]. The verification of reachability properties can take advantage of static analysis to reduce the transitions to explore, as offered by the tool Pint [107], enhancing the scalability of the computation in many cases [108]. The verification of reachability properties using the most permissive updating mode is implemented in the tool MPBN [109]. In practice, due to the differences in complexity, the verification of reachability properties using asynchronous updating modes scales up to networks with around one hundred automata; using the most permissive updating mode, it scales up in the order of hundreds of thousands of automata [111].

4.2.3 Limit configurations

The limit configurations and reachable limit configurations are among the most analysed dynamical properties of Boolean networks when modelling biological systems. Limit configurations generalise fixed points by taking into account non-singleton limit sets. Combined with reachability analysis, they delineate the possible long-term behaviours of the network from a given initial configuration.

In this section, we focus on the following decision problem: *Given a configuration, does it belong to a limit set of the Boolean network with given updating mode?* The advantage of such a formulation is that the size of the input is independent of that of the limit set. Furthermore, it makes a direct link with the problem of identification and enumeration of the limit sets. Moreover, we do not explicitly address the reachability of the limit configurations from a given initial configuration. However, this comes naturally by combining with the decision problem discussed in the previous section.

How can we determine that a configuration x is a limit configuration? This problem is actually tied to the reachability problem: x is a limit configuration if and only if it is reachable from any configuration y reachable from x (with the updating mode μ). Equivalently, x is not a limit configuration if and only if there exists a configuration y reachable from x but x is not reachable from y. Using the same reasoning as for reachability, one can deduce that this problem is in coNPSPACE = PSPACE for the parallel, fully asynchronous, and asynchronous updating modes. As for the reachability, the completeness can be derived by reduction from the same problem in synchronous reaction systems demonstrated to be PSPACE-complete in [48]. Then, using the reduction of synchronous Boolean networks to fully asynchronous and asynchronous updating modes, we obtain that the decision problem of limit configurations is PSPACE-complete.

Proposition 7. *Given a Boolean network f of dimension n and a configuration $x \in \mathbb{B}^n$, deciding if x belongs to a limit set of f with updating mode $\mu \in \{\mathsf{p}, \mathsf{fa}, \mathsf{a}\}$ is PSPACE-complete.*

With the exact same arguments as for the reachability problem, this complexity also applies to the sequential updating modes and any generalisation of them.

The case of the most permissive updating mode is again much more specific [109]. Indeed, it appears that limit sets of the most permissive dynamics have a particular shape: they always correspond to particular sub-hypercubes, namely to the smallest sub-hypercubes which are closed by f, also known as *minimal trap spaces* [85]. A sub-hypercube h is closed by f if and only if for each of its vertices x, $f(x)$ is also a vertex of the hypercube. Fixed points are a particular case of minimal trap spaces where the sub-hypercubes have dimension 0 (all the automata have a fixed state). This property comes from the fact that whenever two configurations lying a diagonal are reachable from each others with the most permissive updating mode, then so are the adjacent vertices. In other words, and without loss of generality, let us consider that configurations $x = a00b$ and $y = a11b$, where a and b are binary vectors, are reachable from each others. This implies that there exists a set of automata W such that x and y belong to the sub-hypercube h computed according to the previous section from x or y, or equivalently, belong to $\Lambda_W \circ \Phi^{\omega}_{W,\nabla}(\{x\}) = \Lambda_W \circ \Phi^{\omega}_{W,\nabla}(\{y\})$, following the notations of Definition 8. Thus, because of the sub-hypercube structure, both $a01b$ and $a10b$ belong to h, and are thus reachable from x and y, and conversely. This is illustrated by Figure 11.

Proposition 8. *$A \subseteq \mathbb{B}^n$ is a limit set of f with the most permissive updating mode if and only if A forms a minimal sub-hypercube closed by f.*

Thus, determining if a configuration x belongs to a limit set with the most permissive updating mode boils down to determining the existence of a minimal sub-hypercube h which contains x and

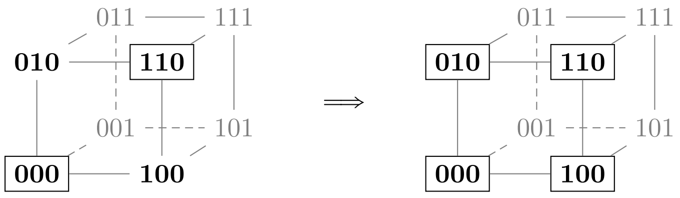


Figure 11: Illustration of the property of limit sets with the most permissive updating mode, where boxed configurations belong to a same limit set: whenever two configurations belong to the same limit set, so do all the configurations of the smallest sub-hybercube which contains both of them.

is closed by f. Consider IS-NOT-CLOSED(f, h) the problem of deciding if the sub-hypercube h *is not closed* by f: it is equivalent to deciding if there exists an automaton $i \in [\![n]\!]$ with $h_i \neq *$ and a vertex z of h such that $f_i(z) \neq h_i$, which is NP-complete in general, and P whenever f is locally monotone. Then, the complementary problem IS-CLOSED(f, h) is in coNP in the general case and in P in the locally monotone case. Now, consider IS-NOT-MINIMAL(f, h) the problem of deciding if the sub-hypercube h closed by f *is not minimal*: it can be solved by deciding wherever there exists a sub-hypercube h' which is strictly included in h and which is closed by f, which is at most $\mathrm{NP}^{\text{IS-CLOSED}}$. Thus, the complementary problem IS-MINIMAL(f, h) is in $\mathrm{coNP}^{\text{IS-CLOSED}}$, i.e. $\mathrm{coNP}^{\mathrm{coNP}} = \Pi_2^{\mathrm{P}}$ in the general case and coNP in the locally monotone case.

Theorem 5. *Given a Boolean network f of dimension n and a configuration $x \in \mathbb{B}^n$, deciding if x is a limit configuration of f with the most permissive updating mode is in coNP whenever f is locally monotone, and in Π_2^P otherwise.*

Software tools Similarly to the analysis of fixed points, the software tools usually focus on the enumeration of the limit sets of a given Boolean network and a given updating mode. The case of the parallel and fully asynchronous updating modes is implemented in several software tools such as BoolSim [59] (parallel and fully asynchronous), BNS [50] (parallel), and Cabean [133] (fully asynchronous). They rely either on SAT solving (BNS) or on symbolic representations of the reachable configurations with decision diagrams. To tackle the potential combinatorial explosion of the number of configurations within a limit set, the methods output them as unions of sub-hypercubes. This representation comes quite directly when using decision diagrams, for instance. In practice, the scalability is roughly similar to the reachability analysis, i.e. in the order of the hundred of automata. In the case of the most permissive updating mode, because the configurations of the limit sets correspond to specific sub-hypercubes, their identification can be reduced to the subset-minimal solutions of an Answer-set programming problem [14] as done by the tool mpbn [109]. In practice, this approach can be applied to networks with several hundreds of thousands of components, at least for a partial enumeration of their limit sets when there are too many of them for an explicit enumeration [111].

5 Conclusion

Updating modes and time In this chapter, we have developed some elements related to Boolean networks and their use as models of biological complex systems, in particular as models of genetic regulation networks. More precisely, our purpose has been focused on the updating modes, by underlining that Boolean network dynamics strongly depend on the manner automata execute their local functions. If we consider these updating modes as means of representation of the relations that automata maintain with *time*, countless questions about the nature and consideration of time raise. These questions are obviously relevant in mathematics and computer science (notice that one main problematic in this discipline concerns synchronism and asynchronism), but they are all the more pertinent in the context of theoretical biology, insofar as the answers that we could bring in this context would possibly increase knowledge of the way time flows at different scales of living organisms.

When a real interacting system $\mathscr{S}$ is modelled by a Boolean network, or by an automata network f by extension, we generally consider that every state of $\mathscr{S}$ can be associated to one (or more) configuration(s) of f, up to a specific encoding. Moreover, we classically think of these systems as collections of entities which interact with each other *over time*. As a consequence, the very concept of a transition from a configuration x to a configuration y in the dynamics of f implicitly integrates a notion of time which places x before y, which leads us to make an association between a trajectory and the intuitive concept of temporal flow. But what can be the semantics of a transition actually? In this paragraph, we propose three distinct visions, and are aware of the non-exhaustiveness of the latter.

Modelling durations When we study automata networks and the dynamical systems they can model, we associate them with transition graphs which are either deterministic, or non-deterministic. In this latter case, they can be either non-stochastic or stochastic, i.e. complemented with a measure of probability in order to weight each of their transitions. In the general case, such systems induce a time domain which is either continuous or discrete (in this chapter, the time domain is $\mathbb{N}$). In this framework, a natural abstraction consists in seeing the trajectories as the temporal flow. However, the coherence with physical time would imply that the discrete transitions embed the concept of duration and represent all the same duration. This corresponds precisely to a real time discretisation. However, whenever an automata network is considered as a model of a real system, this vision can be perceived as unrealistic. This is why some studies increase the concept of transition by adding that of duration, so that the transitions can be associated with distinct durations. They are then labelled by the amount of time they are supposed to last or, more precisely, by the duration of the event(s) that they are supposed to represent [138, 18, 129]. In addition to the questions presented in the following paragraphs, this choice of modelling the duration of a temporal flow raises specific theoretical questions, including: how long does a network take to reach an asymptotic behaviour in the best case, in the worst case or on average?; what is the probability that the network passes asymptotically by this or that configuration?...

However, integrating this concept of chronometric time (through durations) within a fundamentally discrete formalism raises a problem extremely difficult to solve. In particular, in [104], the authors emphasised how this problem leads naturally, almost necessarily, to prefer a continuous framework. In order to avoid this problem, and thus not to give the discrete framework more capacities that it has while retaining its intrinsic advantages, the following paragraphs shows that we can choose to model time differently.

Modelling precedence A second way of conceiving time with discrete dynamical systems consists in no longer considering the trajectories as witnesses of the durations of a temporal flow but as representatives of a relation of precedence. They are then nothing else but sequences of events without duration. In this case, if transitions $x \to^* y$ and $x' \to^* y'$ are both achievable at the same time, then, this vision makes it possible to give meaning that $x \to^* y$ may last longer than $x' \to^* y'$, under certain conditions, and conversely that $x' \to^* y'$ may last longer than $x \to^* y$, under other conditions.

Thus, the different behaviours of an automata network can occur at different time scales without new information being specified to distinguish them. The modelled time then deviates a little from the notion of real time to become a *logical time*, which requires less of knowledge on the transitions between states of the modelled real system. The transitions and trajectories are then only sequences of events which follow each others without integrating any notion of duration. More precisely, a transition $x \to y$ does not hold information about the real process it models but only expresses the result of a possible observation of the system at a certain time step returning that the system is in state y, knowing that the previous observation of the system returned the state x. As a result, the time that transitions and trajectories take cannot be measured, only the number of events, and their succession can be. This is the classical approach of time which is considered when we work on deterministic or non-deterministic discrete dynamical systems. Typical questions highlighting this conception of time are the following [91]: how many steps does a network take to reach limit sets?, is such behaviour always observed after such event?...

Modelling causality Another view we can have of time relates to the causality between events, where the notion of duration of transitions is replaced by a relation of causes and consequences. Causality essentially refers to the fact that an event can only be triggered once a particular condition is met, and triggering an event may preempt other *concurrent* events. This modelling aims at revealing a fine grained dependency structure between transitions and events. Then, precedence becomes an emerging property, as causal relations rule which events *must* appear before others, which events can never appear after others, and which sets of events can be interleaved freely. Causal modelling focuses on the *minimal* conditions for triggering a certain event or observe a given behaviour, which relates to determining the prime implications for transitions. Questions related to this context are typically about the existence or accessibility properties: is this transition possible?, is a configuration that checks such properties reachable?... as well as for the control: which modification of the model is sufficient to make a transition possible or impossible? Notice that, in this context, the transition graph hinders important parts of this information, as it forgets the local functions which generated it. The concurrency theory offers conceptual tools, including event structures for reasoning efficiently on the causality between events, that can be naturally applied to Boolean networks with their deterministic and non-deterministic updating modes [23].

Towards an updating mode hierarchy Echoing with different modelling hypotheses and specifications of time, we presented along this chapter a large range of updating modes for Boolean networks. In this chapter, we endeavoured to define them with a unifying mathematical framework of deterministic and non-deterministic updates. The relationship between the different updating modes is a fundamental question which aims at bringing a structure between these numerous dynamics. Let us say that an updating mode μ is *(weakly) simulated* by an updating mode μ', noted $\mu \preceq \mu'$, if and only if, for any Boolean network f and any pair (x, y) of its configurations, if there exists a path from x to y in the dynamics generated by μ, then there exists a path from x to y in the dynamics generated by μ'. Formally:

$$\mu \preceq \mu' \iff \forall f, \forall x, \forall y,\ x \rightarrow^*_{(f,\mu)} y \implies x \rightarrow^*_{(f,\mu')} y.$$

A class of updating modes C is then weakly simulated by a class of updating modes C', if for any updating mode of the former class, there exists an updating mode of the latter class which weakly simulates it:

$$C \preceq C' \iff \forall \mu \in C, \exists \mu' \in C',\ \mu \preceq \mu'.$$

The obtained hierarchy is depicted in Figure 12.

Software tools In the past twenty years numerous software tools for modelling and analysing dynamics of Boolean networks have been developed. The CoLoMoTo interactive notebook [100] provides a distribution of many of these tools promoting their accessibility and the writing and publishing of reproducible computational analysis of Boolean networks. Based on this software environment, we designed interactive notebooks for reproducing the examples given in this chapter with the different updating modes. They are available at https://github.com/pauleve/updating-modes-notebooks (archived at https://doi.org/10.5281/zenodo.5260025), and can be easily executed and re-used for analysing different Boolean networks and implement custom updating modes.

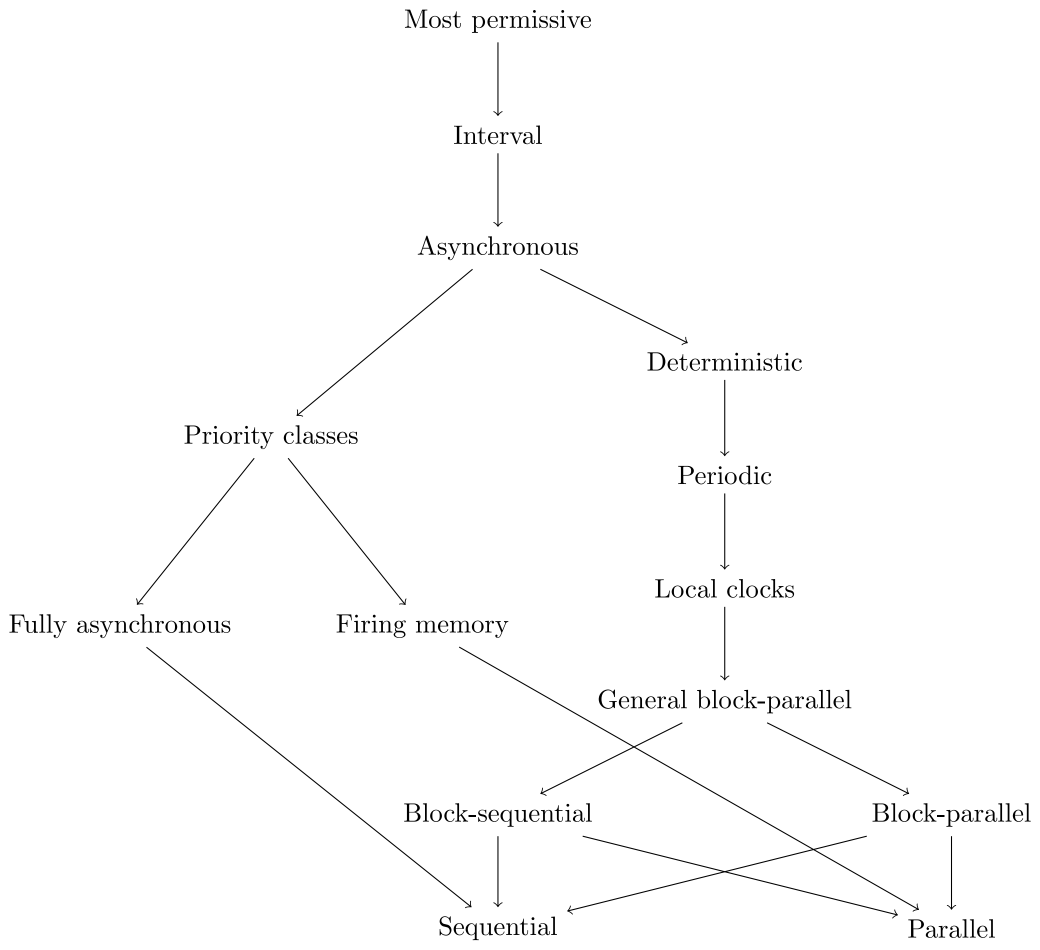


Figure 12: Weak simulation relation between the classes of updating modes mentioned in this chapter with any fixed Boolean network.

Opening on intrinsic simulations The hierarchy presented in Figure 12 shows the relations between the paths generated by the different updating modes on any fixed Boolean network. Another direction is to consider simulation relations between the dynamical systems, i.e. Boolean networks coupled with an updating mode.

A perspective of the work presented in this chapter may focus on simulations of Boolean networks evolving with non-deterministic updates by Boolean networks evolving with deterministic updates. A first natural way is by following a classical determinisation of the dynamics. Indeed, one can encode any set of configurations in $\mathbb{B}^n$ as one configuration in $\mathbb{B}^{2^n}$. Let us consider such an encoding $c : 2^{\mathbb{B}^n} \to \mathbb{B}^{2^n}$ where, for all $x \in \mathbb{B}^n$, $c(X)_x = 1$ if $x \in X$, otherwise $c(X)_x = 0$ (we slightly abuse notations here, by specifying a vector index by its binary representation). Now, it is clear that for any set update $\Phi : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ of a Boolean network f of dimension n, one can define a Boolean network g such that for all sets of configurations $X \subseteq \mathbb{B}^n$, $g(c(X)) = c(\Phi(X))$. This encoding is complete in the sense that any transition generated by Φ is simulated in (g, p). But these simulations are nothing else but a brute-force encoding in which we get rid of the transition relation by increasing exponentially the state space. Moreover, with this deterministic encoding, the structure of the transition relation of $(f, \mu = \Phi)$ is lost, which makes much more difficult characterising dynamical features of (f, μ) such as its limit sets for instance.

Actually, a fundamental matter here lies in the concept of simulation at stake: we are interested in intrinsic simulations which go far beyond the classical concepts of encoding or simulation. Indeed, intrinsic simulations aim at conserving dynamical structures in addition to operated computations. So, one of the first questions to answer would consist in defining formally different kinds of intrinsic simulations. Nevertheless, firstly, consider the following intrinsic simulation: a dynamical system (f, μ) simulates another (g, μ') if $\mathscr{D}_{(g,\mu')}$ is a subgraph of $\mathscr{D}_{(f,\mu)}$. With this rather simple definition, it is direct to state that, with a and $\overline{\mathsf{d}}$ the asynchronous and memory updating modes respectively, for any Boolean network f, (f, a) simulates $(f, \overline{\mathsf{d}})$. Some natural questions related to Boolean networks updated with memory are the following:

- Are there Boolean networks whose dynamics obtained according to $\overline{\mathsf{d}}$ remains deterministic, whatever $\overline{\mathsf{d}}$?
- If so, what are their properties and what are the equivalent deterministic updating modes?

To go further, consider the most permissive updating mode. It is direct that (f, μ) does not simulate (f, MP), except for very particular f. Let us now consider a more general intrinsic simulation: a dynamical system (f, μ) simulates another (g, μ') if $\mathscr{D}_{(g,\mu')}$ is a graph obtained from $\mathscr{D}_{(f,\mu)}$ thanks to edge deletions, and vertex shortcuts. A lot of promising questions arise from this, in particular related to $\overline{\mathsf{d}}$ and MP updating modes, among which for instance:

- Let per be a deterministic periodic updating mode. How can $(f, \overline{\mathsf{d}})$ be simulated by (g, per)? The answer is known for $\mathsf{per} = \mathsf{p}$ [63], but it seems pertinent to find a generalisation to deterministic periodic updating modes, and even more general deterministic updating modes.
- Intuitively, any (f, MP) might be simulated by (g, a), where f and g are Boolean networks and the dimension of g is greater than that of f. But how many automata need to be added to g depending on the dimension of f?

All answers, even partial or negative, will bring a better understanding of updating modes and Boolean networks, which would lead to pertinent further development in both Boolean network theory and their application in systems biology.

Acknowledgements. This work was primarily funded by our salaries as French State agents (S. Sené being officially affiliated to Université d'Aix-Marseille, Université Toulon, CNRS, LIS, UMR 7020, Marseille, France), and secondarily by Agence Nationale pour la Recherche (ANR) in the scope of the projects ANR-18-CE40-0002 FANs (SS) and ANR-20-CE45-0001 BNeDiction (LP).

References

[1] Wassim Abou-Jaoudé, Pedro T. Monteiro, Aurélien Naldi, Maximilien Grandclaudon, Vassili Soumelis, Claudine Chaouiya, and Denis Thieffry. Model checking to assess T-helper cell plasticity. *Frontiers in Bioengineering and Biotechnology*, 2:86, 2015.

[2] Wassim Abou-Jaoudé, Denis Thieffry, and Jérôme Feret. Formal derivation of qualitative dynamical models from biochemical networks. *Biosystems*, 149:70–112, 2016.

[3] Jamil Ahmad, Olivier Roux, Gilles Bernot, Jean-Paul Comet, and Adrien Richard. Analysing formal models of genetic regulatory networks with delays. *International Journal of Bioinformatics Research and Applications*, 4:240–262, 2008.

[4] Aurore Alcolei, Kévin Perrot, and Sylvain Sené. On the flora of asynchronous locally non-monotonic Boolean automata networks. In *Proceedings of the International Workshop on Static Analysis and Sytems Biology*, volume 326 of *Electronic Notes in Theoretical Computer Science (ENTCS)*, pages 3–25. Elsevier, 2016.

[5] Noga Alon. Asynchronous threshold networks. *Graphs and Combinatorics*, 1:305–310, 1985.

[6] Julio Aracena. Maximum number of fixed points in regulatory Boolean networks. *Bulletin of Mathematical Biology*, 70:1398–1409, 2008.

[7] Julio Aracena, Jacques Demongeot, and Eric Goles. Positive and negative circuits in discrete neural networks. *IEEE Transactions on Neural Networks*, 15:77–83, 2004.

[8] Julio Aracena, Eric Fanchon, Marco Montalva, and Mathilde Noual. Combinatorics on update digraphs in Boolean networks. *Discrete Applied Mathematics*, 159:401–409, 2011.

[9] Julio Aracena, Eric Goles, Andres Moreira, and Lilian Salinas. On the robustness of update schedules in Boolean networks. *Biosystems*, 97:1–8, 2009.

[10] Julio Aracena, Luis Gómez, and Lilian Salinas. Limit cycles and update digraphs in Boolean networks. *Discrete Applied Mathematics*, 161:1–12, 2013.

[11] Julio Aracena, Mauricio González, Alejandro Zuñiga, Marco A. Mendez, and Verónica Cambiazo. Regulatory network for cell shape changes during Drosophila ventral furrow formation. *Journal of Theoretical Biology*, 239:49–62, 2006.

[12] Julio Aracena, Adrien Richard, and Lilian Salinas. Number of fixed points and disjoint cycles in monotone Boolean networks. *SIAM Journal on Discrete Mathematics*, 31:1702–1725, 2017.

[13] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[14] Chitta Baral. *Knowledge representation, reasoning and declarative problem solving.* Cambridge University Press, 2003.

[15] Theodore Beauchaine. Vagal tone, development, and Gray's motivational theory: toward an integrated model of autonomic nervous system functioning in psychopathology. *Development and Psychopathology*, 13:183–214, 2001.

[16] Claire Bendix, Carine M. Marshall, and Frank G. Harmon. Circadian clock genes universally control key agricultural traits. *Molecular Plants*, 8:1135–1152, 2015.

[17] Arndt Benecke. Chromatin code, local non-equilibrium dynamics, and the emergence of transcription regulatory programs. *The European Physical Journal E*, 19:353–366, 2006.

[18] Gilles Bernot, Jean-Paul Comet, Adrien Richard, and Janine Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229:339–347, 2004.

[19] Florian Bridoux. Sequentialization and procedural complexity in automata networks. arXiv:1803.00438, 2018.

[20] Florian Bridoux. *Simulations intrinsèques et complexités dans les réseaux d'automates.* PhD thesis, Université d'Aix-Marseille, 2019.

[21] Florian Bridoux, Nicolas Durbec, Kévin Perrot, and Adrien Richard. Complexity of maximum fixed point problem in Boolean networks. In *Proceedings of the Conference on Computability in Europe*, volume 11558 of *Lecture Notes in Computer Science (LNCS)*, pages 132–143. Springer, 2019.

[22] Florian Bridoux, Pierre Guillon, Kévin Perrot, Sylvain Sené, and Guillaume Theyssier. On the cost of simulating a parallel Boolean automata network with a block-sequential one. In *Proceedings of the International Conference on Theory and Applications of Models of Computation*, volume 10185 of *Lecture Notes in Computer Science (LNCS)*, pages 112–128. Springer, 2017.

[23] Thomas Chatain, Stefan Haar, Juraj Kolčák, Loïc Paulevé, and Aalok Thakkar. Concurrency in Boolean networks. *Natural Computing*, 19:91–109, 2020.

[24] Thomas Chatain, Stefan Haar, and Loïc Paulevé. Boolean networks: beyond generalized asynchronicity. In *Proceedings of International Workshop on Cellular Automata and Discrete Complex Systems*, volume 10975 of *Lecture Note in Computer Science (LNCS)*, pages 29–42. Springer, 2018.

[25] Alonso Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33:346–366, 1932.

[26] Alonso Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[27] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: an ppenSource tool for symbolic model checking. In *Proceedings of the International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science (LNCS)*, pages 359–364. Springer, 2002.

[28] Olivier Cinquin and Jacques Demongeot. Positive and negative feedback: strinking a balance between necessary antagonists. *Journal of Theoretical Biology*, 216:229–241, 2002.

[29] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of the Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science (LNCS)*, pages 52–71. Springer, 1981.

[30] David P. A. Cohen, Loredana Martignetti, Sylvie Robine, Emmanuel Barillot, Andrei Zinovyev, and Laurence Calzone. Mathematical modelling of molecular pathways enabling tumour cell invasion and migration. *PLoS Computational Biology*, 11:e1004571, 2015.

[31] Samuel Collombet, Chris van Oevelen, Jose Luis Sardina Ortega, Wassim Abou-Jaoudé, Bruno Di Stefano, Morgane Thomas-Chollier, Thomas Graf, and Denis Thieffry. Logical modeling of lymphoid and myeloid cell specification and transdifferentiation. *Proceedings of the National Academy of Sciences*, 114:5792–5799, 2017.

[32] Louis Comtet. Recouvrements, bases de filtre et topologies d'un ensemble fini. *Comptes rendus de l'Académie des Sciences – Série A*, 262:1091–1094, 1966.

[33] Michel Cosnard and Jacques Demongeot. On the definitions of attractors. In *Proceedings of the International Symposium on Iteration Theory and its Functional Equations*, volume 1163 of *Lecture Notes in Mathematics*, pages 23–31. Springer, 1985.

[34] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the ACM SIGACT-SIGPLAN Symposium on Principles of programming languages*, pages 238–252. Association for Computing Machinery, 1977.

[35] Michael F. Covington and Stacey L. Harmer. The circadian clock regulates auxin signaling and responses in *Arabidopsis*. *PLoS Biology*, 5:e222, 2007.

[36] Yves Crama and Peter L. Hammer. *Boolean functions: theory, algorithms, and applications*, volume 142 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2011.

[37] Haimabati Das and Ritwik Kumar Layek. Estimation of delays in generalized asynchronous boolean networks. *Molecular BioSystems*, 12:3098–3110, 2016.

[38] Maria I. Davidich and Stefan Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One*, 3:e1672, 2008.

[39] Hidde de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9:67–103, 2002.

[40] Max Delbrück. Génétique du bactériophage. In *Unités biologiques douées de continuité génétique*, volume 8 of *Colloques internationaux du CNRS*, pages 91–103, 1949.

[41] Jacques Demongeot, Adrien Elena, and Sylvain Sené. Robustness in regulatory networks: a multi-disciplinary approach. *Acta Biotheoretica*, 56:27–49, 2008.

[42] Jacques Demongeot, Eric Goles, Michel Morvan, Mathilde Noual, and Sylvain Sené. Attraction basins as gauges of the robustness against boundary conditions in biological complex systems. *PLoS One*, 5:e11793, 2010.

[43] Jacques Demongeot, Dan Istrate, Hajer Khlaifi, Lucile Mégret, Carla Taramasco, and René Thomas. From conservative to dissipative non-linear differential systems. An application to the cardio-respiratory regulation. *Discrete & Continuous Dynamical Systems – Series S*, 13:2121–2134, 2020.

[44] Jacques Demongeot, Christelle Jézéquel, and Sylvain Sené. Boundary conditions and phase transitions in neural networks. Theoretical results. *Neural Nerworks*, 21:971–979, 2008.

[45] Jacques Demongeot, Tarek Melliti, Mathilde Noual, Damien Regnault, and Sylvain Sené. *Automata and Complexity: Essays presented to Eric Goles on the occasion of his 70th birthday*, chapter On Boolean automata isolated cycles and tangential double-cycles dynamics. Emergence, Complexity, Computation. Springer, 2021. To appear.

[46] Jacques Demongeot, Mathilde Noual, and Sylvain Sené. Combinatorics of Boolean automata circuits dynamics. *Discrete Applied Mathematics*, 160:398–415, 2012.

[47] Jacques Demongeot and Sylvain Sené. About block-parallel Boolean networks: a position paper. *Natural Computing*, 19:5–13, 2020.

[48] Alberto Dennunzio, Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. Complexity of the dynamics of reaction systems. *Information and Computation*, 267:96–109, 2019.

[49] Olga Dergacheva, Kathleen J. Griffioen, Robert A. Neff, and David Mendelowitz. Respiratory modulation of premotor cardiac vagal neurons in the brainstem. *Respiratory Physiology & Neurobiology*, 174:102–110, 2010.

[50] Elena Dubrova and Maxim Teslenko. A SAT-based algorithm for finding attractors in synchronous Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8:1393–1399, 2011.

[51] Federica Eduati, Patricia Jaaks, Jessica Wappler, Thorsten Cramer, Christoph A. Merten, Mathew J. Garnett, and Julio Saez-Rodriguez. Patient-specific logic models of signaling pathways from screenings on cancer biopsies to prioritize personalized combination therapies. *Molecular Systems Biology*, 16:e8664, 2020.

[52] A. Elena. *Robustesse des réseaux d'automates booléens à seuil aux modes d'itération. Application à la modélisation des réseaux de régulation génétique*. PhD thesis, Université Grenoble 1 – Joseph Fourier, 2009.

[53] François Fages and Sylvain Soliman. Abstract interpretation and types for systems biology. *Theoretical Computer Science*, 403:52–70, 2008.

[54] Nazim Fatès. A guided tour of asynchronous cellular automata. In *Proceedings of the International Workshop on Cellular Automata and Discrete Complex Systems*, volume 8155 of *Lecture Notes in Computer Science (LNCS)*, pages 15–30. Springer, 2013.

[55] Aurélien Fauré, Adrien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22:e124–e131, 2006.

[56] Patrik Floréen and Pekka Orponen. On the computational complexity of analyzing Hopfield nets. *Complex Systems*, 3:577–587, 1989.

[57] Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. Fixed points and attractors of reaction systems. In *Proceedings of the Conference on Computability in Europe*, volume 8493 of *Lecture Notes in Computer Science (LNCS)*, pages 194–203. Springer, 2014.

[58] Guilhem Gamard, Pierre Guillon, Kévin Perrot, and Guillaume Theyssier. Rice-like theorems for automata networks. In *Proceedings of the Symposium of the Theoretical Aspects of Computer Science*, volume 187 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:17. Schloss Dagstuhl, 2021.

[59] Abhishek Garg, Alessandro Di Cara, Ioannis Xenarios, Luis Mendoza, and Giovanni De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24:1917–1925, 2008.

[60] Carlos Gershenson. Updating schemes in random Boolean networks: do they really matter? In *Proceedings of the International Conference on Simulation and Synthesis of Living Systems*, pages 238–243. MIT Press, 2004.

[61] Leon Glass and Stuart A. Kauffman. Logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.

[62] Albert Goldbeter. A model for circadian oscillations in the Drosophila period protein (PER). *Proceedings of the Royal Society of London B: Biological Sciences*, 261:319–324, 1995.

[63] Eric Goles, Fabiola Lobos, Gonzalo A. Ruz, and Sylvain Sené. Attractor landscapes in Boolean networks with firing memory: a theoretical study applied to genetic networks. *Natural Computing*, 19:295–319, 2020.

[64] Eric Goles and Servet Martínez. *Neural and automata networks: dynamical behavior and applications*, volume 58 of *Mathematics and Its Applications*. Kluwer Academic Publishers, 1990.

[65] Eric Goles and Mathilde Noual. Block-sequential update schedules and Boolean automata circuits. In *Proceedings of the International Workshop on Cellular Automata and Discrete Complex Systems*, pages 41–50. Discrete Mathematics and Theoretical Computer Science, 2010.

[66] Eric Goles and Jorge Olivos. Comportement périodique des fonctions à seuil binaires et applications. *Discrete Applied Mathematics*, 3:93–105, 1981.

[67] Eric Goles and Lilian Salinas. Sequential operator for filtering cycles in Boolean networks. *Advances in Applied Mathematics*, 45:346–358, 2010.

[68] Jean-Luc Gouzé. Positive and negative circuits in dynamical systems. *Journal of Biological Systems*, 6:11–15, 1998.

[69] Alex Graudenzi and Roberto Serra. A new model of genetic networks: the gene protein Boolean network. In *Proceedings of the Workshop italiano su vita artificiale e calcolo evolutivo*, pages 283–291. World Scientific, 2009.

[70] Alex Graudenzi, Roberto Serra, Marco Villani, Annamaria Colacci, and Stuart A. Kauffman. Robustness analysis of a Boolean model of gene regulatory network with memory. *Journal of Computational Biology*, 18:559–577, 2011.

[71] Alex Graudenzi, Roberto Serra, Marco Villani, Chiara Damiani, Annamaria Colacci, and Stuart A. Kauffman. Dynamical properties of a Boolean model of gene regulatory network with memory. *Journal of Computational Biology*, 18:1291–1303, 2011.

[72] Paul E. Hardin, Jeffrey C. Hall, and Michael Rosbah. Feedback of the Drosophila period gene product on circadian cycling of its messenger RNA levels. *Nature*, 343:536–540, 1990.

[73] Inman Harvey and Terry Bossomaier. Time out of joint: attractors in asynchronous random Boolean networks. In *Proceedings of the European Conference on Artificial Life*, pages 67–75. MIT Press, 1997.

[74] Ben Hesper and Paulien Hogeweg. Bioinformatica: een werkconcept. *Kameleon*, 1:18–29, 1970.

[75] David Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8:437–479, 1902.

[76] Paulien Hogeweg. Simulating the growth of cellular forms. *Simulation*, 31:90–96, 1978.

[77] Paulien Hogeweg and Ben Hesper. Interactive instruction on population interactions. *Computers in Biology and Medicine*, 8:319–327, 1978.

[78] Shuji Ishihara, Koichi Fujimoto, and Tatsuo Shibata. Cross talking of network motifs in gene regulation that generates temporal pulses and spatial stripes. *Genes to Cells*, 10:1025–1038, 2005.

[79] Arthur Kahn. Topological sorting of large networks. *Communications of the ACM*, 5:558–562, 1962.

[80] Stuart A. Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224:177–178, 1969.

[81] Stuart A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[82] Stuart A. Kauffman. *Current topics in developmental biology*, volume 6, chapter Gene regulation networks: A theory for their global structures and behaviors, pages 145–181. Elsevier, 1971.

[83] Stuart A. Kauffman, Carsten Peterson, Björn Samuelsson, and Carl Troein. Random Boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences of the USA*, 100:14796–14799, 2003.

[84] Marcelle Kaufman, Jacques Urbain, and René Thomas. Towards a logical analysis of the immune response. *Journal of Theoretical Biology*, 114:527–561, 1985.

[85] Hannes Klarner, Alexander Bockmayr, and Heike Siebert. Computing maximal and minimal trap spaces of Boolean networks. *Natural Computing*, 14:535–544, 2015.

[86] Stephen C. Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112:727–742, 1936.

[87] Stephen C. Kleene. λ-definability and recursiveness. *Duke Mathematical Journal*, 2:340–353, 1936.

[88] Scott A. Mangan and Uri Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100:11980–11985, 2003.

[89] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Journal of Mathematical Biophysics*, 5:115–133, 1943.

[90] Karim Mekhail and Danesh Moazed. The nuclear envelope in genome organization, expression and stability. *Nature Reviews Molecular Cell Biology*, 11:317–328, 2010.

[91] Tarek Melliti, Damien Regnault, Adrien Richard, and Sylvain Sené. On the convergence of Boolean automata networks without negative cycles. In *Proceedings of International Workshop on Cellular Automata and Discrete Complex Systems*, volume 8155 of *Lecture Notes in Computer Science (LNCS)*, pages 124–138. Springer, 2013.

[92] Luis Mendoza. A network model for the control of the differentiation process in Th cells. *BioSystems*, 84:101–114, 2006.

[93] Luis Mendoza and Elena R. Alvarez-Buylla. Dynamics of the genetic regulatory network for *Arabidopsis thaliana* flower morphogenesis. *Journal of Theoretical Biology*, 193:307–319, 1998.

[94] Luis Mendoza, Denis Thieffry, and Elena R. Alvarez-Buylla. Genetic control of flower morphogenesis in Arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15:593–606, 1999.

[95] John Milnor. On the concept of attractor. *Communications in Mathematical Physics*, 99:177–185, 1985.

[96] Davi J. A. Moraes, Benedito H. Machado, and Daniel B. Zoccal. Coupling of respiratory and sympathetic activities in rats submitted to chronic intermittent hypoxia. *Progress in Brain Research*, 212:25–38, 2014.

[97] Katsuhiko Nakamura. Synchronous to asynchronous transformation of polyautomata. *Journal of Comuter and System Sciences*, 23:22–37, 1981.

[98] Aurélien Naldi, Jorge Carneiro, Claudine Chaouiya, and Denis Thieffry. Diversity and plasticity of Th cell types predicted from regulatory network modelling. *PLoS Computational Biology*, 6:e1000912, 2010.

[99] Aurélien Naldi, Céline Hernandez, Wassim Abou-Jaoudé, Pedro T. Monteiro, Claudine Chaouiya, and Denis Thieffry. Logical modeling and analysis of cellular regulatory networks with GINsim 3.0. *Frontiers in Physiology*, 9:646, 2018.

[100] Aurélien Naldi, Céline Hernandez, Nicolas Levy, Gautier Stoll, Pedro T. Monteiro, Claudine Chaouiya, Tomáš Helikar, Andrei Zinovyev, Laurence Calzone, Sarah Cohen-Boulakia, Denis Thieffry, and Loïc Paulevé. The CoLoMoTo interactive notebook: accessible and reproducible computational analyses for qualitative biological networks. *Frontiers in Physiology*, 9:680, 2018.

[101] Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks. In *Proceedings of the International Conference on Computational Methods in Systems Biology*, volume 4695 of *Lecture Notes in Computer Science (LNCS)*, pages 233–247. Springer, 2007.

[102] Mathilde Noual. Dynamics of circuits and intersecting circuits. In *Proceedings of the International Conference on Language and Automata Theory and Applications*, volume 7183 of *Lecture Notes in Computer Science (LNCS)*, pages 433–444. Springer, 2012.

[103] Mathilde Noual. *Updating automata networks*. PhD thesis, École normale supérieure de Lyon, 2012.

[104] Mathilde Noual and Sylvain Sené. Towards a theory of modelling with Boolean automata networks – I. Theorisation and observations. arXiv:1111.2077, 2011.

[105] Mathilde Noual and Sylvain Sené. Synchronism versus asynchronism in monotonic Boolean automata networks. *Natural Computing*, 17:393–402, 2018.

[106] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1995.

[107] Loïc Paulevé. Pint: a static analyzer for transient dynamics of qualitative networks with IPython interface. In *Proceedings of the International Conference on Computational Methods in Systems Biology*, volume 10545 of *Lecture Notes in Computer Science (LNCS)*, pages 309–316. Springer, 2017.

[108] Loïc Paulevé. Reduction of qualitative models of biological networks for transient dynamics analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15:1167–1179, 2018.

[109] Loïc Paulevé, Juraj Kolčák, Thomas Chatain, and Stefan Haar. Reconciling Qualitative, Abstract, and Scalable Modeling of Biological Networks. *Nature Communications*, 11:4256, 2020.

[110] Loïc Paulevé and Sylvain Sené. Non-deterministic updates of Boolean networks. In *Proceedings of the International Workshop on Cellular Automata and Discrete Complex Systems*, volume 90 of *OpenAccess Series in Informatic (OASIcs)*, pages 10:1–10:16. Schloss Dagstuhl, 2021.

[111] Loïc Paulevé. Notebooks demonstrating Most Permissive Boolean Networks. 2020.

[112] Élisabeth Remy, Brigitte Mossé, Claudine Chaouiya, and Denis Thieffry. A description of dynamical graphs associated to elementary regulatory circuits. *Bioinformatics*, 19 (Suppl. 2):ii172–ii178, 2003.

[113] Élisabeth Remy, Paul Ruet, and Denis Thieffry. Graphic requirement for multistability and attractive cycles in a Boolean dynamical framework. *Advances in Applied Mathematics*, 41:335–350, 2008.

[114] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74:358–366, 1953.

[115] A. Richard. Positive circuits and maximal number of fixed points in discrete dynamical systems. *Discrete Applied Mathematics*, 157:3281–3288, 2009.

[116] A. Richard. Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics*, 44:378–392, 2010.

[117] A. Richard and J.-P. Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155:2403–2413, 2007.

[118] Adrien Richard. Fixed point theorems for boolean networks expressed in terms of forbidden subnetworks. *Theoretical Computer Science*, 583:1–26, 2015.

[119] Martín Ríos Wilson. *On automata networks dynamics: an approach based on computational complexity theory*. PhD thesis, Universidad de Chile and Université d'Aix-Marseille, 2021.

[120] Martín Ríos Wilson and Guillaume Theyssier. On symmetry versus asynchronism: at the edge of universality in automata networks. arXiv:2105.08356, 2021.

[121] François Robert. Itérations sur des ensembles finis et automates cellulaires contractants. *Linear Algebra and its Applications*, 29:393–412, 1980.

[122] François Robert. *Discrete iterations: a metric study*, volume 6 of *Springer Series in Computational Mathematics*. Springer, 1986.

[123] Guillermo Rodrigo and Santiago F. Elena. Structural discrimination of robustness in transcriptional feedforward loops for pattern formation. *PLoS One*, 6:e16904, 2011.

[124] Gonzalo A. Ruz, Eric Goles, M. Montalva, and Gary B. Fogel. Dynamical and topological robustness of the mammalian cell cycle network: A reverse engineering approach. *Biosystems*, 115:23–32, 2014.

[125] Yolanda Schaerli, Andreea Munteanu, Magüi Gili, James Cotterell, James Sharpe, and Mark Isalan. A unified design space of synthetic stripe-forming networks. *Nature Communications*, 5:4905, 2014.

[126] Amita Sehgal, Jeffrey L. Price, Bernice Man, and Michael W. Young. Loss of circadian behavioral rhythms and per RNA oscillations in the Drosophila mutant timeless. *Science*, 263:1603–1606, 1994.

[127] Sylvain Sené. Sur la bio-informatique des réseaux d'automates. Université d'Évry – Val d'Essonne, 2012.

[128] Mau-Hsiang Shih and Jian-Lang Dong. A combinatorial analogue of the Jacobian problem in automata networks. *Advances in Applied Mathematics*, 34:30–46, 2005.

[129] H. Siebert and A. Bockmayr. Temporal constraints in the logical analysis of regulatory networks. *Theoretical Computer Science*, 391:258–275, 2008.

[130] El Houssine Snoussi. Necessary conditions for multistationarity and stable periodicity. *Journal of Biological Systems*, 6:3–9, 1998.

[131] Christophe Soulé. Graphical requirements for multistationarity. *ComPlexUs*, 1:123–133, 2003.

[132] Christophe Soulé. Mathematical approaches to differentiation and gene regulation. *Comptes Rendus – Biologies*, 329:13–20, 2006.

[133] Cui Su and Jun Pang. CABEAN: a software for the control of asynchronous Boolean networks. *Bioinformatics*, 37:879–881, 2020.

[134] The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408:796–815, 2000.

[135] Michel Thellier, Jacques Demongeot, Vic Norris, Janine Guespin, Camille Ripoll, and René Thomas. A logical (discrete) formulation for the storage and recall of environmental signals in plants. *Plant Biology*, 6:590–597, 2004.

[136] René Thom. *Structural stability and morphogenesis*. W. A. Benjamin, Inc., 1975.

[137] R. Thomas. On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. In *Numerical methods in the study of critical phenomena*, volume 9 of *Springer Series in Synergetics*, pages 180–193. 1981.

[138] R. Thomas. Logical vs. continuous description of systems comprising feedback loops: the relation between time delays and parameters. *Studies in Physical and Theoretical Chemistry*, 28:307–321, 1983.

[139] René Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42:563–585, 1973.

[140] René Thomas. Regulatory networks seen as asynchronous automata: a logical description. *Journal of Theoretical Biology*, 153:1–23, 1991.

[141] René Thomas and Richard d'Ari. *Biological feedback*. CRC Press, 1990.

[142] René Thomas and Marcelle Kaufman. Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11:180–195, 2001.

[143] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[144] John von Neumann. *Theory of self-reproducing automata*. University of Illinois Press, 1966. Edited and completed by A. W. Burks.

[145] Norbert Wiener. *Cybernetics: or control and communication in the animal and the machine*. Hermann & C$^{\text{ie}}$, Paris, and John Wiley & Sons, New York, 1948.

[146] David J. Wooten, Jorge G. T. Zañudo, David Murrugarra, Austin M. Perry, Anna Dongari-Bagtzoglou, Reinhard Laubenbacher, Clarissa J. Nobile, and Réka Albert. Mathematical modeling of the *Candida albicans* yeast to hyphal transition reveals novel control strategies. *PLoS Computational Biology*, 17:e1008690, 2021.

[147] Jorge G. T. Zañudo, Maurizio Scaltriti, and Réka Albert. A network modeling approach to elucidate drug resistance mechanisms and predict combinatorial drug treatments in breast cancer. *Cancer Convergence*, 1:5, 2017.