



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

FILTERS ON DISJUNCTIVE BOOLEAN NETWORKS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN
COMPUTACIÓN
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

FRANCISCO ANTONIO PLANA PERILLÁN

PROFESOR GUÍA:
JORGE PÉREZ ROJAS
PROFESOR CO-GUÍA:
ERIC GOLES CHACC

MIEMBROS DE LA COMISIÓN:
PABLO BARCELÓ BAEZA
IVÁN RAPAPORT ZIMERMANN
GONZALO RUZ HEREDIA

Este trabajo ha sido parcialmente financiado por
CONICYT-PCHA/MagísterNacional/2013-221320164

SANTIAGO DE CHILE
2014

Resumen

Una red Booleana es un modelo de redes en el cual, cada nodo o elemento de la red tiene asociado una función Booleana que determina el estado del nodo respectivo, y de esta forma la evolución de la red en el tiempo. Los puntos fijos de una red Booleana, esto es, estados particulares de la red que permanecen constantes en el tiempo, han ganado importancia, por ejemplo, en el contexto de redes de regulación génica, donde los puntos fijos tienen un correlato biológico. Para cada red finita y estado inicial posible, dentro de una cantidad finita de actualizaciones de la red, los estados de la red alcanzarán un punto fijo o un ciclo límite, la cual es una secuencia de estados de la red que se repiten a lo largo del tiempo. Puntos fijos y ciclos límites son denominados atractores de la red.

En este trabajo nos concentramos en estudiar ciertos aspectos de los atractores de redes Booleanas, incluyendo aspectos computacionales, caracterizaciones, entre otros, utilizando la noción de “filtro” de una red. Un filtro es un procedimiento consistente en aplicar de forma iterativa transformaciones a una red, cada una de las cuales simula con dinámica paralela cierto modo de actualización, produciendo una nueva red cuyas propiedades y dinámica pueden ser relacionadas con la red inicial. Se ha mostrado que estos filtros pueden ser muy útiles, dado que filtros asociados a actualizaciones secuenciales pueden entregar información eficientemente sobre los puntos fijos de una red (Goles y Salinas 2010).

Nuestro análisis se restringe a redes Booleanas disyuntivas, lo cual permite concentrarse solo en la topología de la red. Nos concentramos además en esquemas de actualización bloque-secuencial, los cuales son una generalización de los esquemas paralelo y secuencial. Los principales resultados de este trabajo establecen cotas polinomiales para la complejidad de tiempo de un filtro, así como condiciones sobre la red y esquema de entrada que aseguran ciertas propiedades en la red de salida, incluyendo la remoción de ciclos límites. Los resultados obtenidos hacen uso de teoría de matrices positivas, y fueron formulados con la ayuda de simulaciones computacionales ejecutadas con una aplicación desarrollada para este fin.

Abstract

A Boolean network is a network model in which every node or element of the network has an associated Boolean function that determines the respective node state, and thus the evolution of the network over time. Fixed points of Boolean networks, that is, particular states of the network that remain constants over time, have gained importance, for example, in the context of gene regulatory networks, where fixed points have a biological correlate. For every finite network and every possible initial state, in a finite number of updates of the network the state of the network will reach either a fixed point or a limit cycle, which is a sequence of network states that repeats over time. Fixed points and limit cycles are called attractors of the network.

In this work we concentrate on studying certain aspects of attractors of Boolean networks, including computational aspects, characterizations, and so on, using the notion of a “filter” for a network. A filter is a procedure consisting in iteratively applying transformations to the network, each of which simulates certain manner to update the network with parallel dynamics, producing a new network whose properties and dynamics can be related to the initial network. It has been shown that these filters can be very useful as filters associated to sequential updates deliver information efficiently about its fixed points (Goles and Salinas 2010).

We restrict our analysis to disjunctive Boolean networks, allowing us to concentrate only in the topology of the network. We focus also in block-sequential update schedules, which are a generalization of the parallel and sequential update schedules. The main results of this work establish polynomial bounds for the time complexity of a filter, and conditions over the input network and schedule ensuring some properties in the output network including the removal of limit cycles. The results obtained make use of positive matrix theory, and were formulated with the aid of computational simulations performed with a computer application developed to that effect.

In memory of Marta and Toño, who would certainly share my happiness

Acknowledgments

In first place, I want to thank to my parents, Carmen and Sergio, by its endless support and understanding throughout my life. From Sergio I have learned first hand to do the right thing, although usually is not the most comfortable or pleasant. Carmen has taught me every day that unconditional love not only exists, but is powerful and healing. The decision to continue education immediately after completing engineering studies, is not the obvious path to take, particularly with the excessive economic cost of higher education in this side of the world. My thanks go to them for their support and love at all times.

My sister Javiera deserves special mention. She has been with me the last couple of years in Santiago, giving me support, valuable advice, and I owe her many life lessons, which are the most valuable. I have been given the opportunity to share with her joy, fun, conversations, difficulties, cries and friends. Thank you, thanks a lot.

While I have studied in Santiago the last few years, most of my immediate family lived in Curicó. Lucía, Antonia and Checa, despite the distance that separate us, I have always felt you close. I am also very grateful of the nice tea meetings with our cousin Paty, the musical enthusiasm of Francisco, the weird affection of our uncle Miguel, and the support from afar of uncle Poque and aunt Kena.

I must mention my aunt's family, The Díaz Perillán family -Cecilia, José, Gonzalo, Francisca, Consuelo and Marta-, because they have always supported me, and visiting them is really a break in my life as a student from Santiago. I always laugh a lot with them, although sometimes my chores had my head elsewhere. Particularly, I am grateful to my uncle José, Pepe, who is a graduated engineer from this faculty. He was the first person who told me, some summer day at Iloca, that there was a degree called mathematical engineering. He has always cared about me and my sister.

I am particularly grateful to all the guidance, discussions and feedback given by my thesis tutors, Eric and Jorge. Both have been important examples to me that it is possible to combine rigorous scientific work with the wisdom of living well. From the beginning, Eric has been enormously generous, for giving me the opportunity to work on some of the topics studied by him, for his valuable advice about pursuing an academic career, and even after finishing the thesis, with his invitation to work in France. I must thank here also the willingness and expertise of the group associated with the doctoral program in complex systems from the Universidad Adolfo Ibáñez -from which Eric is member-, specially to professors Gonzalo Ruz and Marco Montalva. Jorge is one of those few scientists who genuinely cares about scientific divulgation and the social value of science. Virtues that the text of this thesis may

have regarding its structure and redaction are mainly due to the thoroughness of the reviews made by Jorge. Both Jorge and Eric contributed with financial support extracted from their research projects during the realization of this thesis.

I am grateful and proud of the role that Universidad de Chile has achieved, being an agent of critical reflection of society -despite starvation which undergoes-, and the high standards of quality in training and research that its Facultad de Ciencias Físicas y Matemáticas (FCFM) can exhibit as an institutional heritage. I would like to thank in this place to the collaborative staff of the University, specially to Eterin, Mrs. Gladys, Mrs. Silvia, Mr. Oscar and Mr. Luis. Their work is absolutely essential and generally not valued appropriately.

I would like to thank also to my group of friends from school. Although the circumstances are constantly changing and the time flies, they remain available to listen and make sense to everything else. By this reason, this thesis and many other things would not be possible without the presence of Seba, Cristian, Javi, Josefa and Beatriz. I am very fortunate to have met you guys!

I would like to thank some of my fellow classmates, both on specialty and common plan courses, to Valentina, Sandra, Pedro -who introduced me to Professor Eric-, Rifo, Abelino, Mónica, Poly, Johan, Emilio, César, Chino, Tata, Anne, Chiri, Tomacho and Seba. Probably without their moral and academic support I would still be trying to finish my studies. Most of these guys are now continuing graduate studies, many of them abroad. I wish them all the best.

Finally, my time of work on the thesis, roughly from mid 2012, has coincided with my time belonging to the Escuela Carnavalera Chinchintirapié. My partners from this group have helped me to open unknown doors, to recover the ability to pursue other dreams -which is simply dreaming-, and thanks to them I have been given such an amount of affection and joy, that I will never finish to give it back. I am talking about Paloma, Jana, Gina, Manu, Esteban, Eduardo, Marifer, Catalina, the CVC group, the wind ensemble and the group of figurines.

The work of this thesis has been partially supported by Project Fondecyt 1140090, Project Fondecyt Regular 1140790 and CONICYT-PCHA/MagísterNacional/2013-221320164.

Contents

Contents	xii
Introduction	1
1 Definitions and theoretical framework	5
1.1 Preliminaries	5
1.2 Boolean networks	6
1.3 Update schedules	7
1.4 Dynamics of Boolean networks	8
1.5 Attractors: fixed points and limit cycles	10
1.6 Gauss-Seidel Operator	11
1.7 Filters	12
1.8 Outlook	15
2 Disjunctive Boolean networks	16
2.1 Comparison with conjunctive networks	17
2.2 Characterization of Gauss-Seidel network	18
3 Matrix representation	21
3.1 Theory of positive matrices	21
3.2 Relation with disjunctive networks	24
4 Filter convergence	35
4.1 Computing the Gauss-Seidel operator	35
4.2 Convergence for two blocks	37
4.3 Complexity of computing the attractor	44
5 Filtering dynamic cycles	47
5.1 The Cycle-filter condition	47
5.2 Sufficiency of Cycle-filter	52
5.3 Complexity of checking Cycle-filter	54
5.4 Complexity of checking positive circuits	58
6 Filter convergence to fixed points	61
6.1 Schedules of two blocks	61
6.2 The general case	64

Conclusion	68
7.1 Main theorem	68
7.2 Applications	69
7.3 Outlook and future work	70
A Proofs	74
A.1 Proofs for Section 1	74
A.2 Proofs for Section 2	75
A.3 Proofs for Section 3	76
A.4 Proofs for Section 4	77
A.5 Proofs for Section 5	85
A.6 Relation between a block and the previous	90
B Software	92
B.1 Dynamics simulation	93
B.2 Computing the filter	94
Bibliography	97

Introduction

A Boolean network is a mathematical model proposed by Kauffman in the late 60's, initially to address the problem of spontaneous generation of order in complex systems [54], [55]. A Boolean network consists of a finite set of Boolean functions, and is referred as a “network” since it can be represented as a digraph, where every node has a Boolean state that is the output of the respective Boolean function. The value returned by this function depends on the states of the neighbor nodes, and determines in turn, the evolution of the configuration of the network states over time.

These models have acquired importance in last years as a tool to simulate the dynamics of gene regulatory networks, since the article published in 1998 by Mendoza & Alvarez-Buylla [71], which studies floral morphogenesis of *Arabidopsis thaliana*. Unlike other models based on differential equations, which exploit reaction kinetics in terms of rates and concentrations, Boolean networks do not require a large number of biochemical parameters difficult to estimate [87]. Although these networks are a simplification of the observed phenomenon -since they do not consider spatial aspects or continuous nature of gene and protein levels-, there are arguments to discretize the involved magnitudes: the functions that map the regulatory input signal (for example, stimulating protein level) to the response (for example, enzyme activity) can be approximated by functions similar to Heaviside step functions; several mechanisms [78], [29] that would make this approximation valid have been suggested.

In our setting, each function from the Boolean network is updated according to a deterministic ordering. For example, in the sequential case, functions are updated one after other one in some established order. In the parallel case, all network functions are updated at the same time. Or it can be a mixture of previous modes, which is called the block-sequential schedule, where some node groups are updated in parallel, and others in sequential way. Since the total number of network configurations is finite and the update mode is deterministic, the network states configuration converges, in a finite number of updates of the network from every possible initial state, to equilibrium points or attractors, where depending on the period length two behaviors are distinguished: fixed points, particular states of the network that remain constants over time, and limit cycles, sequences of network states that repeat over time.

Several papers published in the last years have shown that fixed points of these networks may correspond, in many contexts, with observable biological states [28], [2], [7]. For example, the networks for the fission yeast cell cycle (the species *Saccharomyces cerevisiae* [65] and *Schizosaccharomyces Pombe* [19]), whose dominant attractors correspond with observed

stationary state G1. Or also in the network for the development of *Arabidopsis thaliana* [71], the authors achieve to identify 4 of its 6 fixed points to the four specific tissues of the flower (sepals, petals, carpels and stamens). In the other hand, limit cycles do not have a clear biological meaning.

In this work we concentrate on studying certain aspects of attractors of Boolean networks, including computational aspects, characterizations, and so on, using the notion of “filter”. A filter is a procedure consisting in iteratively applying transformations to a given network, each of these transformations simulates a certain way of updating a network under parallel dynamics. This produces finally a new network whose properties and dynamics can be related to the initial network. It has been shown that these filters can be very useful, as they deliver information efficiently about the fixed points of the input network (Goles and Salinas 2010 [39]).

In particular, the authors proved that the filtering procedure, under certain conditions over the initial network, outputs a Boolean network that preserves some attractors of the original network, the fixed points, and removes or filters out the limit cycles. By this reason, this procedure is named as “Filter”, and it can be used to compute fixed points in polynomial time. This is a non-trivial property, since it is known that the general problem of existence of fixed points for Boolean networks is NP-Complete [90]. Thereby, the filters associated to sequential updates can be seen as part of a large set of algorithms developed in recent years to find Boolean network attractors: reduced order binary decision diagrams [30], [93], scalar equation method [43], [27], based in power law [48], out-degree based gene ordering [91], SAT-based [24], constraint programming [21]. In general, existing algorithms were proposed for networks updated synchronously (in parallel), or asynchronously, where one node, selected randomly, is updated at each time.

We restrict our analysis to disjunctive Boolean networks, allowing us to concentrate only in the topology of the network (see Section 2). These are networks whose Boolean functions only have disjunctions of variables. It is a good start point to restrict the goal of study in this sense, given the difficulties that may arise when functions defining the network become slightly more expressive. Potential complex or intractable behaviors in disjunctive networks may be evidence suggesting these difficulties.

There are theoretical works that have studied the phenomenon of distinct update schedules producing different dynamics in the same network [38]. The lack of empirical knowledge about the order in which genetic regulations occur in biological networks, there makes no arguments to support the election of a more realistic iteration mode. However, community of biologists tend to agree that it is unlikely that genes involved in the same physiological function evolve in parallel [49]. Other paper [59] affirms that a large fraction of attractors are an artifact produced by the parallel update, in the sense they are unstable to small perturbations or shifts of update events. By these reasons, it is of interest the study of Boolean network dynamics under more realistic and general update schedules than the parallel update. For this reason, we focus also in block-sequential update schedules (see Subsection 1.3), which are a generalization of the parallel and sequential update schedules. These schedules have been shown consistent with empirical data. In fact, in the seminal work by Mendoza & Alvarez-Buylla [71], a specific block-sequential update schedule was used in agreement to

available experimental data regarding the activation order of gene groups.

The main results of this work establish polynomial bounds for the time complexity of a filter (Sections 3 and 4), and conditions over the input network and schedule ensuring some properties in the output network including the removal of limit cycles (Sections 5 and 6). The polynomial time for computing a filter (Theorem 7.1) is a critical issue if the filter is thought of as an algorithm to find Boolean network attractors. This bound is derived from a general bound for the converge time to attractors in disjunctive Boolean networks (Theorem 3.9). However, there are some cases in which the time for computing a filter could grow even super-polynomially; this situation is addressed in Theorem 4.6. This intractable behavior appears when the filtering procedure converges to a set called the filter attractor that results to have a super-polynomial size. This behavior can be avoided, for example, when the filter attractor is a structural fixed point, that is, the filter attractor is a set formed by only one network. Theorem 6.2 establishes a condition on the input network that ensures that the filter attractor is a structural fixed point. In the other hand, Theorem 5.5 establishes a condition on the input network that ensures that the networks in the filter attractor do not have limit cycles when updated with the parallel schedule. The results obtained make use of positive matrix theory (see Subsection 3.1), and were formulated with the aid of computational simulations performed with the purpose of testing the working hypothesis, thanks to a computer application (Appendix B) developed to that effect.

Chapter 1

Definitions and theoretical framework

1.1 Preliminaries

A *digraph* (or *directed graph*) $D = (V, A)$ consists of a finite set V of elements called *nodes* (or *vertices*) and a prescribed set A of ordered pairs of (not necessarily distinct) vertices of V . Every ordered pair $\alpha = (a, b)$ of vertices a and b in A is called an *arc* (or *directed edge*) of the digraph D . For the arc $\alpha = (a, b)$, the vertices a and b are called the *endpoints* of α , a is called the *initial vertex* and b is called the *terminal vertex*; also, it is said that vertex a is *incident to* vertex b . The vertex set of D is referred to as $V(D)$, and its arc set as $A(D)$.

A *subdigraph* of D is a digraph $D' = (V', A')$ where $V' \subseteq V$ and $A' \subseteq (V' \times V') \cap A$. We write $D' \subseteq D$. For a digraph $D = (V, A)$ and a set $V' \subseteq V$, we define the subdigraph *induced in D by V'* , which we denote as $D(V')$, as the digraph whose vertex set is V' , and its arc set is $(V' \times V') \cap A$.

For a digraph $D = (V, A)$, it is defined the *transpose* (or *reverse*) of D , which we will denote as D^T , as the digraph having the same vertex set V , and whose arcs are reversed with respect to the corresponding arcs of A , that is, $D^T = (V, A^T)$, where $A^T = \{(u, v) | (v, u) \in A\}$.

For a digraph $D = (V, A)$, it is defined the *in-neighborhood* $N_D^-(i)$ of a node $i \in V$, as the set of incident vertices to i : $N_D^-(i) = \{j \in V | (j, i) \in A\}$. The *in-degree* of i is the cardinal of this set, denoted as $\deg_D^-(i)$ ($\deg_D^-(i) = |N_D^-(i)|$). The *out-neighborhood* $N_D^+(i)$ of a node $i \in V$, is defined as the set of vertices at which i is incident: $N_D^+(i) = \{j \in V | (i, j) \in A\}$. The *out-degree* of i is the cardinal of this set, denoted as $\deg_D^+(i)$ ($\deg_D^+(i) = |N_D^+(i)|$).

A *directed walk* from vertex v_0 to vertex v_m in a digraph D is a sequence of vertices v_0, v_1, \dots, v_m of $V(D)$ such that $(v_k, v_{k+1}) \in A(D)$ for all $k = 0, \dots, m - 1$. We say that a walk v_0, v_1, \dots, v_m has *length* equal to m . A *directed path* (or *directed chain*) v_0, v_1, \dots, v_m is a directed walk with no repeated vertices, except perhaps v_0 and v_m . A *closed walk* or *circuit* is a directed walk v_0, v_1, \dots, v_m where $v_0 = v_m$. A *closed path*, *simple circuit* or *cycle* is a directed path v_0, v_1, \dots, v_m where $v_0 = v_m$. A *loop* is a cycle of length equal to 1.

Two vertices a and b are called *strongly connected* provided there are directed walks from a to b and from b to a . A vertex is regarded as trivially strongly connected to itself. Defined in this way, strong connectivity is an equivalence relation: reflexive, symmetric and transitive. Hence strong connectivity yields a partition $(V_i)_{i=1}^C$ of the vertex set of digraph D . The subdigraphs $D(V_1), D(V_2), \dots, D(V_C)$ induced in D by each equivalence class are called the *strongly connected components* (s.c.c.'s) of D . A digraph D is *strongly connected* if it has exactly one strongly connected component.

Let D be a digraph, and let $D(V_1), D(V_2), \dots, D(V_C)$ the strongly connected components of D . Let D^* be the digraph whose vertices are the sets V_1, V_2, \dots, V_C in which there is an arc from V_i to V_j if and only if $i \neq j$ and there is an arc in D from some vertex in V_i to some vertex in V_j . The digraph D^* is the *condensation digraph* of D . It is not difficult to prove that D^* does not have loops or closed directed walks (Lemma 3.2.2 in [13]), therefore D^* is a *directed acyclic graph* (DAG) -that is, a directed graph with no directed cycles- and consequently admits a *topological sort*, which is an ordering of its vertices such that for every arc, the initial vertex comes before the terminal vertex in the ordering.

A *graph* (or *undirected graph*) $G = (V, E)$ consists of a finite set V of elements called *nodes* (or *vertices*) and a prescribed set E of 2-element subsets of V called *edges* or *lines*. For undirected graphs we can define *walks*, *paths*, *circuits* and *cycles* similarly to what was done for digraphs. We say an undirected graph $G = (V, E)$ is *connected* if there is a path in G between every pair of vertices in V .

For a directed graph $D = (V, A)$ we can define the *underlying graph* of D as the undirected graph $G_D = (V, E_D)$, where $E_D = \{\{v, w\} | (v, w) \in A \vee (w, v) \in A\}$. We say that a digraph D is *weakly connected* if its underlying graph is connected.

Finally, we will refer to some of the basic operations of Boolean algebra as follows: the conjunction (AND) of variables x and y is denoted as $x \wedge y$, and it satisfies $x \wedge y = 1$ if and only if $x = y = 1$. The disjunction (OR) of x and y , denoted as $x \vee y$, meets that $x \vee y = 0$ if and only if $x = y = 0$. The negation (NOT), denoted as $\neg x$, is defined as follows: $\neg x = 1$ if and only if $x = 0$.

1.2 Boolean networks

This section follows conceptualizations presented in [37] and [39]. A *Boolean network* N_F of size n is defined by a *global transition function* $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, where $F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$, and for each $i \in \{1, \dots, n\}$, $f_i : \{0, 1\}^n \rightarrow \{0, 1\}$ is called the i -th *local transition function*. The vector $\vec{x} \in \{0, 1\}^n$ taken as argument by these functions is called vector of *network configurations* or *states*. The intuition behind is, if current configuration of the network is \vec{x} , after updating node i its new state is given by $f_i(\vec{x})$. The *graph associated to* N_F is the directed graph $G^F = (V, A)$, where:

- $V = \{1, \dots, n\}$
- $(i, j) \in A$ iff there exists some $\vec{x} \in \{0, 1\}^n$ such that

$$f_j(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \neq f_j(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n).$$

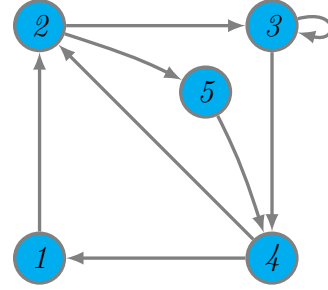
The intuition from the last definition is given by the fact that each node represents one component from the network state vector. So, there is an arc (i, j) starting from node i and ending on j , if function f_j (or in other words, the component j of network state vector after updating its value) depends on component i from the current configuration vector. As a consequence of this, the nodes without incident arcs are those corresponding to variables having a constant value, that is, those nodes j such that

$$f_j(\vec{x}) = 1 \quad \forall \vec{x} \in \{0, 1\}^n \quad \text{or} \quad f_j(\vec{x}) = 0 \quad \forall \vec{x} \in \{0, 1\}^n$$

Notice that a variable having a dependency on itself is represented in the graph as a loop: a cycle of length equal to one, connecting a node to itself. Observe that the graph associated to some Boolean network displays only dependencies existing between nodes, but without additional information the global transition function cannot be recovered from the associated graph.

Example 1 A Boolean network N_F with its respective associated digraph is the following:

$$\begin{aligned} N_F : \\ f_1(\vec{x}) &= x_4 \\ f_2(\vec{x}) &= x_1 \vee \neg x_4 \\ f_3(\vec{x}) &= x_2 \wedge x_3 \\ f_4(\vec{x}) &= x_3 \wedge x_5 \\ f_5(\vec{x}) &= \neg x_2 \end{aligned}$$



1.3 Update schedules

In order to see Boolean networks as dynamical systems, an updating procedure of the network must be specified. Let N_F be a Boolean network with global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, and $G_F = (V, A)$ be the associated digraph, where we will use the next convention: $V = \{1, \dots, n\}$. We define a general *update schedule* S by an *update function* $s_S : V \rightarrow \mathcal{P}(\{0, \dots, n-1\})$, such that for each $i \in V$, $s_S(i)$ corresponds to the set of instants at which node i is updated during one update sequence. A schedule S is represented either by an *update sequence* given by the sequence of sets $B_0^S, B_1^S, \dots, B_{m-1}^S$, such that m is the number of update instants of the sequence, and for each $j \in \{0, \dots, m-1\}$, $B_j^S \subseteq V$ refers to the set of nodes updated at instant j of the update sequence. From now on B_j^S will be referred as the j -th *block* of the schedule. Notice that the update function and the update sequence of a schedule are indeed equivalent representations, since it holds that:

$$B_j^S = \{k \in V \mid s_S(k) = j\} = s_S^{-1}(\{j\})$$

Given this equivalence between these two representations, we will refer to a update schedule as a function or as a sequence of blocks interchangeably. When there is no confusion about what schedule is referred, we will simply write B_j instead of B_j^S , and s instead of s_S . A notation frequently used is to write the nodes belonging to one update block enclosed by a pair of parentheses, as can be seen in the next example.

Example 2 Let $V = \{1, 2, 3, 4\}$ be the set of nodes of the digraph G associated to a Boolean network N_F , and let $s : V \rightarrow \mathcal{P}(\{0, 1, 2, 3\})$ the following update function: $s(1) = \emptyset$, $s(2) = \{0\}$, $s(3) = \{0, 2\}$, $s(4) = \{1\}$. A more convenient notation for this schedule is to write its update sequence $(2, 3)(4)(3)$, where nodes updated at the same time belong to some update block and are written in brackets; also the blocks are displayed in increasing order according the update instant.

Notice that, in an arbitrary update schedule with update function s , some nodes could never be updated (a node j verifying $|s(j)| = 0$), while others could be updated more than once during one update sequence (a node j verifying $|s(j)| > 1$). In this work, an important definition will be that of *block-sequential* update schedules: these are schedules where each node is updated exactly once during one update sequence, that is, for every $i \in V$, it holds that $|s(i)| = 1$. For a block-sequential update schedule, since for each node $i \in V$ the set $s(i)$ results to be a singleton, we can compare update instants with the usual order in \mathbb{N} : we say the node i is updated after node j if $s(j) < s(i)$. Also, for a block-sequential update schedule S , the update function results to be $s : V \rightarrow \{0, \dots, n-1\}$ (the update function in general is defined as $s : V \rightarrow \mathcal{P}(\{0, \dots, n-1\})$, since a node could be updated several times), the number of update instants verifies $m \leq n = |V|$, and the set of blocks $(B_j)_{j=0}^{m-1}$ is a partition of the set of nodes V .

Two important block-sequential update schedules are the following:

- The *parallel* schedule, usually denoted as π , is a schedule in which all the nodes of the network are updated at the same time. Therefore, in the parallel schedule there is just one block $B_0^\pi = V$.
- A *sequential* schedule S' is that in which one node is updated at every instant, that is, for every $t < m$, $|B_t^{S'}| = 1$, where m is the number of blocks of S' . Necessarily for these schedules, if n is the number of nodes of the network, then $m = n$. It can be assumed, without loss of generality (via a graph isomorphism), that the sequential schedule S' corresponds to the fixed permutation $(1)(2) \cdots (n)$.

1.4 Dynamics of Boolean networks

Given a Boolean network N_F of size n with global transition function F and local transition functions $(f_i)_{i=1}^n$ (that is, for each $\vec{x} \in \{0, 1\}^n$, it holds that $F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$), and some update schedule S with blocks $(B_\ell)_{\ell=0}^{m-1}$, we want to formalize the notion of network dynamics. For this, it is defined the *State Transition Graph* $STG(F, S) = (V, A)$, where $V = \{0, 1\}^n$ and $(\vec{x}, \vec{y}) \in A$ if and only if $\vec{y} = F_{[B_{m-1}]} \circ \dots \circ F_{[B_1]} \circ F_{[B_0]}(\vec{x})$, where

$$F_{[B_\ell]}(\vec{x})_i = \begin{cases} f_i(\vec{x}) & i \in B_\ell, \\ x_i & \text{otherwise} \end{cases}$$

$F_{[B_\ell]}(\cdot)$ is called the *transition function with respect to the block* $B_\ell \subseteq V$ of nodes updated in parallel. The intuition of $STG(F, S)$ is that arcs represent configuration or state transitions, from any possible state to the state obtained after a complete update sequence, where the first nodes to be updated are those in the first block B_0 , then the nodes in B_1 , and so on up to the last block B_{m-1} .

Now we can define the *global transition function of a network with update schedule* S , as the function $F_S : \{0, 1\}^n \rightarrow \{0, 1\}^n$ that outputs the new network configuration after a whole update sequence:

$$F_S(\cdot) = F_{[B_{m-1}]} \circ \dots \circ F_{[B_1]} \circ F_{[B_0]}(\cdot)$$

Example 3 Let N_F be the Boolean network of size 4 and global transition function F , given by transition functions $f_1(\vec{x}) = x_4$, $f_2(\vec{x}) = x_1 \vee x_4$, $f_3(\vec{x}) = x_1 \wedge x_2$ and $f_4(\vec{x}) = x_3$, and let S be the block-sequential schedule given by blocks $B_0 = \{2, 3\}$, $B_1 = \{1, 4\}$. If we start from state vector $\vec{y} = (0, 1, 0, 1)$, to get the transition obtained after a complete update sequence with S , we first compute the vector $F_{[B_0]}(\vec{y}) = (0, 0 \vee 1, 0 \wedge 1, 1) = (0, 1, 0, 1)$, and then the vector $F_{[B_1]}(F_{[B_0]}(\vec{y})) = F_{[B_1]}(0, 1, 0, 1) = (1, 1, 0, 0)$. If we now start from state vector $\vec{z} = (1, 1, 0, 0)$, to get the transition obtained after a complete update sequence with S , we first compute the vector $F_{[B_0]}(\vec{z}) = (1, 1 \vee 0, 1 \wedge 1, 0) = (1, 1, 1, 0)$, and then the vector $F_{[B_1]}(F_{[B_0]}(\vec{z})) = F_{[B_1]}(1, 1, 1, 0) = (0, 1, 1, 1)$.

Now, if we work with the same schedule showed in Example 2, where $V = \{1, 2, 3, 4\}$, $S = (2, 3)(4)(3)$, and we have a initial state vector $\vec{x} \in \{0, 1\}^4$, to compute the network configuration after one complete update sequence, we must compute the vector $F_S(\vec{x})$, therefore we need to compute the following values

$$\begin{aligned} F_{[\{2,3\}]}(\vec{x}) &= (x_1, f_2(\vec{x}), f_3(\vec{x}), x_4) \\ F_{[\{4\}]}(F_{[\{2,3\}]}(\vec{x})) &= (x_1, f_2(\vec{x}), f_3(\vec{x}), f_4(F_{[\{2,3\}]}(\vec{x}))) \\ F_{[\{3\}]}(F_{[\{4\}]}(F_{[\{2,3\}]}(\vec{x}))) &= (x_1, f_2(\vec{x}), f_3(F_{[\{4\}]}(F_{[\{2,3\}]}(\vec{x}))), f_4(F_{[\{2,3\}]}(\vec{x}))) = F_S(\vec{x}) \end{aligned}$$

In this example, $F_{[\{2,3\}]}(\vec{x})$, $F_{[\{4\}]}(F_{[\{2,3\}]}(\vec{x}))$, $F_{[\{3\}]}(F_{[\{4\}]}(F_{[\{2,3\}]}(\vec{x})))$ are the consecutive network configurations when the network is updated according to the sequence given by the schedule S starting from state vector \vec{x} . The final configuration obtained $F_{[\{3\}]}(F_{[\{4\}]}(F_{[\{2,3\}]}(\vec{x})))$ corresponds to the vector $F_S(\vec{x})$, the configuration of the network after a whole update sequence.

For a Boolean network N_F with global transition function $F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$, updated with a block-sequential update schedule S (and respective update function s) it is defined, for a node $i \in V$, the *local transition function* f_i^S relative to S as the i -th component function of F_S , that is

$$F_S(\vec{x}) = (f_1^S(\vec{x}), \dots, f_n^S(\vec{x}))$$

The function $f_i^s(\cdot)$ corresponds to the state that node i has once it has been updated during one update sequence and its update in the next update sequence. Recall that for block-sequential update schedules, each node is updated only once during one update sequence. Defined as above, the functions f_i^S can be characterized as follows (the proof of this can be found in Lemma A.1 of the appendix):

$$\forall \vec{x} \in \{0, 1\}^n, f_i^S(\vec{x}) = f_i(x_1^i(\vec{x}), \dots, x_n^i(\vec{x})) \text{ where } x_j^i(\vec{x}) = \begin{cases} x_j & s(i) \leq s(j), \\ f_j^S(\vec{x}) & s(i) > s(j) \end{cases}$$

1.5 Attractors: fixed points and limit cycles

We are interested in studying the limit behavior of the system given by the state updating of a Boolean network. First, we need some notation. For some update schedule S , we will employ the iterated composition of function F_S , notated as F_S^t :

$$F_S^t(\cdot) \equiv \underbrace{F_S \circ \dots \circ F_S(\cdot)}_{t \text{ times}}$$

For an initial configuration $\vec{x} \in \{0, 1\}^n$, the following notation will be used hereinafter: $\vec{x}(0) \equiv \vec{x}$ and $\vec{x}(t) \equiv F_S^t(\vec{x})$ (that is, $\vec{x}(t)$ is the network configuration after applying t times the global transition function with schedule S , F_S). Now, since the set of possible state configurations of a network is finite (with cardinal equal to 2^n if the size of the network is n), and the updating procedure is deterministic, every possible trajectory $\vec{x}(t)$ (starting from any initial state $\vec{x}(0)$) necessarily ends up looping or reaching some previous configuration. That is, for all $\vec{x}(0) \in \{0, 1\}^n$, there exist $t, p \in \mathbb{N}$, such that $\vec{x}(t+p) = \vec{x}(t)$. For any initial condition $\vec{x}(0) \in \{0, 1\}^n$, we define *the transient length of $\vec{x}(0)$* , and *the period length of $\vec{x}(0)$* , denoted respectively by $tran(\vec{x}(0))$ and $per(\vec{x}(0))$, as the smallest positive integers t and p satisfying the last equality, for a trajectory $\vec{x}(t), t \in \mathbb{N}$, starting from $\vec{x}(0)$. If there is no confusion about the initial condition, we will refer to the transient and the period simply as t and p .

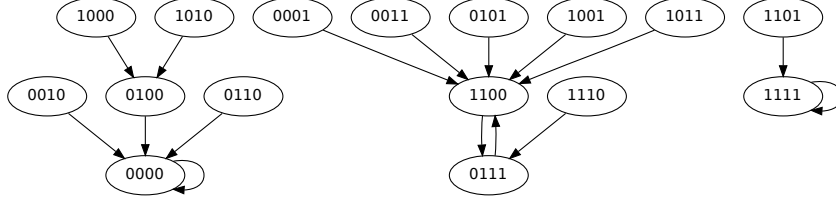
Two limit behaviors are usually characterized. When $p = 1$, or in other words, $\vec{x}(t+1) = \vec{x}(t)$, the configuration $\vec{x}(t)$ is said to be a *fixed point*. The other case, when $p > 1$ and

$$\begin{aligned} \vec{x}(t+p) &= \vec{x}(t) \\ \vec{x}(t+i) &\neq \vec{x}(t+j) \quad i \neq j, \text{ and } i, j \in \{0, \dots, p-1\} \end{aligned}$$

then the set of configurations $\{\vec{x}(t+i) | i = 0, \dots, p-1\}$ is called a *limit cycle*. Fixed points and limit cycles are denominated *attractors* of the network. Fixed points are characterized equivalently as those configurations $\vec{x}(t)$ meeting that $F_S(\vec{x}(t)) = \vec{x}(t)$, ie, configuration vectors that are fixed points for function $F_S(\cdot)$, given that:

$$\vec{x}(t+1) = F_S^{t+1}(\vec{x}(0)) = F_S(F_S^t(\vec{x}(0))) = F_S(\vec{x}(t)) = \vec{x}(t)$$

Example 4 *If we compute all the possible state transitions of a network with some update schedule, we can plot the state transition graph defined previously. Here we make this with network N_F and schedule S from Example 3, thus the image below is a picture which shows $STG(N_F, S)$:*



With this representation it is easy to visualize the attractors in the network dynamics as cycles in this graph (since (\vec{x}, \vec{y}) is a transition in this graph if and only if $\vec{y} = F_S(\vec{x})$); thus, the loops in this graph, $(0, 0, 0, 0)$ and $(1, 1, 1, 1)$, are fixed points, and the cycle of length 2 formed by state vectors $(1, 1, 0, 0)$ and $(0, 1, 1, 1)$ is a limit cycle with period length equal to 2.

1.6 Gauss-Seidel Operator

In one of the first theoretical studies about update schedules in the context of automata networks [81], F. Robert proved that Boolean networks without circuits have a unique fixed point. He also proposed a procedure, namely the Gauss-Seidel operator, which maps a network with asynchronous update (in our context, a sequential update schedule) into a synchronous model (ie, a parallel schedule). Specifically, for the sequential schedule $S = (1)(2) \dots (n)$, and a Boolean network N_F of size n and global transition function F , the output of Gauss-Seidel operator is given by $N_{F'}$, where the function $F'(x) = (f'_1(x), \dots, f'_n(x))$ is defined as follows:

$$\begin{aligned} f'_1(x) &= f_1(x) \\ f'_i(x) &= f_i(f'_1(x), \dots, f'_{i-1}(x), x_i, \dots, x_n), \quad \forall i = 2, \dots, n \end{aligned}$$

Robert proved that the iterative application of this operator on Boolean networks without circuits becomes stationary and the resulting network has the same fixed point than the original one ([81]). The synchronous simulation of sequential update done by this operator has proven useful in itself in other fields (see for example, Chapter 6 in [64]).

Let $\mathcal{N}_n = \{N_F \mid F : \{0, 1\}^n \rightarrow \{0, 1\}^n\}$ be the set of all Boolean networks of size n , and let \mathcal{S}_n be the set of all block-sequential update schedules for Boolean networks of size n . In this work, we will use a generalization of the cited operator: the operator $GS : \mathcal{N}_n \times \mathcal{S}_n \rightarrow \mathcal{N}_n$, which will be also referred as Gauss-Seidel operator, is defined as follows

$$\begin{aligned} GS : \mathcal{N}_n \times \mathcal{S}_n &\rightarrow \mathcal{N}_n \\ (F, S) &\mapsto F_S \end{aligned}$$

that is, for a global transition function F and a block-sequential update schedule S , $GS(F, S)$ returns F_S , the global transition function of network F with update schedule S , which complies that

$$STG(F, S) = STG(F_S, \pi)$$

Last equality means that Gauss-Seidel operator returns the global transition function of a network that simulates in parallel the dynamics of $STG(F, S)$, that is, one update of F_S in parallel, produces the same output of one update of network F with schedule S .

1.7 Filters

From the contribution of Robert of having applied the Gauss-Seidel operator in iterative manner, a line of investigation has initiated concerning the properties of this procedure. We need to introduce some notation necessary for the study and discussions conducted on this section and the rest of the entire work. For a Boolean network N and a block-sequential update schedule S , we define the *Filter* associated to N and S as the following system

$$\begin{aligned} N^0 &\equiv N \\ N^{i+1} &\equiv GS(N^i, S), \quad i \geq 0 \end{aligned} \tag{1.1}$$

The Gauss-Seidel operator receives as input a Boolean network of size n ($|V| = n$) and returns a Boolean network of the same order. The space of the Boolean networks of size n is finite (there are n local transition functions, and the number of possible non-equivalent Boolean functions of n variables is bounded by the number of different truth tables of n variables, then the total is at most $(2^{2^n})^n$), then necessarily the iterated application of Gauss-Seidel operator ends in a loop of networks. Formally, there exist $\bar{t}, \bar{p} \in \mathbb{N}$ such that

$$N^j \neq N^{j+1}, \quad j \in \{\bar{t}, \dots, \bar{t} + \bar{p} - 1\} \tag{1.2}$$

$$N^{\bar{t} + \bar{p}} = N^{\bar{t}} \tag{1.3}$$

We define the *filter attractor of network N and schedule S* , $\mathcal{A}(N, S)$, as the set of Boolean networks of size n which satisfy

$$\mathcal{A}(N, S) = \{N^l \mid l = \bar{t}, \dots, \bar{t} + \bar{p} - 1\}$$

We see that $|\mathcal{A}(N, S)| = p$; in the special case that $p = 1$, the filter attractor is said to be a *structural fixed point*, while if $p > 1$, the attractor is named a *structural cycle*. Analogously to the context of dynamical attractors of a Boolean network seen in Subsection 1.5, we can talk here of *the transient length of a network N* , $t(N)$, and *the period length of a network N* ,

$p(N)$, as the smallest positive integers verifying the relation (1.3). If there is no confusion about the initial network, we will refer to the transient and the period simply as t and p .

It is not difficult to see that fixed points of final network are the same as the original input network, since it is known that fixed points of a network are the same for all block-sequential schedules. Indeed, it suffices to prove the following:

Lemma 1.1 *Let $F(\cdot) = (f_1(\cdot), \dots, f_n(\cdot))$ be the global transition function of some Boolean network of size n and S be a block-sequential update schedule. It holds that $\vec{z} \in \{0, 1\}^n$ is a fixed point of $F(\cdot)$ if and only if \vec{z} is a fixed point of $F_S(\cdot)$.*

PROOF. Let $(B_l)_{l=0}^{m-1}$ be the block set and s the update function of schedule S .

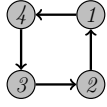
If \vec{z} is a fixed point of F , each component function outputs $f_i(\vec{z}) = z_i$ for all $i \in \{1, \dots, n\}$. Thus, $F_{[B_0]}(\vec{z}) = \vec{z}$, and $F_{[B_1]}(F_{[B_0]}(\vec{z})) = F_{[B_1]}(\vec{z}) = \vec{z}$, and so on it is concluded that $F_S(\vec{z}) = \vec{z}$.

Now, if $F_S(\vec{z}) = \vec{z}$, then, for every $i \in \{1, \dots, n\}$, the i -th component function of F_S verifies $f_i^S(\vec{z}) = z_i$. Then, for every $i, j \in \{1, \dots, n\}$, the quantity $x_j^i(\vec{z})$ given by characterization in Lemma A.1 (valid for block-sequential schedules) holds that $x_j^i(\vec{z}) = z_j$, then by the same characterization $f_i^S(\vec{z}) = f_i(x_1^i(\vec{z}), \dots, x_n^i(\vec{z})) = f_i(z_1, \dots, z_n) = f_i(\vec{z})$, which implies $F(\vec{z}) = \vec{z}$. \square

Since the schedule S in the statement of last result is any block-sequential schedule, it holds that fixed points of a network are the same for all block-sequential schedules. Now, in the system (1.1), by definition of Gauss-Seidel operator, for all $i \geq 0$, N^{i+1} has the same fixed points that network N^i . Hence, fixed points of networks in the filter attractor are the same as the original input network.

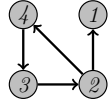
Other interesting property shown by this procedure is the fact that, the parallel simulation of a network, in some manner reduces the communication paths between nodes, which would result in a shortening of attractor periods, and finally a remotion of some limit cycles (if not all). In the work by Goles & Salinas (2010, [39]), it is proved that, under some conditions (in Section 5.4 we formalize this condition and the result, see Theorem 5.8), the application of Gauss-Seidel operator associated to the sequential operator (1)(2) ... (n) onto a Boolean network of size n , outputs after $n-1$ iterations, a network whose only attractors, with parallel update, are fixed points. This can be seen in the following example:

Example 5 *In this example, extracted from [39], we have the network N of size 4 with local transition functions given by $f_1(x) = x_2$, $f_2(x) = \neg x_3$, $f_3(x) = \neg x_4$ and $f_4(x) = x_1$. We will write, $N^0 = N$, and for $i \geq 1$, $N^i = \mathcal{S}(N^{i-1})$ the network obtained after i applications onto N of operator \mathcal{S} , which corresponds to operator $\mathcal{S}(\cdot) = GS(\cdot, (1)(2)(3)(4))$ (\mathcal{S} is the notation employed in [39]).*



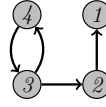
N

$$\begin{aligned} f_1(\vec{x}) &= x_2 \\ f_2(\vec{x}) &= \neg x_3 \\ f_3(\vec{x}) &= \neg x_4 \\ f_4(\vec{x}) &= x_1 \end{aligned}$$



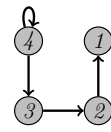
N^1

$$\begin{aligned} f_1(\vec{x}) &= x_2 \\ f_2(\vec{x}) &= \neg x_3 \\ f_3(\vec{x}) &= \neg x_4 \\ f_4(\vec{x}) &= x_2 \end{aligned}$$



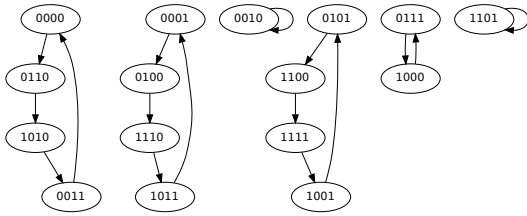
N^2

$$\begin{aligned} f_1(\vec{x}) &= x_2 \\ f_2(\vec{x}) &= \neg x_3 \\ f_3(\vec{x}) &= \neg x_4 \\ f_4(\vec{x}) &= \neg x_3 \end{aligned}$$

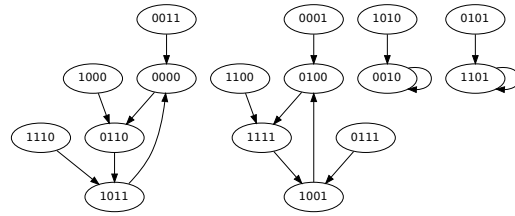


N^3

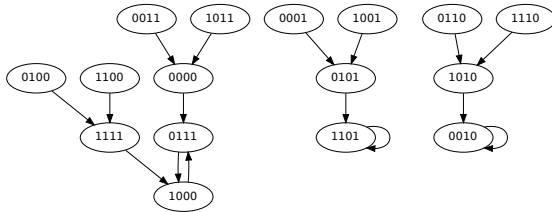
$$\begin{aligned} f_1(\vec{x}) &= x_2 \\ f_2(\vec{x}) &= \neg x_3 \\ f_3(\vec{x}) &= \neg x_4 \\ f_4(\vec{x}) &= x_4 \end{aligned}$$



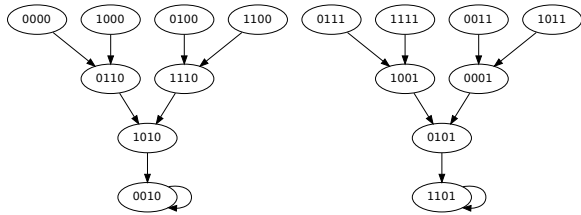
Dynamics of N



Dynamics of N^1



Dynamics of N^2



Dynamics of N^3

It can be seen that iterated composition of operator \mathcal{S} on network N of size 4 stabilizes at network $N^3 = \mathcal{S} \circ \mathcal{S} \circ \mathcal{S}(N)$, that is, $N^4 = N^3$. The only attractors of N^3 -updating N^3 with parallel schedule- are fixed points, which necessarily match with fixed points from initial network N .

The result cited above motivated the authors of [39], to name the iterative application or composition of Gauss-Seidel operator as filter, given that this procedure finally outputs, at least under some conditions met by the input network, a new network where the cyclic attractors are “filtered” out or removed, therefore its only attractors are fixed points.

Another important feature, specially for practical applications, is the size of the transient and period length, if this magnitude can be polynomially bounded in terms of the size of the network. In more colloquial terms, how many iterations of Gauss-Seidel are necessary for the filter to stabilize or to reach some network in the filter attractor. Theorem 5.8 also assures that, for sequential schedules, the transient length for some kind of Boolean networks

is always linear, and the filter attractor is a structural fixed point. It is not known if this remains true in other or more general settings; investigate these issues is the goal of this work.

1.8 Outlook

In this work, we will study the filtering procedure with block-sequential update schedules, which are a generalization of sequential schedules studied in the paper by Goles & Salinas (2010). It was argued in Section that the framework of block-sequential schedules appears to provide a more realistic simulation of network dynamics, and it has been shown consistent with empirical data [71]. Block-sequential update schedules are a generalization also of the synchronous or parallel simulation, a well studied topic from a theoretical standpoint.

In this work, we are interested on the information that can be extracted from the topology of the digraph associated to a network. It is a good start point to restrict the goal of study to this aspect, given the difficulties that may arise when functions defining the network become slightly more expressive. By this reason, we concentrate on disjunctive Boolean networks, which are introduced in Section 2. Section 3 explains some of the advantages and properties when considering this kind of networks. Section 4 addresses the issue of the time required by the filtering procedure. Section 5 investigates the condition over the initial network to ensure the remotion of limit cycles in the final network. Finally, Section 6 studies when the filter attractor results to be a structural fixed point.

The methodology in this work included the development of software intended for computing the network outputted by the filtering procedure and network dynamics. This software was critical during the research that produced several of other results in this work, since it allowed to acquire intuition about some study objects and to test working hypothesis. Details about this software can be reviewed on Appendix B. The results whose proofs are not displayed in the body of the sections can be reviewed on Appendix A.

Chapter 2

Disjunctive Boolean networks

The main work tool in this thesis is the digraph associated with the network returned by the Gauss-Seidel operator: many conclusions can be drawn by studying only this network representation. The assumption taken on these networks is that only one logical operation appears in transition functions definition, so disjunctive Boolean networks -which only have the \vee connector- provide a suitable setting meeting this condition. This choice is given by the fact that the digraph associated with these networks is an equivalent representation of the network, in the sense that the definition of the transition functions of the network can be recovered from the digraph. This allows working with the digraph associated with the network, and so do the algorithms presented in this work, as they are defined in terms of a digraph that is received as input, and return an output digraph that represents a new network. Hereinafter in this work, we refer interchangeably to Boolean networks as digraphs.

Recall that for a digraph $D = (V, A)$, it is defined the *in-neighborhood* $N_D^-(i)$ of a node $i \in V$, as the set of incident nodes to i : $N_D^-(i) = \{j \in V | (j, i) \in A\}$. The *in-degree* of i is the cardinal of this set, denoted as $\text{deg}_D^-(i)$ ($\text{deg}_D^-(i) = |N_D^-(i)|$). The *disjunctive networks* use exclusively the *OR* logic connective (usually written as \vee) to connect variables in their transition functions. Then, for a disjunctive network, the *local transition function* f_i associated to node $i \in V$ is given by:

$$f_i : \{0, 1\}^n \longrightarrow \{0, 1\}$$
$$\vec{x} \mapsto \bigvee_{j \in N_D^-(i)} x_j$$

Unless otherwise stated, we suppose hereinafter in this work that all digraphs have non-null in-degrees: for all $i \in V$, $\text{deg}_D^-(i) > 0$. Nodes with a constant state equals 0, have no effect on the states of other nodes in the graph, thus they can be removed safely in a pre-processing of the original digraph that takes in polynomial time. Nodes with a constant state equals 1 are not considered: informally, a node k propagates its fixed state equal to 1 to every node j reachable from k and leaving out the dependency of j on other variables, producing a dynamic different from the goal of study. This is explained with more detail in Remark 1.

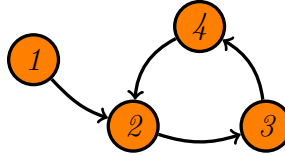
2.1 Comparison with conjunctive networks

The next result shows that the election of disjunctive networks instead of conjunctive networks (ie, networks which only have the \wedge connector) is arbitrary, since for each conjunctive network N_G , we can study its dynamics through a disjunctive network N_F which presents “dual” dynamics relative to that of N_G , in the sense specified by the next lemma (whose proof is on the appendix).

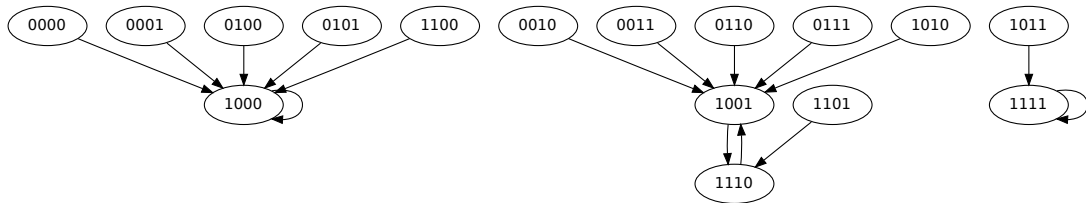
Lemma 2.1 *Let N_G be a conjunctive Boolean network defined by the local transition functions g_i ($G(\vec{x}) = (g_1(\vec{x}), \dots, g_n(\vec{x}))$), and let S be a block-sequential update schedule for this network. Then there exists a disjunctive Boolean network N_F defined by local transition functions f_i ($F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$) such that*

$$f_i^S(\vec{x}) = \neg g_i^S(\neg \vec{x}), \quad \forall i = 1, \dots, n, \quad \forall \vec{x} \in \{0, 1\}^n$$

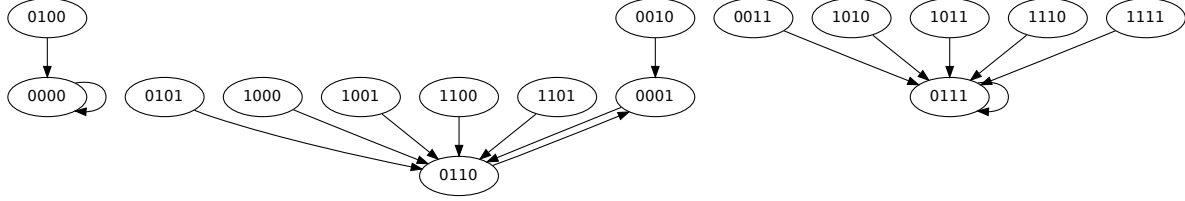
Example 6 *Let us see an example of what affirms the last result. Let N_G be a conjunctive Boolean network given by the following local transition functions: $g_1(\vec{x}) = 1$, $g_2(\vec{x}) = x_1 \wedge x_4$, $g_3(\vec{x}) = x_2$ and $g_4(\vec{x}) = x_3$. The “dual” disjunctive network N_F relative to N_G (built according the procedure employed in the proof of 2.1) has the following local transition functions: $f_1(\vec{x}) = 0$, $f_2(\vec{x}) = x_1 \vee x_4$, $f_3(\vec{x}) = x_2$ and $f_4(\vec{x}) = x_3$. Both networks have the same digraph associated, depicted below.*



Updated with the schedule $S = (12)(34)$, the network N_G has the dynamics that is illustrated in the image below.



In this second image we see the state transitions experimented by network N_F and the same schedule S . The last lemma explains what happens here, where $(\vec{x}, F_S(\vec{x}))$ is a state transition of N_F updated with S if and only if $(\neg \vec{x}, \neg G_S(\neg \vec{x}))$ is a state transition of N_G updated with S , for all $\vec{x} \in \{0, 1\}^4$.



2.2 Characterization of Gauss-Seidel network

In this subsection it is presented some of the fundamental tools employed in this work, given by the framework of disjunctive networks. For a network $D = (V, A)$ with global transition function F , and some update schedule S , it was already noticed in Section 1 that the network returned by Gauss-Seidel operator, which we name as $D^S = (V, A^S)$, is the network whose global transition function corresponds to F_S . If the component function of F_S associated to node $i \in V$ is f_i^S , then the set of arcs A^S of the network D^S corresponds (since it is the digraph associated to the network with global transition function F_S) to the following:

$$A^S = \{(j, i) \mid f_i^S(\vec{x}) \text{ depends on } x_j\}$$

The following lemma is a characterization of this set A^S , that will be very useful for subsequent arguments, since it relates the arcs of network D^S to arcs from the input network D . The proof of this result is the one provided on [37], and it is exposed here by the importance of this characterization throughout this entire work.

Lemma 2.2 ([37]) *Let $D = (V, A)$ be a Boolean disjunctive network, and let S be a block-sequential update schedule, with update function s . The set of arcs A^S of the network D^S returned by Gauss-Seidel on (D, S) is characterised by*

$$A^S = \{(j, i) \mid \text{there exists in } D \text{ a path } (v_0, v_1, \dots, v_l) \text{ from } j = v_0 \\ \text{to } i = v_l \text{ such that } s(v_0) \geq s(v_1) \wedge \forall 1 \leq k < l, s(v_k) < s(v_{k+1})\}$$

PROOF. First, let us suppose that there exists such a path from node j to node i in D . Let $\vec{x} = \vec{x}(t)$ be an arbitrary configuration of network states (on a certain time t). The configuration of the next time step, $\vec{x}(t+1) = F_S(\vec{x})$, satisfies, for each $k \in V$, $x(t+1)_k = f_k^S(\vec{x})$. Thanks to Lemma A.1 and the hypothesis met by the path in D , it also holds that $\forall 1 \leq k < l$, $x(t+1)_{i_{k+1}}$ depends on $f_{i_k}^S(\vec{x}) = x(t+1)_{i_k}$. Using this, with an induction on k this leads to $\forall 1 \leq k < l$, $x(t+1)_{i_{k+1}}$ depends on $x(t+1)_{i_1}$. By Lemma A.1 again, $x(t+1)_{i_1}$, in turn, depends on $x_{i_0} = x_j$. As a consequence, $\forall 1 \leq k < l$, $x(t+1)_{i_{k+1}}$ depends on x_j , and in particular, $x(t+1)_{i_l} = x(t+1)_i = f_i^S(\vec{x})$ depends on x_j , and so $(j, i) \in A^S$.

Now, to prove the converse let us suppose that $(j, i) \in A^S$, or equivalently that $f_i^S(\vec{x})$ depends on x_j , and we proceed by induction on $s(i)$. First suppose that $s(i) = 0$, it can only

be that $s(i) \leq s(j)$, then $x_j^i(\vec{x})$ from Lemma A.1 holds that $x_j^i(\vec{x}) = x_j$ for $j = 1, \dots, n$, so $f_i^S(\vec{x}) = f_i(\vec{x})$ depends on x_j , or in other words, $(j, i) \in A$ (ie, (j, i) is an arc from input network D). Next, suppose that $s(i) > 0$, and that the inductive hypothesis holds for each $k \in \{0, \dots, s(i) - 1\}$, that is, there is a path in D from j to k meeting the required. If $s(i) \leq s(j)$, by the same previous reasoning it holds $(j, i) \in A$, and there is a path in D from j to i with the required property. If $s(i) > s(j)$, by the first hypothesis and Lemma A.1 there is $\bar{k} \in N_D^-(i)$ such that $s(\bar{k}) < s(i)$, $x(t+1)_i$ depends on $x(t+1)_{\bar{k}}$ and $x(t+1)_{\bar{k}}$ depends on x_j . Now by induction hypothesis there exists a path with the desired properties from j to \bar{k} and, using the arc (\bar{k}, i) , this path can be extended to a path with the desired properties from j to i . \square

Remark 1 *We shall see the last characterization is optimal for disjunctive networks, in the sense that it fails when considering any additional connective than \vee in transition functions of the network. The proof of the last lemma employs the hypothesis that the network is disjunctive in applying the following property:*

Let N_F be a Boolean network defined by the local transition functions f_i ($F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$), and let S be a block-sequential update schedule for this network (with function s). Suppose $f_i(\cdot)$ depends on variable x_j , $s(i) > s(j)$ and $f_j^S(\cdot)$ depends on variable x_k . Then $f_i^S(\cdot)$ depends on variable x_k .

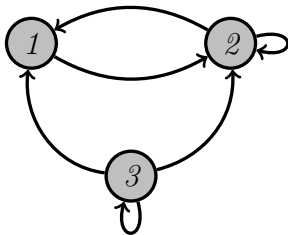
The property above is trivially true if the network is disjunctive, because every function that uses only disjunction as connective depends on every variable appearing in its definition (except when there are nodes with constant state equal to 1). This property is not true for functions where the negation connective appears, as shows the following example. Let N be a Boolean network whose local transition functions are given by $f_1(\vec{x}) = x_2 \vee x_3$, $f_2(\vec{x}) = x_1$ and $f_3(\vec{x}) = \neg x_2$, and the associated digraph D is bottom left. With the schedule of two blocks $S = (3)(1, 2)$, Gauss-Seidel outputs the network N^S defined by the functions $f_1^S(\vec{x}) = x_2 \vee \neg x_2 = 1$, $f_2^S(\vec{x}) = x_1$ and $f_3^S(\vec{x}) = \neg x_2$, and the associated digraph D^S is bottom right.



According to the characterization, due to the fact that D has the arcs $(2, 3)$ and $(3, 1)$, with $s(2) \geq s(3)$ and $s(3) < s(1)$, the network D^S should have the arc $(2, 1)$; however, this does not occur (in fact, the node labeled 1 has no incident arcs in D^S as it has a constant state equal to 1).

The property neither holds for networks without negations: let N be a Boolean network whose local transition functions are given by $f_1(\vec{x}) = x_2 \vee x_3$, $f_2(\vec{x}) = (x_1 \wedge x_3) \vee x_2$ and $f_3(\vec{x}) = x_3$, and the associated digraph D is bottom center. With the schedule of two blocks $S = (2)(1, 3)$, Gauss-Seidel outputs the network N^S defined by the functions $f_1^S(\vec{x}) = x_3 \vee f_2^S(\vec{x}) = x_3 \vee$

$(x_1 \wedge x_3) \vee x_2$, $f_2^S(\vec{x}) = (x_1 \wedge x_3) \vee x_2$ and $f_3^S(\vec{x}) = x_3$, and the associated digraph D^S (which results in this example to be equal to D) is bottom center.



$$D = D^S$$

According to the characterization, due to the fact that D has the arcs $(1, 2)$ and $(2, 1)$, with $s(1) > s(2)$, the network D^S should have the arc $(1, 1)$; however, this does not occur: although in this example $f_1(\cdot)$ depends on variable x_2 , $s(1) > s(2)$ and $f_2^S(\cdot)$ depends on variable x_1 , $f_1^S(\cdot)$ does not depend on variable x_1 (in the sense that does not exist a vector $\vec{y} \in \{0, 1\}^3$ such that $f_1^S(0, y_2, y_3) \neq f_1^S(1, y_2, y_3)$).

These two examples show that Lemma 2.2, which allows to study the network N^S returned by Gauss-Seidel through its digraph of dependencies D^S , has a well-defined range of validity established by the hypothesis.

Chapter 3

Matrix representation

In this section we will review some elements of theory of positive matrices that will be critical for some of the main results of this work. The results of this section have a broader application range of that given by Boolean networks, and are important in itself: it is interesting that algebraic properties of matrices have impact in the evolution of dynamic linear systems. However this section may result a bit technical and independent from the previous discussion, so that the reader who only looks for the main part of this section is referred to Theorem 3.9. For details of the theory of positive matrices the reader is referred to [13], [72], and [88].

3.1 Theory of positive matrices

Let $M = (M_{ij})_{i,j=1}^n$ be a matrix of order n with real coefficients. A matrix is said to be *nonnegative* if, for all $i, j = 1, \dots, n$, it holds that $M_{ij} \geq 0$, and is *positive* if this inequality is strict. To each matrix M there corresponds a digraph $D = D(M)$ of order n as follows: the vertex set is the set $V = \{1, \dots, n\}$, and there is an arc (i, j) from vertex i to vertex j if and only if $M_{ij} \neq 0$ (for all $i, j = 1, \dots, n$, we say that M is the *adjacency matrix* of digraph $D(M)$). In the case that M is nonnegative, we may think of $M_{ij} \geq 0$ as being the *multiplicity* $m(i, j)$, ie, the number of arcs of the form (i, j) . We will not work with *general digraphs*, which allow multiple arcs in a same pair of vertices, so we will put our attention in $(0, 1)$ -matrices. The link between matrices and digraphs permits to assert statements as the next lemma:

Lemma 3.1 ([13]) *Let M be the adjacency matrix of a digraph $D(M)$. There is a directed walk of length m from vertex i to vertex j if and only if the element in position (i, j) of M^m is positive.*

A matrix M of order n is called *reducible* if by simultaneous permutations of its lines it can be obtained a matrix of the form

$$\begin{pmatrix} M_1 & 0_{12} \\ M_{21} & M_2 \end{pmatrix}$$

where M_1 and M_2 are square matrices of order n_1, n_2 , respectively (where $n_1, n_2 \geq 1$), M_{21} is a matrix of size $n_2 \times n_1$, and 0_{12} is the null matrix (ie, each of its coefficients is equal to 0) of size $n_1 \times n_2$. If M is not reducible, then M is *irreducible*. Notice that a matrix of order 1 is irreducible. Irreducibility has a precise meaning if we again see the matrices as adjacency matrices of digraphs.

Theorem 3.2 ([13]) *Let M be a matrix of order n . Then M is irreducible if and only if its digraph $D(M)$ is strongly connected.*

Primitivity

Let D be a strongly connected digraph of order n . Let $k(D)$ the greatest common divisor of the lengths of the closed directed walks of D . If $n = 1$ and D does not contain a loop, $k(D)$ is undefined. The integer $k(D)$ is called the *index of imprimitivity* of D . The digraph is said to be *primitive* if $k(D) = 1$, and *imprimitive* if $k(D) > 1$. Notice that the length of a closed directed walk is the sum of the lengths of one or more directed cycles, hence the index of imprimitivity of D can also be defined as the greatest common divisor of the lengths of the directed cycles of D . Defined in this way, the integer $k(D)$ does not exceed the length of any directed cycle of D . Using the same relation as before, one can speak of primitive and imprimitive matrices. The following is a characterization and one of the most important properties of primitive matrices.

Theorem 3.3 ([13]) *Let M be a nonnegative matrix of order n . It holds that M is primitive if and only if there exists a positive integer N such that M^n is a positive matrix for each integer $n \geq N$.*

The next lemma can be found at [13], and describes a partition of the set of nodes of a strongly connected digraph D and several facts related to the index of imprimitivity $k(D)$.

Lemma 3.4 ([13]) *Let D be a strongly connected digraph of order n with index of imprimitivity equal to k . The following statements hold:*

- (i) *For each vertex a of D , k equals the greatest common divisor of the lengths of the closed directed walks containing a .*
- (ii) *For each pair of vertices a and b , the lengths of the directed walks from a to b are congruent modulo k .*
- (iii) *The set V of vertices of D can be partitioned into k nonempty sets V_1, V_2, \dots, V_k with $V_{k+1} = V_1$, where for each arc (a, b) of D there is some $i \in \{1, \dots, k\}$ such that $a \in V_i$ and $b \in V_{i+1}$.*
- (iv) *For $x_i \in V_i$ and $x_j \in V_j$, the length of a directed walk from x_i to x_j is congruent to $j - i$ modulo k , ($1 \leq i, j \leq k$).*

We refer to the vertex partition given by previous lemma as the *imprimitivity sets*. Let $a \in V(D)$ be any arbitrary fixed node of digraph D . We can define the i -th imprimitivity set as follows [13].

$$V_i = \{x_i \in V \mid \text{there is a directed walk from } a \text{ to } x_i \text{ with length } \equiv_{k(D)} i\} \quad (3.1)$$

The exponent

The smallest positive integer N that verifies the property of Theorem 3.3, that is, the smallest N such that M^n is a positive matrix for all $n \geq N$, is called the *exponent* of M , and is denoted by $\text{exp}(M)$. The exponent of M depends only on the digraph $D(M)$ (and not on the magnitude of the elements of matrix M), so in studying the exponent there is no loss in generality in considering only $(0, 1)$ -matrices. To conclude this introduction to positive matrices theory, we present the following theorem that states a quadratic upper bound for the exponent of a primitive matrix.

Theorem 3.5 ([45]) *Let M be a primitive $(0, 1)$ -matrix of order $n \geq 2$. Then*

$$\text{exp}(M) \leq (n - 1)^2 + 1$$

The last is a general bound for primitive matrices, but for some matrices it is possible to find tighter bounds, as the exponent can be evaluated in terms of other more basic quantities. Let $\text{exp}(M : i)$ be the smallest positive integer p such that all the elements in row i of M^p are nonzero ($1 \leq i \leq n$), or in other words, $\text{exp}(M : i)$ equals the smallest positive integer p such that there are directed walks of length p from vertex i to each vertex of $D(M)$. The following equality can be established [13].

Lemma 3.6 ([13]) $\text{exp}(M) = \max_{i \in \{1, \dots, n\}} \text{exp}(M : i)$

Lemma 3.6 is useful since it allows to obtain upper bounds for the exponent of a primitive matrix M . If l_1, \dots, l_n are n integers such that there are directed walks of length l_i from node i to every node of $D(M)$ ($i = 1, \dots, n$), then $\text{exp}(M) \leq \max\{l_1, \dots, l_n\}$. The previous lemma can be used to prove the following result.

Theorem 3.7 ([45]) *Let M be an irreducible matrix of order n having $p \geq 1$ nonzero elements on its main diagonal (or equivalently, p nodes with loop in its associated digraph $D(M)$). Then M is a primitive matrix and $\text{exp}(M) \leq 2n - p - 1$.*

Now, the next theorem makes use of Lemma 3.6 to find an upper bound for the exponent of matrices having a subdigraph that is complete. A complete digraph (V, A) of order d , is a directed graph where $|V| = d$, and each pair of different nodes i and j are connected by the arcs $(i, j), (j, i) \in A$. A complete digraph does not have nodes with loops.

Theorem 3.8 *Let M be an irreducible $(0, 1)$ -matrix of order $n \geq 3$ such that $D(M) = (V, A)$ is a (strongly connected) digraph having a complete subdigraph of order d with $d \geq 3$. Then M is primitive and $\text{exp}(M) \leq 2(n - d + 1)$.*

PROOF. Since $d \geq 3$, the complete subdigraph has at least a cycle of length 2 and other of length 3, then M is primitive. Let $W \subseteq V$ be the set of nodes of the complete subdigraph:

$|W| = d$, and let i and j be any two nodes of V . Since $D(M)$ is strongly connected, there is a directed path P_{i,k_1} from i to some $k_1 \in W$, without other node in W than k_1 , whose length is at most $n - d$ (simply find a path from i to any node in W , and take k_1 as the first node from W reached with this path). Similarly, there exists a directed path $P_{k_2,j}$ from some $k_2 \in W$ to j , without other node in W than k_2 , whose length is at most $n - d$. Now, since W induces a complete digraph in $D(M)$, there is a walk W_{k_1,k_2} from k_1 to k_2 of length at most 2 (when $k_1 = k_2$; in general, when $k_1 \neq k_2$, k_2 is reachable from k_1 in one step). Joining these three walks, P_{i,k_1} , W_{k_1,k_2} and $P_{k_2,j}$, gives us a directed walk from i to j of length at most equal to $2(n - d + 1)$. If we had that the sum of the two path lengths is equal to $2(n - d) - r$, where $r = 0, \dots, 2(n - d)$, since any node in W is reachable from any node in W with walks of length greater or equal than 2, we can take a walk W_{k_1,k_2} of length $r + 2$ (whatever nodes $k_1, k_2 \in W$) and obtain a directed walk from i to j whose length is exactly $2(n - d + 1)$. \square

3.2 Relation with disjunctive networks

In the context of this work, we are going to restrict to $(0, 1)$ -matrices when representing adjacency matrices of digraphs. Then these matrices belong to $\mathcal{M}(\{0, 1\})_{n \times n}$, for $n \in \mathbb{N}$, space equipped with the logical sum (or the usual disjunction \vee) and the logical product (or the usual conjunction \wedge). It was already said that the main results from the theory of positive matrices remain valid with this restriction, and there is no loss of generality for our purposes.

Example 7 *In the figure below it can be appreciated a digraph D with its respective adjacency matrix $M(D)$:*



If we get the logical product of $M(D)$ with itself, $M(D)^2$, by Lemma 3.1 this matrix corresponds to the digraph whose arcs represent the walks of length 2 in D . We call this digraph as D^2 .



Note 1 *The relation between disjunctive Boolean networks and the theory of positive matrices is established through the adjacency matrix, and it will be clearer with the following observation. Disjunctive Boolean networks verify the following property ([37]): if $M(D)$ is the adjacency matrix of the disjunctive network $D = (V, A)$ where $|V| = n$, ie, the $n \times n$ $(0, 1)$ -matrix meeting that $M(D)_{ij} = 1$ iff $(i, j) \in A$, then if D is updated with parallel schedule, it holds that for all $\vec{x}(t) \in \{0, 1\}^n$, and for all $k \in \mathbb{N}$, $\vec{x}(t+k) = \vec{x}(t) \cdot (M(D))^k$. It is not difficult to see the veracity of last property: for all $i \in \{1, \dots, n\}$, $x(t+1)_i = 1$ if and only if there exists some $k \in V$ such that $x(t)_k = 1$ and the arc $(k, i) \in A$, since local transition functions are OR functions. The latter is equivalent to the existence of some $k \in V$ such that $x(t)_k = M(D)_{ki} = 1$, and this is equivalent to the fact that the sum $\sum_{k \in V} x(t)_k \cdot M(D)_{ki} = (\vec{x}(t) \cdot M(D))_i$ is equal to 1.*

With this observation, we can state the main theorem of this section, which claims a polynomial bound of the transient length of a disjunctive network:

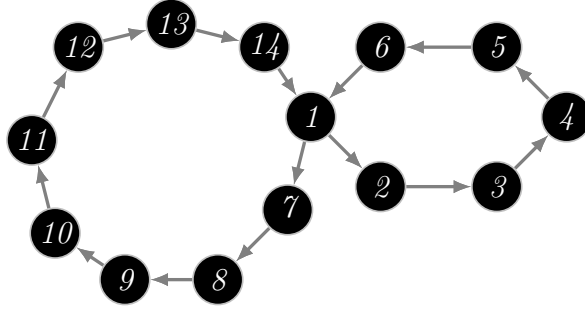
Theorem 3.9 *Let N_F be a Boolean disjunctive network of size n , with global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, and S be a block-sequential update schedule for N_F , such that the digraph associated to F_S , the global transition function with schedule S , is weakly connected. Then, from any initial condition, in at most $\mathcal{O}(n^2)$ updates of N_F according to S , the state vector gets to an attractor, which can be a fixed point or a limit cycle.*

The hypothesis of being weakly connected is not restrictive, since the network dynamics of different weakly connected components on any given digraph are completely independent (there is no incidence between them), then these different components can be studied separately. Notice that in the particular case when S is the parallel schedule, and the digraph associated with N_F (whose transition function is F) is strongly connected and primitive, the result is direct from Note 1 and Theorem 3.5; if the initial condition is any non-null vector, since after a quadratic number of updates the matrix of adjacency of the network obtained is positive, the state vector results to be positive and it stays positive in ulterior iterations, which is a fixed point. Indeed we can restrict the analysis to the parallel schedule thanks to the fact that updates with S are simulatable in parallel with the transition function relative to schedule S , F_S .

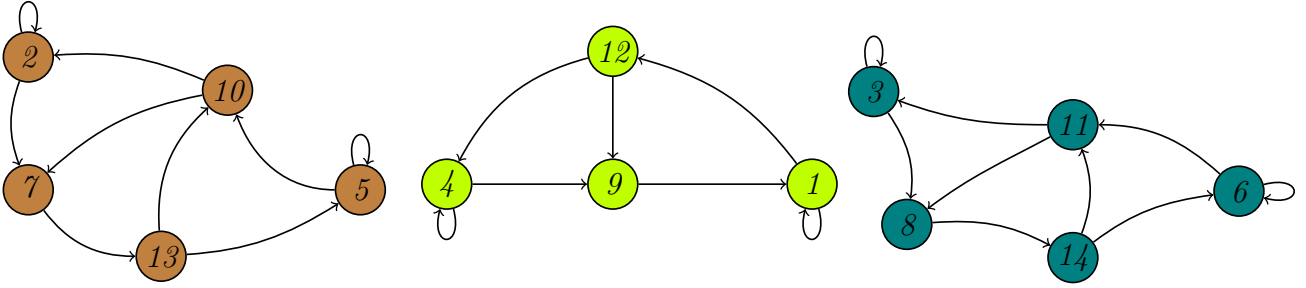
By the above, the necessary work to prove Theorem 3.9 is to study the case when the digraph associated is neither primitive nor strongly connected. In fact, it is possible to say something when the adjacency matrix of the network is irreducible and imprimitive. For this purpose, the following observation will be useful.

Lemma 3.10 *Let $M = M(D)$ be an imprimitive irreducible matrix (or in other words, the digraph $D = (V, A)$ associated to M is strongly connected, where the index of imprimitivity of D , $k(D)$, verifies that $k(D) > 1$), and let ℓ be the length of the shortest circuit in D . Then the strongly connected components of the digraph associated with the matrix M^ℓ are primitive. In fact, these strongly connected components correspond exactly to the sets $(V_i)_{i=1}^{k(D)}$ from Lemma 3.4.*

Example 8 *Let us see an example of what is stated in the last lemma. Let D be the digraph below:*



It can be seen that D has a circuit of length equal to 6, and another of length 9, then $k(D) = 3$. The length of the shortest circuit is 6. On the other hand, if M is the adjacency matrix of D , then the digraph associated to M^6 , which we will call as D^6 (by Lemma 3.1, this digraph has an arc (i, j) if and only if there is a walk in D of length 6 that starts on i and ends on j), has 3 s.c.c.:



The lemma guarantees that these 3 sets of nodes correspond to the equivalence classes induced by the relation between two nodes given by the existence of a walk of length congruent to zero modulo $k(D) = 3$, ie, these 3 sets correspond to the sets V_i ($i = 1, 2, 3$) as described in Lemma 3.4. This is indeed what happens in this example.

Note 2 Observe that, using the same assumptions as in Lemma 3.10, it holds that, in the digraph associated with matrix M^ℓ , which we will refer to as D^ℓ , there are no arcs connecting nodes in different strongly connected components: suppose there is an arc starting on $x_i \in V_i$ (V_i is a strongly connected component of D^ℓ by Lemma 3.10), and ending on $x_j \in V_j$, with $1 \leq j \neq i \leq k(D)$ (V_j is another s.c.c. of D^ℓ different than V_i , by Lemma 3.10). The above implies that there exists a walk in D connecting x_i and x_j of length ℓ , call this walk as W_{x_i, x_j} . Since ℓ is multiple of $k(D)$, $|W_{x_i, x_j}| \equiv_{k(D)} 0$, but this contradicts (iv) from Lemma 3.4, which states that $|W_{x_i, x_j}| \equiv_{k(D)} j - i \neq 0$.

Note 3 Observe that, using the same assumptions as in Lemma 3.10, and referring to the shortest circuit of D as C_ℓ , it holds that $|C_\ell \cap V_i| = \frac{\ell}{k(D)}$ for $i = 1, \dots, k(D)$. Observe this quotient is, by definition of $k(D)$, a positive integer. Notice that as every arc in D starts from a node in V_i and ends on a node in V_{i+1} , for some i , $1 \leq i \leq k(D)$ (condition (iii) in Lemma 3.4), and as there are $k(D)$ sets V_i and ℓ nodes in C_ℓ , where $k(D) \leq \ell$, necessarily for each $i \in \{1, \dots, k(D)\}$ there is at least one node in $C_\ell \cap V_i$. A simple method to enumerate the set of nodes in $C_\ell \cap V_i$ is to take any node in this set (recall we already saw this set is not null), and moving within circuit C_ℓ by thrusting forward $k(D)$ nodes each time. Since in C_ℓ there are ℓ different nodes, with this procedure we will find $\frac{\ell}{k(D)}$ different nodes, which belong

to $C_\ell \cap V_i$ by (iii) in Lemma 3.4. By (iv) in Lemma 3.4, every node in $C_\ell \cap V_i$ is found with this procedure and the claim holds.

As said before, the last observation allows to analyze the subcase of Theorem 3.9 when the digraph of network is strongly connected and imprimitive, which leads us to next proposition.

Proposition 3.11 *Let $M = M(D)$ be an irreducible imprimitive $(0, 1)$ -matrix of order n ($D = (V, A)$ is a strongly connected digraph, where $k(D) > 1$), ℓ the length of the shortest circuit in D , and $\vec{x}(0) \in \{0, 1\}^n$ any configuration vector. If we consider the linear system*

$$\vec{x}(h) = \vec{x}(0) \cdot M^h, \quad h \in \mathbb{N}$$

then, there exist $t \in \mathcal{O}(n)$, $p \in \mathcal{O}(\ell)$, such that $\vec{x}(t+p) = \vec{x}(t)$. Also it holds that $p = 1$ if and only if for every set V_i , there exists $j_i \in V_i$ such that $x(0)_{j_i} = 1$, where $(V_i)_{i=1}^{k(D)}$ is the vertex partition given by Lemma 3.4.

PROOF. Assume the vector $\vec{x}(0)$ is not null (otherwise the statement holds trivially) and consider the following system.

$$\begin{aligned} \vec{y}(h) &= \vec{y}(0) \cdot B^h, \quad h \in \mathbb{N} \\ \vec{y}(0) &= \vec{x}(0) \end{aligned} \tag{3.2}$$

where $\vec{y}(h) \in \{0, 1\}^n$, for all $h \in \mathbb{N}$, and $B = M^\ell$. As before, we refer to the digraph associated with matrix B as D^ℓ (D^ℓ is the digraph resulting in considering the walks of length ℓ in D). Consider the partition $(V_i)_{i=1}^{k(D)}$ of the set of nodes V given by Lemma 3.4. We are able to study the system (3.2) restricting the analysis to each of the strongly connected components of D^ℓ (since the behavior of these subsystems are independent, by Note 2), which correspond by Lemma 3.10 to the sets $(V_i)_{i=1}^{k(D)}$. Therefore consider the i -th subsystem associated to the set V_i ($i = 1, \dots, k(D)$):

$$\begin{aligned} \vec{y}^i(h) &= \vec{y}^i(0) \cdot B_i^h, \quad h \in \mathbb{N} \\ \vec{y}^i(0) &= \vec{x}^i(0) \end{aligned}$$

where $B_i \in \mathcal{M}_{|V_i| \times |V_i|}(\{0, 1\})$ is the matrix formed by the rows and columns of B corresponding to nodes belonging to the set V_i , and $\vec{y}^i(0), \vec{x}^i(0) \in \{0, 1\}^{|V_i|}$ are the vectors resulting in restricting $\vec{y}(0), \vec{x}(0)$ respectively to the vector components associated to nodes belonging to V_i . Since for $i = 1, \dots, k(D)$, V_i induces in D^ℓ a primitive subdigraph (Lemma 3.10), each matrix B_i is primitive and then there exist $\text{exp}(B_i) \in \mathbb{N}$ for $i = 1, \dots, k(D)$. We can obtain now an estimation of these magnitudes. We will refer to the shortest circuit in D as C_ℓ . By realizing that every node in C_ℓ has a loop in digraph D^ℓ , thanks to Note 3 we conclude that each digraph $D^\ell(V_i)$ has at least $\frac{\ell}{k(D)}$ nodes with loop, then by Theorem 3.7 it holds that $\text{exp}(B_i) \leq 2 \cdot |V_i| - \frac{\ell}{k(D)} - 1$. This bound implies that

$$B_i^h \text{ is positive for all } h \geq \max_{1 \leq j \leq k(D)} 2 \cdot |V_j| - \frac{\ell}{k(D)} - 1, \text{ for all } i = 1, \dots, k(D) \tag{3.3}$$

We prove first the left direction of the equivalence. We assume that $\vec{x}(0)$ has at least one positive component for every set V_i , which implies by previous definitions that for each $i = 1, \dots, k(D)$, the vector $\vec{y}^i(0)$ is not null. By (3.3), we obtain that each vector $\vec{y}^i(h)$ is positive for all $h \geq 2 \cdot n - \frac{\ell}{k(D)} - 1$ (since for all $i = 1, \dots, k(D)$, $|V_i| \leq n$), and from this, that

$$\vec{y}^i(h) \text{ is positive for all } h \geq 2 \cdot n - \frac{\ell}{k(D)} - 1$$

From the last assertion and definition of system (3.2), we deduce that $\vec{x}(m)$ is positive for $m = \ell \cdot [2 \cdot n - \frac{\ell}{k(D)} - 1]$. But this implies directly that $\vec{x}(m+1) = \vec{x}(m) \cdot M$ is positive (if not, there is in M a zero column, say row j , meaning that node j has out-degree equal to zero, but this is not possible since D is strongly connected), and inductively that

$$\vec{x}(m) \text{ is positive for all } m \geq \ell \cdot [2 \cdot n - \frac{\ell}{k(D)} - 1]$$

which shows that there exist $t \in \mathcal{O}(n)$, $p = 1$, such that $\vec{x}(t+p) = \vec{x}(t)$.

To prove the converse, suppose now there are $1 \leq r < k(D)$ sets $(V_i)_{i \in R}$, where $|R| = r$, such that $\vec{x}^i(0)$ is null for every $i \in R$ and $\vec{x}^i(0)$ is not null for every $i \in \{1, \dots, k(D)\} \setminus R$. Again by (3.3) and previous definitions, it is obtained that

$$\begin{aligned} \vec{y}^i(h) \text{ is positive, for all } h \geq 2 \cdot n - \frac{\ell}{k(D)} - 1, \text{ for all } i \in \{1, \dots, k(D)\} \setminus R \\ \vec{y}^i(h) \text{ is null, for all } h \in \mathbb{N}, \text{ for all } i \in R \end{aligned}$$

which means -by previous definitions- that

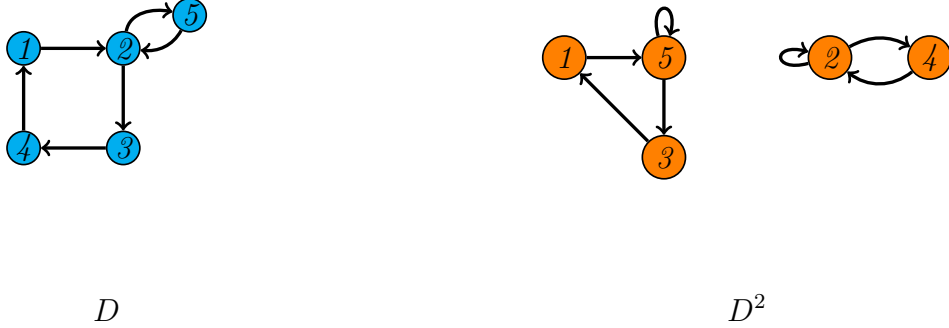
$$\vec{x}((t+1) \cdot \ell) = \vec{x}(t \cdot \ell) \quad \text{for } t = 2 \cdot n - \frac{\ell}{k(D)} - 1$$

ie, in at most $t \in \mathcal{O}(n)$ iterations the linear system gets to a state vector verifying what is stated by the lemma, where $p \leq \ell$. Suppose that $p = 1$, then $\vec{x}(t \cdot \ell + 1) = \vec{x}(t \cdot \ell)$. According to the values that $\vec{y}^i(h)$ takes, $\vec{x}^i(t \cdot \ell)$ is null for $i \in R$ and $\vec{x}^i(t \cdot \ell)$ is positive for $i \notin R$, so if $\vec{x}(t \cdot \ell + 1) = \vec{x}(t \cdot \ell) \cdot M$ is equal to $\vec{x}(t \cdot \ell)$, this means that

$$(\vec{x}(t \cdot \ell) \cdot M)_w = 0, \quad \forall w \in V_i, \quad \forall i \in R$$

By the above necessarily, $M_{b,w} = 0$ for all $w \in V_i, i \in R, b \in V_q$ and $q \notin R$. On the other hand, as $1 \leq r < k(D)$ necessarily there is $g \in \{1, \dots, k(D)\}$ such that $g \notin R$ and $g+1 \in R$ (g could be $k(D)$ and in this case we take $g+1$ as 1), and by (iii) in Lemma 3.4, there are $b \in V_g$ and $w \in V_{g+1}$ such that $(b, w) \in A$, which is absurd. This shows that $p > 1$, which concludes the proof. \square

Example 9 Let $D = (V, A)$ be the following disjunctive Boolean network of size 5, which is strongly connected, and imprimitive since $k(D) = 2$. The length of the shortest circuit is $\ell = 2$. We can also see the digraph D^2 formed by the walks of length 2 in D , where each strongly connected component of D^2 has $\frac{\ell}{k(D)} = \frac{2}{2} = 1$ node with loop.



If we update network D in parallel, the transient and period lengths for every initial condition have a linear size by Proposition 3.11. Indeed, from the proof (see (3.3)) of this result we know the longest transient length is upper bounded by

$$\ell \cdot \left(\max_{1 \leq j \leq k(D)} 2 \cdot |V_j| - \frac{\ell}{k(D)} - 1 \right)$$

where V_j correspond to the strongly connected components of D^ℓ (this bound arise from the bound of the exponent of these components). In our example, $|V_1| = 3$ and $|V_2| = 2$, therefore the last bound is equal to $2 \cdot \max\{4, 2\} = 8$. The longest transient length for network D updated in parallel is equal to 5, achieved with the initial condition $(1, 0, 0, 0, 0)$ that gets to a limit cycle of length 2, and the initial conditions $(1, 0, 0, 1, 0)$, $(1, 1, 0, 1, 0)$ and $(1, 1, 0, 0, 0)$ that get to a fixed point.

At this point we have all the ingredients to prove the Theorem 3.9:

PROOF OF THEOREM 3.9. Let F be the transition function of N_F and $\vec{x}(0) \in \{0, 1\}^n$ be a non-null state vector. Without loss of generality we can assume the schedule $S = \pi$ is the parallel schedule, since a complete update sequence with any block-sequential schedule S is simulatable with one update of function F_S in parallel, where F_S is also a disjunctive network (thanks to the fact F is disjunctive and the definition of F_S). Now, thanks to Note 1, studying the trajectory $\vec{x}(t)$ of network states is equivalent to study the linear system

$$\vec{x}(h) = \vec{x}(0) \cdot M^h, \quad h \in \mathbb{N}$$

where $M \in \mathcal{M}_{n \times n}(\{0, 1\})$ is the adjacency matrix of the digraph D associated to F_S . The hypothesis that D is weakly connected leaves us two main cases to analyze: D is strongly connected or the condensation digraph of D is a weakly connected DAG. If D is a trivial strongly connected component, that is, it is formed by just one node without loop, $n = 1$ and $M_{1,1} = 0$ and the statement holds trivially. If D is a non-trivial s.c.c., it is either primitive or imprimitive. In the first case, M is irreducible and primitive, then by Theorem 3.5 the statement of the theorem holds, and the attractor is a fixed point consisting in a positive

vector. In the second case, Proposition 3.11 gives the (linear, in particular quadratic) rate of convergence, and the attractor can be a fixed point or a limit cycle.

Assume now the other case, in which there is a weakly connected DAG (the condensation digraph D^*) where each node is a s.c.c. from D . This DAG admits a topological sort; consider first a non-trivial s.c.c. C' such that $\vec{x}(0)$ restricted to C' is non-null and there is no other non-trivial s.c.c. \bar{C} verifying this property, that reaches C' by some walk (necessarily \bar{C} comes first than C' according to the sort), that is, $\vec{x}(0)$ restricted to \bar{C} is null. From Lemma 3.1 we deduce that

$$x(h)_i = 1 \text{ iff there is a node } j \in D \text{ such that } x(0)_j = 1 \text{ and there is in } D \text{ a walk of length } h \text{ from } j \text{ to } i \quad (3.4)$$

By the above, if we let the system evolve, for every s.c.c. \bar{C} that reaches C' by some walk, it holds that, for all $h \in \mathbb{N}$, $\vec{x}(h)$ restricted to \bar{C} is null. And by Theorem 3.5 and Proposition 3.11 in at most $h \in \mathcal{O}(|C'|^2)$, $\vec{x}(h)$ restricted to C' converges to some attractor of period $1 \leq p' \leq \ell$, where ℓ is the length of the shortest circuit of C' (Proposition 3.11). Whether there exists some non-trivial s.c.c. C reachable from C' by a walk (for simplicity, without other non-trivial s.c.c.'s), it holds by (3.4) that a number 1 (a positive component of the state vector) is diffused from C' to C after a number of iterations linear on the length of the walk, with a certain periodicity at most p' . So we will focus now the analysis on component C . We assume, for simplicity, that component C receives a forced input on vertex $v \in C$ from time $h = 0$ onwards of periodicity $p \geq 1$, where by the above $p \in \mathcal{O}(n)$. In other words

$$x(h)_v = 1 \quad \text{for all } h = z \cdot p, \quad \text{where } z = 0, 1, 2, \dots \quad (3.5)$$

Fix the positive integer f_1 . The property (3.5) holds for any h equal to some multiple of p , in particular for $h_r = (f_1 - r) \cdot p$ where $0 \leq r \leq f_1$. Let $u \in V(D)$ be a vertex such that there exists a walk (in D) of length $r \cdot p$ from v to u . Then, by the equivalence (3.4) we have that

$$\begin{aligned} \forall u \in V(D) \text{ s.t. there exists a walk (in } D) \text{ of length } r \cdot p \text{ from } v \text{ to } u, \text{ where } 0 \leq r \leq f_1, \\ \text{then } x(h_r + r \cdot p)_u = x(f_1 \cdot p)_u = 1 \end{aligned} \quad (3.6)$$

Last property says us that in the state vectors obtained at times multiple of p , there are positive components for vertices at distance from v equal to every lesser multiple of p . This fact determines how the positive components are diffused in the s.c.c. C . Let $k(C)$ be the index of imprimitivity of C , and $(V_i)_{i=1}^{k(C)}$ the imprimitivity sets (of C , Lemma 3.4). Which of these sets have positive components in vectors $\vec{x}(f_1 \cdot p)$ with this setting? Recalling the

definition (3.1), where we take v as the arbitrary fixed node, the sets V_i that have positive components are those where $i \in \{1, \dots, k(C)\}$ and the walks W_{v, x_i} from v to nodes $x_i \in V_i$ meet the following

$$|W_{v, x_i}| \equiv_p 0 \equiv_{k(C)} i \implies i = x \cdot p + y \cdot k(C), (x, y \in \mathbb{Z})$$

By Lemma A.2, necessarily it holds that i is a multiple of $\gcd(p, k(C))$, and then the number of distinct imprimitivity sets having positive components in vectors $\vec{x}(f_1 \cdot p)$, for f_1 sufficiently large, is $k(C)/\gcd(p, k(C))$. How large f_1 needs to be? We claim that after $f_1 \cdot p$ updates, where $f_1 = k(C)/\gcd(p, k(C))$, no new imprimitivity sets are reached. Indeed, let $f_2 \geq f_1$ be a positive integer, we can write it (by euclidean division) as

$$f_2 = q \cdot f_1 + r, \text{ where } q \in \mathbb{N}, 0 \leq r < f_1$$

With this equality, we can write the following

$$\begin{aligned} f_2 \cdot p &= q \cdot f_1 \cdot p + r \cdot p = \frac{q \cdot k(C) \cdot p}{\gcd(p, k(C))} + r \cdot p \\ &= q \cdot \text{lcm}(p, k(C)) + r \cdot p \quad (\text{Theorem 52 [40]}) \\ &\equiv_{k(C)} r \cdot p \end{aligned}$$

Which shows, by definition (3.1) of imprimitivity sets and (3.6), that in state vectors obtained at times $h = f_2 \cdot p$ multiple of p with $f_2 \geq f_1$, no new imprimitivity sets are reached by positive components. Assume for now that $\gcd(p, k(C)) > 1$ (then $p, k(C) > 1$). Therefore, if we let the linear system evolve (without considering the forced input) with initial condition given by $\vec{x}(f_1 \cdot p)$, by Proposition 3.11 in at most $\mathcal{O}(|C|)$ iterations from the alluded initial condition the state vector converges to an attractor consisting in a limit cycle. In the particular case that $\gcd(p, k(C)) = 1$ and $k(C) > 1$, the number of distinct imprimitivity sets reachable by positive components is $k(C)/1 = k(C)$, ie, all the imprimitivity sets are reachable with the forced input, and by Proposition 3.11 in at most $\mathcal{O}(|C|)$ iterations from the alluded initial condition the state vector converges to a fixed point. When $k(C) = 1$, by Theorem 3.5 in at most $\mathcal{O}(|C|^2)$ iterations from any initial condition (particularly if there is a forced input) the state vector converges to a fixed point. When $p = 1$, (3.6) allows to say that in at most $\mathcal{O}(|C|)$ the state vector converges to a fixed point. The bottleneck in the general case is given by the number of iterations $f_1 \cdot p$ necessary to reach all the distinct imprimitivity sets reachable by the forced input; this number can be bounded as follows

$$f_1 \cdot p = \frac{k(C) \cdot p}{\gcd(p, k(C))} = \text{lcm}(p, k(C)) \leq p \cdot k(C) \in \mathcal{O}(n^2)$$

The previous analysis is valid for only one forced input. Assume now that component C receives, from time $h = 0$ onwards, a forced input on vertex $v_1 \in C$ of periodicity $p_1 \geq 1$, and another forced input on vertex $v_2 \in C$ with periodicity $p_2 \geq 1$, where $p_1, p_2 \in \mathcal{O}(n)$. The effect on dynamics of component C due to the presence of a forced input on vertex v_i with periodicity p_i , can be described as an initial condition $\vec{y}^{v_i, p_i} \in \{0, 1\}^{|C|}$, where the vector component associated to vertex v_i is positive, and equal to zero otherwise. This initial condition is added to the state vector each p_i time steps, starting from $h = 0$, and diffuses

according to the network connectivity in successive iterations, that is, for $h \geq 0$, if we refer to $\vec{x}^{v_i, p_i}(h) \in \{0, 1\}^{|C|}$ as the contribution of the i -th forced input, this vector corresponds to

$$\begin{aligned}\vec{x}^{v_i, p_i}(h) &= \vec{y}^{v_i, p_i} \cdot M(C)^h + \vec{y}^{v_i, p_i} \cdot M(C)^{h-p_i} + \dots \\ &= \vec{y}^{v_i, p_i} \cdot \sum_{j=0}^{\xi_i} M(C)^{h-j \cdot p_i}\end{aligned}\tag{3.7}$$

where ξ_i is such that $h - \xi_i \cdot p_i < p_i$, that is, ξ_i is the quotient of the division of h by p_i , and $M(C)$ is the adjacency matrix of component C . Thus, we can see the dynamics of the two forced inputs as a linear superposition of the contributions of each forced input, that is, for $h \geq 0$,

$$\vec{x}_C(h) = \vec{x}^{v_1, p_1}(h) + \vec{x}^{v_2, p_2}(h)\tag{3.8}$$

where $\vec{x}_C(h) \in \{0, 1\}^{|C|}$ is the state vector $\vec{x}(h)$ restricted to the strongly connected component C . By the above, if we call t_i to the number of iterations, starting from $h = 0$, necessary for the i -th forced input to reach an attractor, it holds that after $t_{max} = \max\{t_1, t_2\}$, each vector $\vec{x}^{v_i, p_i}(t_{max})$ has reached its attractor, and then $\vec{x}_C(t_{max})$ has reached an attractor state configuration. Since each $t_i \in \mathcal{O}(n^2)$ (by the analysis for only one forced input), and we have a finite amount of forced inputs, it holds that $t_{max} \in \mathcal{O}(n^2)$. It is apparent that this analysis remains valid for any finite number of forced inputs (not only two).

Now, in the general case, a node $v \in C$ receives, with some periodicity $1 \leq p \in \mathcal{O}(n)$, a $(0, 1)$ -word $w = c_1 c_2 \dots c_p$, where each $c_i \in \{0, 1\}$ and $p = |w|$. The characters c_i of w that are equal to 1 correspond to different forced inputs on vertex v with periodicity p starting at different time steps, and all these forced inputs associated to word w share the same bound $t_w \in \mathcal{O}(n^2)$ of number of iterations needed to reach an attractor, then in at most $\mathcal{O}(t_w + p) = \mathcal{O}(n^2)$ iterations, counted from the first time this word is received by vertex v , all these forced inputs have reached an attractor state configuration, in the component C .

Finally, in order to get an estimation of the upper bound of iterations necessary, starting from $h = 0$, to reach a global attractor for network D (global in the sense that is not restricted to some s.c.c., but for all the network), we can think in a network whose condensation digraph consists of only a path connecting the different strongly connected components, one after another (maybe separated by some trivial s.c.c.). This is the worst case in terms of the number of iterations, because when the system evolves, it does not happen that the dynamics of some s.c.c. evolves in parallel to the dynamics of some other s.c.c.: at the beginning, the first non-trivial s.c.c. having positive components (first according to the topological sort of the condensation digraph) gets to some attractor of length p , then it sends a word of length p , with periodicity p , to the next non-trivial s.c.c., this component gets to some attractor, and so on until the last non-trivial component. Summing all the iterations, both the necessary to

reach the non-trivial s.c.c. attractors and for sending the words from one non-trivial s.c.c. to the next, it gives a bound of $\mathcal{O}(NC \cdot n^2 + n)$, where NC is the number of non-trivial strongly connected components of digraph D .

□

If we argue similarly to the last theorem, we can claim a bound linear on the size of the network when the associated digraph has some of the substructures seen in the subsection regarding theory of positive matrices.

Theorem 3.12 *Let N_F be a Boolean disjunctive network of size n , with global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, and S be a block-sequential update schedule for N_F , such that the digraph associated to F_S is weakly connected, and every strongly connected component has a loop or a complete subdigraph of order greater or equal than 3. Then, from any initial condition, in at most $\mathcal{O}(n)$ updates of N_F according to S , the state vector gets to an attractor, which results to be a fixed point.*

PROOF. Let F be the transition function of N_F and $\vec{x}(0) \in \{0, 1\}^n$ be a non-null state vector. Without loss of generality we can assume the schedule $S = \pi$ is the parallel schedule, since a complete update sequence with any block-sequential schedule S is simulatable with one update of function F_S in parallel, where F_S is also a disjunctive network (thanks to the fact F is disjunctive and the definition of F_S). Now, thanks to Note 1, studying the trajectory $\vec{x}(t)$ of network states is equivalent to study the linear system

$$\vec{x}(h) = \vec{x}(0) \cdot M^h, \quad h \in \mathbb{N}$$

where $M \in \mathcal{M}_{n \times n}(\{0, 1\})$ is the adjacency matrix of the digraph D associated to F_S . The hypothesis that D is weakly connected leaves us two main cases to analyze: D is strongly connected or the condensation digraph of D is a weakly connected DAG. If D is a trivial strongly connected component, that is, it is formed by just one node without loop, $n = 1$ and $M_{1,1} = 0$ and the statement holds trivially. Since every non-trivial s.c.c. of D has a loop or a complete subgraph of order $o \geq 3$, every non-trivial s.c.c. of D is primitive. Thus, if D is a non-trivial strongly connected component, M is irreducible and primitive, then by Theorem 3.5 the attractor is a fixed point consisting in a positive vector. By the hypothesis, Theorems 3.7 and 3.8 give the linear rate of convergence.

Assume now the other case, in which there is a weakly connected DAG (the condensation digraph D^*) where each node is a s.c.c. from D . This DAG admits a topological sort; consider first a non-trivial s.c.c. C' such that $\vec{x}(0)$ restricted to C' is non-null and there is no other non-trivial s.c.c. \bar{C} verifying this property, that reaches C' by some walk (necessarily \bar{C} comes first than C' according to the sort), that is, $\vec{x}(0)$ restricted to \bar{C} is null. From Lemma 3.1 we deduce that

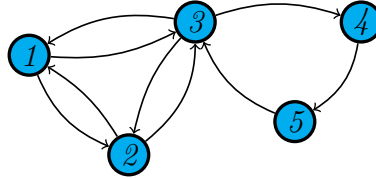
$$x(h)_i = 1 \text{ iff there is a node } j \in D \text{ such that } x(0)_j = 1 \text{ and there is in } D \text{ a walk of length } h \text{ from } j \text{ to } i \quad (3.9)$$

By the above, if we let the system evolve, for every s.c.c. \bar{C} that reaches C' by some walk, it holds that, for all $h \in \mathbb{N}$, $\vec{x}(h)$ restricted to \bar{C} is null. And by Theorems 3.5, 3.7 and 3.8 in at most $h' \in \mathcal{O}(|C'|)$, $\vec{x}(h')$ restricted to C' converges to an attractor consisting in a fixed point: $\vec{x}(h')_k = 1$, for every vertex k belonging to C' . Since C' is strongly connected, the last property is kept in ulterior executions (ie, the attractor is a fixed point). In particular, for those vertices $k' \in C'$ from which there are walks starting from C' and carrying to other strongly connected components, it holds that $\vec{x}(h)_{k'} = 1$, for every $h \geq h'$.

Now, for every node $v \in V = V(D)$ reachable from C' , there is a walk from C' to v of length $n_v \leq |V|$, which implies by the equivalence (3.9) that $\vec{x}(n_v + h)_v = 1$, for every $h \geq h'$. Finally, it holds that, after a number $\bar{n} = h' + \max_{v \in V} n_v$, which is at most $\mathcal{O}(n)$, it holds that $\vec{x}(\bar{n})_v = 1$, for every $v \in V$ reachable from C' in D . Since this is valid for every non-trivial component C verifying the same as C' , the attractor consists in a fixed point.

□

Example 10 *Let us see an example of what is stated in the last result. Let N_F be a disjunctive Boolean network of size 5, with global transition function F and local transition functions $f_1(\vec{x}) = x_2 \vee x_3$, $f_2(\vec{x}) = x_1 \vee x_3$, $f_3(\vec{x}) = x_1 \vee x_2 \vee x_5$, $f_4(\vec{x}) = x_3$ and $f_5(\vec{x}) = x_4$. This network has the next associated digraph D , depicted below.*



It can be seen that this digraph has a complete subdigraph of order $o = 3$ constituted by nodes 1, 2 and 3. The longest transient length when this network is updated in parallel, is for the initial condition $(0, 0, 0, 1, 0)$, which gets to the fixed point $(1, 1, 1, 1, 1)$ after 6 updates. This transient length matches (in the case of this specific digraph D) to the upper bound of the exponent given by Theorem 3.8, for digraphs containing a complete subdigraph of order greater or equal than 3: $6 = 2 \cdot (5 - o + 1) = 2 \cdot (5 - 3 + 1)$, which in turn is an application of Theorem 3.12.

Chapter 4

Filter convergence

The problem addressed in this section is the estimation of the time complexity of a filter applied onto a disjunctive Boolean network, and a block-sequential update schedule, to converge to its filter attractor. This goal requires the estimation of time required by every application of Gauss-Seidel operator, matter treated in Subsection 4.1, and the number of applications of Gauss-Seidel necessary to reach the filter attractor. This latter issue is analyzed at Subsection 4.2, where the question is restricted to the case of schedules with two blocks. The solution presented establishes that the evolution of the filter, that is, the different networks produced in the process, can be described as the parallel dynamics of some disjunctive network, which allows to deduce a quadratic estimation of the number of applications of Gauss-Seidel, as a consequence of Theorem 3.9. Finally, in Subsection 4.3 it is discussed the overall cost of applying a filter in this setting.

4.1 Computing the Gauss-Seidel operator

The following pseudo-code is the formalization of the algorithm that returns the network $D^S = (V, A^S)$ calculated by the Gauss-Seidel operator associated to a certain block-sequential update schedule S (received as input) applied onto an input network represented as a digraph $D = (V, A)$. The intuition behind what makes this procedure is, when there is an arc $(u, j) \in A$ where the node u is updated before the node j , or in other words, the local j -th function depends on the u -th states vector component which has already been updated when j -th function is updated, then, in the expression defining the j -th function the u -th variable is substituted by the expression defining the u -th local function. Thereby, the j -th function, previously depending on variable u , now in A^S depends on the variables which u -th function depended in A . This makes the parallel simulation of j -th function in network D^S to incorporate dependencies acquired by the fact that, according to schedule S , there are certain variables on which the j -th function depends that are updated before this function. All remaining arcs in D , in particular those connecting nodes updated synchronously or in parallel -nodes belonging to the same block of synchronous updating-, are preserved and added to the network D^S , which is the network finally returned by the algorithm. Thus, the

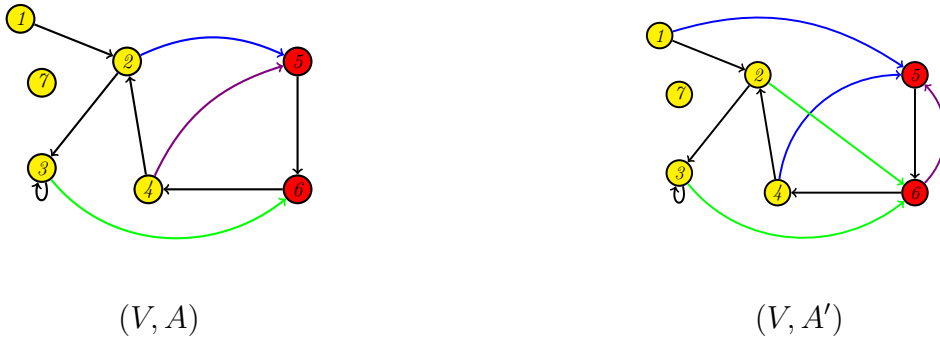
following algorithm provides a procedural characterization of Gauss-Seidel operator.

```

GAUSS-SEIDEL( $D = (V, A), S$ )
1   $A' = \phi, (A' \subseteq A)$  // initialization
2   $I = \phi, (I \subseteq V)$ 
   Let  $B_0, \dots, B_{m-1}$  be the blocks of  $S$ , and let  $s$  be the update function
3  for  $i = 0$  to  $m - 1$ 
4      for  $j \in B_i$ 
5          for  $(u, j) \in A$  // we go over every arc incident to node  $j$ 
6              if  $s(u) \geq s(j)$ 
7                   $A' = A' \cup \{(u, j)\}$ 
8              else
9                  for  $(t, u) \in A'$ 
10                      $A' = A' \cup \{(t, j)\}$ 
11          $I = I \cup \{j\}$ 
12  return  $(I, A')$ 

```

Example 11 In the figure below it can be seen the application of GAUSS-SEIDEL algorithm onto a network $D = (V, A)$ and schedule S of two blocks B_0 and B_1 . The initial network is (V, A) , and (V, A') is the network returned by the algorithm applied onto the initial network and the schedule S . In both networks, the nodes belonging to the first block B_0 are yellow (nodes 1, 2, 3, 4, 7), and those in the second block B_1 are red (nodes 5, 6):



The black arcs $((1, 2), (2, 3), (6, 4), (3, 3)$ and $(4, 3))$ are those that are preserved by the algorithm. The other arcs are modified by the algorithm. The respective colors show how different arcs are modified, moving the dependencies to all prospective parent nodes. For example, the purple arc $(4, 5)$ becomes an internal arc $(6, 5)$ from the second block in (V, A') (and it is preserved in ulterior applications of GAUSS-SEIDEL). The green arc $(3, 6)$ is kept in every application of the algorithm since the node 3 has a loop, and it also produces a second arc $(2, 6)$ starting from node 2. The blue arc $(2, 5)$ generates two arcs starting from nodes 1 and 4 (which are nodes incident to node 2).

The correctness proof of GAUSS-SEIDEL algorithm -which makes use of characterization 2.2- can be revised in Lemma A.3 from appendix. A natural question when studying an algorithm is to ask for its computational complexity. If we look at the three first for loops in GAUSS-SEIDEL, we realize than they are simply an ordered manner to traverse each arc of input network (V, A) , then we must multiply the cost of each internal action of the

loop by $\mathcal{O}(|A|)$. The action in line 7 is $\mathcal{O}(1)$, but the action in line 10 is executed once for each incident arc (t, u) to node u in digraph (V, A') . If the data-structures used for the arc set are in-adjacency lists, it suffices to check the in-list of node u , whose size is $\mathcal{O}(\deg_{(V, A')}^-(u)) = \mathcal{O}(|V|)$. Whether the data-structure is the adjacency matrix of (V, A') , we can know the incident arcs to u by checking the column associated to node u , whose size is $\mathcal{O}(|V|)$, which shows that the total cost of else-bifurcation of the for loop at line 5 (over all the blocks in the schedule) is at most $\mathcal{O}(|V| \cdot |A|)$. If all other costs are assumed to be $\mathcal{O}(1)$, we have proven the following:

Proposition 4.1 *The Gauss-Seidel operator, when applied onto a disjunctive Boolean network (V, A) of size $n = |V|$ and a block-sequential update schedule, can be computed on time at most $\mathcal{O}(|V| \cdot |A|) = \mathcal{O}(n^3)$.*

In subsection A.4 it is proved that, for a disjunctive Boolean network (V, A) of size $n = |V|$ and a block-sequential update schedule of m blocks, the network returned by Gauss-Seidel operator can be computed through $m - 1$ products of square matrices of size n . Since the product of square matrices of size n is $\mathcal{O}(n^3)$, this procedure to compute this operator has roughly the same cost as the procedural characterization seen at this section.

4.2 Convergence for two blocks

Now let us try to get a better understanding of what the Gauss-Seidel operator makes. The arcs of A that are changed after an execution of the algorithm are those starting from some block, and end up at a block updated later. These arcs are replaced in A' by other arcs with the same terminal vertex, but whose initial vertex is a parent of the starting vertex. Hence it holds if $(u, v) \in A$, where the block of u is updated before the block of v , and u has a parent in A' ($(t, u) \in A'$), then the arc (t, v) is added to A' . If t belongs to a block updated later than the one of v , the arc (t, v) is preserved in subsequent applications of GAUSS-SEIDEL. But if t belongs to a block updated before than the one of v , the arc (t, v) is again replaced by a new application of GAUSS-SEIDEL (with the same mechanism of shifting dependence to parent vertices). Thus, it may be conjectured that stabilization of composing Gauss-Seidel operator -understanding stabilization as the non-production of new networks with this procedure- depends significantly, for some block k , on the network topology of the previous $k - 1$ blocks.

Let us examine the case when there are just **two** blocks. Let D be a disjunctive network, and a block-sequential update schedule S for this network with update function s and having only two update instants or blocks -without loss of generality these instants are 0 and 1-, that is, each node of the network belongs to the first block B_0 or to the second block B_1 . To model this, it is proposed the following: given the subdigraph of the network induced by the nodes in B_0 , which we assume it has q nodes ($|B_0| = q$), and some arbitrary fixed node $a \in B_1$, we define the state vector $\vec{x}(a, D) \in \{0, 1\}^q$ representing, for each node i in B_0 , if there exists an arc starting from i and ending on a :

$$\forall i \in \{1, \dots, q\}, \quad x(a, D)_i = 1 \iff (i, a) \in A \quad (s(i) < s(a)) \quad (4.1)$$

If (u, v) is an interior arc of the first block $(u, v \in B_0)$, and $x(a, D)_v = 1$, after applying the algorithm the output network D^S satisfies $x(a, D^S)_u = 1$. It is enough to have a single arc (u, v) of all those that depart from u verifying the above, to have that $x(a, D^S)_u = 1$. This is similar to what happens with a disjunctive Boolean network, where the transition local functions are *OR* functions (disjunctions of variables): here it is enough that some parent of node u has state equal to 1, to have that the state of u equals 1 in the next iteration. If we consider a Boolean network $D(B_0)^T = (B_0, E^T)$, where E^T is the set of arcs that results from reversing the arcs of $D(B_0)$ (the adjacency matrix of $D(B_0)^T$ is the transpose of the adjacency matrix of $D(B_0)$), with local transition functions given by *OR* functions, we could simulate the effect of applying GAUSS-SEIDEL on D and S .

The idea presented in last paragraphs is established in the next lemma, whose proof is on the appendix:

Lemma 4.2 *Let $D = (V, A)$ be a disjunctive network, and a block-sequential update schedule S for this network, with only two blocks B_0 and B_1 , where the block B_0 has q nodes. Let $a \in B_1$ be a node in the second block. Let $M \in \mathcal{M}_{q \times q}(\{0, 1\})$ be the adjacency matrix of $D(B_0)^T$, the subdigraph induced in D by B_0 with reversed arcs, that is:*

$$M_{ij} = 1 \iff (j, i) \in A, (i, j \in B_0)$$

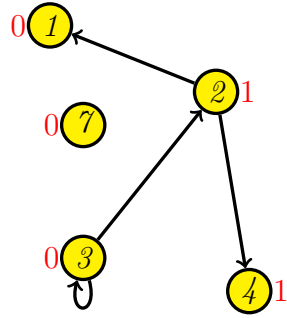
Then $\vec{x}(a, D^S) = \vec{x}(a, D) \cdot M$.

By definition of vectors $\vec{x}(a, D)$, we have that the quantity

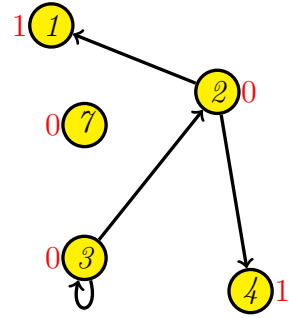
$$\vec{x}(D) = \sum_{a \in B_1} \vec{x}(a, D)$$

is the vector in $\{0, 1\}^q$ which in each component i equals 1 iff there exists some node $a \in B_1$ such that $(i, a) \in A$ (if the sum consists in computing the logical *OR*). If the sum is in \mathbb{Z}^q , then each component i stores the number of arcs in D starting from node i to some node in B_1 .

Example 12 *Let us consider the network D of Example 11, where the schedule S has two updating blocks B_0 and B_1 , and obtain the new network $D(B_0)^T$ comprised of nodes in the first block B_0 (nodes 1, 2, 3, 4, 7, the yellow nodes in the figure), and a set of arcs that results from inverting the arcs from the original network D . This network $D(B_0)^T$ is depicted in the figure below. Consider the vector $\vec{x}(5, D)$, that is, the vector that indicates the arcs incident to node 5 that depart from the first block B_0 in D , and assuming $M(D(B_0)^T)$ is the adjacency matrix of $D(B_0)^T$, obtain the vector $\vec{x}(5, D) \cdot M(D(B_0)^T)$ that results from updating in parallel the network $D(B_0)^T$ with initial condition $\vec{x}(5, D)$.*

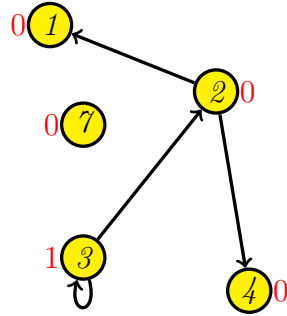


$$\vec{x}(5, D)$$

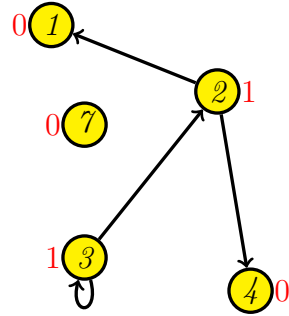


$$\vec{x}(5, D) \cdot M(D(B_0)^T)$$

The last lemma assures $\vec{x}(5, D) \cdot M(D(B_0)^T)$ corresponds to $\vec{x}(5, D^S)$, the vector indicating the arcs starting from block B_0 and ending up on node 5 in the network D^S , the network returned by GAUSS-SEIDEL(D, S), which can be checked in the example. The same can be said about node 6:



$$\vec{x}(6, D)$$



$$\vec{x}(6, D) \cdot M(D(B_0)^T)$$

The previous lemma allows to study the effect of the GAUSS-SEIDEL algorithm -through the adjacency matrix- on the arcs whose starting node belongs to a block updated before the block containing the terminal node, which are the arcs of the input network modified by the algorithm (other arcs of the input are preserved in the output). As a consequence of this fact, iterated applications of the GAUSS-SEIDEL algorithm on an input network can be simulated by the product of integer powers of the adjacency matrix and the input vector $\vec{x}(a, D)$. This provides a tool for studying the stability of the deterministic system consisting of successively applying GAUSS-SEIDEL, which is the stability of powers of matrices, tool already employed on Section 3 and a well studied problem in itself.

Consider a disjunctive network $D = (V, A)$, and a block-sequential update schedule S for this network that has only two blocks of updating B_0 and B_1 . Suppose that the subdigraph $D(B_0)^T$ induced in D by the set of nodes in the first block B_0 (the one updated first), with reversed arcs, is strongly connected (we suppose that this block has q nodes: $|B_0| = q$). By Theorem 3.2 the adjacency matrix M of the subdigraph $D(B_0)^T$ is irreducible, and it could be either primitive or imprimitive. If M is primitive, there exists an exponent $exp(M)$ for which, M^n is positive for all $n \geq exp(M)$, and by Theorem 3.5, M^n is positive for all $n \geq (q-1)^2 + 1$. On the other hand, the Lemma 4.2 states that for every node a in the second

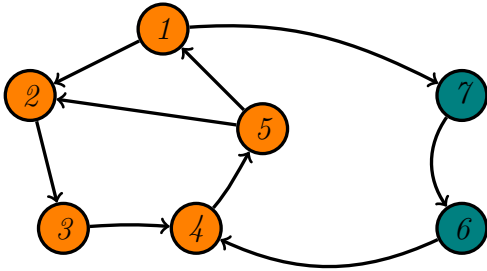
block, the effect of applying GAUSS-SEIDEL algorithm on a vector $\vec{x}(a, D)$, is obtained with the product of M and $\vec{x}(a, D)$. Thus, whether $\vec{x}(a, D)$ is not the null vector (ie, if there is in D an arc with initial node in the first block, and its terminal node a is in the second block), then the product

$$\vec{x}(a, D) \cdot M^n$$

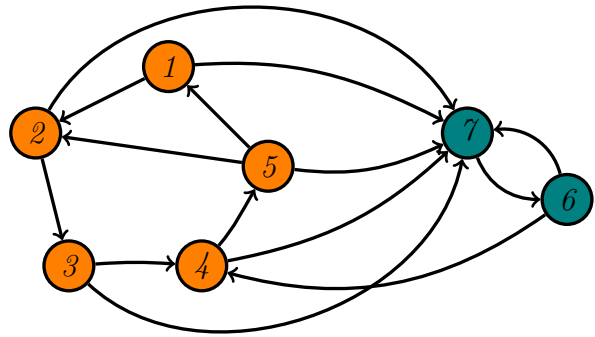
is positive for all $n \geq (q - 1)^2 + 1$. This means that the network returned after $(q - 1)^2 + 1$ applications of GAUSS-SEIDEL, has an arc (q, a) for each node $q \in B_0$, and this is kept in the subsequent applications of the algorithm, resulting in a structural fixed point consisting in a saturated network full of arcs of the form (q, a) . This is the proof sketch of the next lemma:

Lemma 4.3 *Let $D = (V, A)$ be a disjunctive Boolean network, and a block-sequential update schedule S for this network which has only two update instants or blocks B_0 and B_1 , where the first block has q nodes, and the subgraph $D(B_0)$ induced by the first block is strongly connected and primitive. Then, in $\mathcal{O}(q^2)$ executions of GAUSS-SEIDEL, the filter converges to an attractor consisting in a structural fixed point ($|\mathcal{A}(D, S)| = 1$).*

Example 13 *In the images below there is a disjunctive network D of seven nodes, and a block-sequential update schedule of two blocks for this network, where nodes in the first block B_0 are orange, and in the second block are blue. We see the subdigraph induced by the first block, $D(B_0)$, is strongly connected and primitive: it has a circuit of length 5 and other of length 4. Moreover it is known [13] that the adjacency matrix M of $D(B_0)$ is one of the two matrices having an exponent satisfying the relation from Theorem 3.5 with equality: hence $\exp(M) = (5 - 1)^2 + 1 = 17$. This means that after 17 applications of GAUSS-SEIDEL onto network D (and not in a previous iteration, since $\exp(M) = 17$) the filter attractor is reached, and the last result assures this attractor is a structural fixed point. Observe that the arc $(6, 7)$ appears in the third application of GAUSS-SEIDEL: these arcs appear after a number of iterations linear on $|B_0| = 5$.*



D



D^{17}

To study the subcase when the subdigraph induced by the first block $D(B_0)$ is strongly connected and imprimitive, since we have used the same tools from Section 3, we can analo-

Finally, if we want to check for the transient and period length bounds of the filter stated by Lemma 4.4, we can realize that $D(B_0)$ is the same disjunctive network studied in Example 9.

The next result addresses a more general case for block sequential update schedules of two blocks: it encompasses the subcases described by Lemmas 4.3 and 4.4. The following result results to be a nice consequence of Theorem 3.9, since it exploits again Lemma 4.2 to represent the evolution of the filter as the parallel dynamics of certain disjunctive network.

Proposition 4.5 *Let $D = (V, A)$ be a disjunctive Boolean network, and a block-sequential update schedule S for this network which has only two update instants or blocks B_0 and B_1 , where the first block has q nodes, and such that the subdigraph induced by the first block, $D(B_0)$, is weakly connected. Then, in $\mathcal{O}(q^2)$ executions of GAUSS-SEIDEL, the filter converges to an attractor.*

PROOF. Let s be the update function of S . First suppose that D has no arcs (u, v) where $u \in B_0$ and $v \in B_1$. Since $s(u) \geq s(v)$ for every arc (u, v) in G , GAUSS-SEIDEL adds to D^S every arc of D without changes and then returns. Therefore D is a structural fixed point reached after one execution of GAUSS-SEIDEL.

Assume now that there is an arc $(i, a) \in A$ such that $s(i) < s(a)$, meaning that the vector $\vec{x}(a, D)$ is not null. Let $M \in \mathcal{M}_{q \times q}(\{0, 1\})$ be the adjacency matrix of the subdigraph induced in D by the first block, $D(B_0)$, with inverted arcs, ie:

$$M_{ij} = 1 \iff (j, i) \in A, j, i \in B_0$$

Then, by the Lemma 4.2, for every $n \in \mathbb{N}$, $\vec{x}(a, D^n) = \vec{x}(a, D) \cdot M^n$, where D^n is the digraph obtained after n iterated executions of GAUSS-SEIDEL over D (with schedule S). Thus, we see the iterated application of GAUSS-SEIDEL is equivalent (Note 1) to the parallel update of network $D(B_0)^T$, the subdigraph induced in D by the first block with transposed arcs, with initial condition $\vec{x}(a, D)$. Since the subdigraph $D(B_0)^T$ is weakly connected, by Theorem 3.9, there exist $t \in \mathcal{O}(q^2)$, $p \in \mathbb{N}$, such that

$$\vec{x}(a, D^{t+p}) = \vec{x}(a, D^t) \tag{4.2}$$

Arcs connecting nodes in the same block are always preserved by GAUSS-SEIDEL. Assume that there is an arc $(j, k) \in A$ such that $s(k) < s(j)$ -this arc (j, k) is preserved without changes by GAUSS-SEIDEL- and an arc $(i, a) \in A$ such that $s(i) < s(a)$. Assume there is a directed path in $D(B_0)$ from k to i of length $p \in \mathcal{O}(q)$. By Lemma 3.1, $M_{i,k}^p > 0$, and given that $x(a, D)_i > 0$, from Lemma 4.2 it follows that $x(a, D^p)_k > 0$, namely there is in D^p an arc (k, a) , where $s(k) < s(a)$. By the definition of GAUSS-SEIDEL algorithm, in D^{p+1} is added the arc (j, a) , where $s(j) = s(a) = 1$, meaning that GAUSS-SEIDEL is adding an arc that connects nodes belonging to the second block B_1 . These arcs are not described by the vectors $\vec{x}(a, D)$ (because the components of these vectors depict nodes in the first block), but what matters here is that these arcs are added in the iteration $p + 1 \in \mathcal{O}(q)$ of GAUSS-SEIDEL, and these arcs remain in the digraphs D^m , with $m \geq p + 1$.

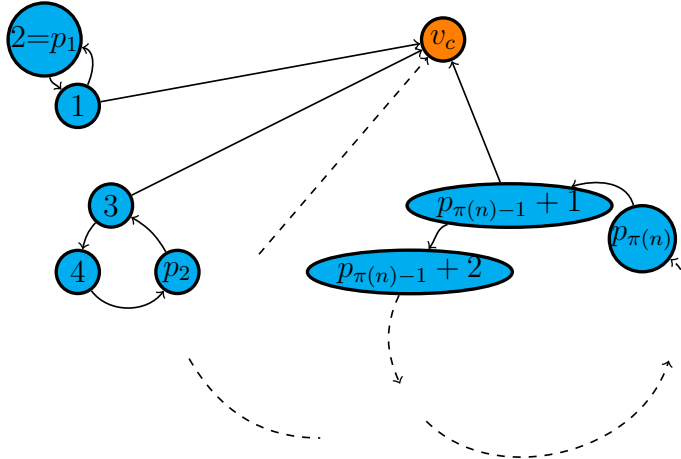
The foregoing paragraph and equality (4.2) show that in at most $\mathcal{O}(q^2)$ executions of GAUSS-SEIDEL over D with schedule S , this iterative procedure outputs a network in the filter attractor $\mathcal{A}(D, S)$.

□

Last results of this subsection show that the transient length of a wide range of filters of two blocks applied to a disjunctive network can be bounded by a quadratic polynomial on the size of the network. Unfortunately, this statement does not apply for the period length. Although Lemma 4.4 could suggest the opposite, it happens that limit cycles of different lengths produced by distinct strongly connected components, interact with each other by multiplying its lengths, which quickly grows the size of the filter attractor. Next theorem shows this effect in networks having disconnected cycles with different lengths, but it is easy to produce this same effect even with cycles with different lengths belonging to the same weakly connected component of the network.

Theorem 4.6 *There is a family of disjunctive Boolean networks $(D_n)_{n \geq 3}$, and a family of block-sequential update schedules $(S_n)_{n \geq 3}$ of two blocks, such that, the filter applied to (D_n, S_n) , for every $n \in \mathbb{N}, n \geq 3$ exhibits super polynomial period length.*

PROOF. Let n be a positive integer, $\pi(n)$ be the number of primes not exceeding n , and $\{p_1, p_2, \dots, p_{\pi(n)}\}$ be the first $\pi(n)$ primes. The network $D_n = (V_n, A_n)$ has a set of $|V_n| = 1 + \sum_{i=1}^{\pi(n)} p_i$ vertices arranged as follows: there are $\pi(n)$ cycles, each one of length p_i and connected through an arc starting on the cycle and ending on a common single vertex v_c as the figure shows. The schedule S_n has two blocks, where the first block to be updated has cyan nodes, and the second block has one orange node (the node v_c).



If we iterate Gauss-Seidel over network D_n and schedule S_n , the procedure does not loop by producing some network produced before since the networks produced are all different. We only recover the initial network after $lcm\{p_1, p_2, \dots, p_{\pi(n)}\} = \prod_{i=1}^{\pi(n)} p_i$ applications of Gauss-Seidel. Or in other words, if we refer to the network obtained after $k \geq 0$ applications of

Gauss-Seidel on (D_n, S_n) as D_n^k , it holds that

$$D_n^{lcm\{p_1, p_2, \dots, p_{\pi(n)}\}} = D_n^0 = D_n, \quad \forall n \geq 3 \quad (4.3)$$

therefore, the transient length is 0 and the period length is equal to $lcm\{p_1, p_2, \dots, p_{\pi(n)}\}$. To obtain a lower bound of the period length, we follow the arguments of Montealegre & Goles (2013) [73] and Kiwi et al (1994) [57]. We have that

$$|V_n| = 1 + \sum_{i=1}^{\pi(n)} p_i \leq 1 + \pi(n) \cdot n \quad (4.4)$$

From the Prime Number Theorem (Theorem 6 [40]) it is known that $\pi(n) = \Theta(n/\log(n))$, then, it holds, for some constant c , that

$$n \leq c \cdot \pi(n) \cdot \log(n) \quad (4.5)$$

replacing (4.5) in (4.4) gives that

$$|V_n| \leq 1 + c \cdot \pi(n)^2 \cdot \log(n) \implies \pi(n) \geq c_1 \cdot \frac{\sqrt{|V_n|}}{\sqrt{\log(n)}} \quad (4.6)$$

for some constant c_1 . From $\pi(n) \leq n$, replacing this inequality in (4.4), it holds also that

$$|V_n| \leq 1 + n^2 \implies n \geq c_2 \cdot \sqrt{|V_n|} \quad (4.7)$$

for some constant c_2 . In the other hand

$$lcm\{p_1, p_2, \dots, p_{\pi(n)}\} = \prod_{i=1}^{\pi(n)} p_i = \exp(\theta(n)) \quad (4.8)$$

where $\theta(n) = \sum_{i=1}^{\pi(n)} \log(p_i)$. It is known (Theorem 420 [40]), that this magnitude meets that $\theta(n) = \Theta(\pi(n) \log(n))$, ie, there exists some constant c_3 such that

$$\theta(n) \geq c_3 \cdot \pi(n) \cdot \log(n) \quad (4.9)$$

replacing (4.9) in (4.8), and then employing (4.6) and (4.7), leads to

$$lcm\{p_1, p_2, \dots, p_{\pi(n)}\} \geq \exp(c_1 c_3 \sqrt{|V_n|} \sqrt{\log(n)}) \geq \exp(\Omega(\sqrt{|V_n|} \cdot \log |V_n|))$$

which shows that the period length of (D_n, S_n) is not bounded by any polynomial in $|V_n|$. \square

4.3 Complexity of computing the attractor

In this subsection we make an estimation of the total computational cost of the filtering procedure for schedules of two blocks, which covers the calculation of the output network on every Gauss-Seidel iteration, and an important aspect in practice related with network comparison in order to determine the filter attractor, or when to stop the filtering procedure.

Let us precise the latter: we already said in Subsection 1.7 that for any network N there exist a transient t and a period p satisfying

$$N^{(t+p)} = N^t \quad (4.10)$$

where for $i \in \mathbb{N}$, N^i denotes the network obtained after i successive applications of Gauss-Seidel operator starting from N . Since we do not know a priori the values of the transient and the period, we must iterate the GAUSS-SEIDEL algorithm until we reach some network in the attractor, indeed, since the transient and the period are the smallest integers verifying (4.10), we iterate until we encounter the first network verifying this equality. But again, since we do not know the value of the period, we must compare the current network with all the previous ones.

Thus, according to previous discussion, for a disjunctive Boolean network $D = (V, A)$ of size n ($|V| = n$), if we name $GSOp(k)$ to the number of elementary operations executed when computing the k -th iteration of GAUSS-SEIDEL, $CompOp(k, h)$ to the number of elementary operations necessary to perform the comparison of k -th network with h -th network, the total number of operations necessary to compute the networks in the filter attractor of network D (and some block-sequential update schedule S) is given by

$$\sum_{k=1}^{(t+p)} \left(GSOp(k) + \sum_{h=0}^{k-1} CompOp(k, h) \right) \quad (4.11)$$

The network comparisons in this situation are made over networks having the same vertex set V (and the same vertex labellings), therefore the comparison is simply checking the equality of the respective set of arcs, which can be approximated by a time cost of $\mathcal{O}(|A|) = \mathcal{O}(n^2)$. By Proposition 4.1, $GSOp(k)$ is approximated by $\mathcal{O}(|V| \cdot |A|) = \mathcal{O}(n^3)$, then (4.11) is bounded by

$$\mathcal{O}(n^3) \cdot \sum_{k=1}^{(t+p)} (1+k) = \mathcal{O}(n^3) \cdot \left((t+p) + \frac{(t+p) \cdot (t+p+1)}{2} \right) \quad (4.12)$$

By Proposition 4.5, we can assume that for schedules of two blocks, in general (at least when the subdigraph induced by the first block is weakly connected) the transient length t of a filter is $\mathcal{O}(n^2)$. If we assume the period length is produced by the length of some cycle belonging to one strongly connected component, or even by several cycles with the same length, this allows to linearly bound the period length: $p \in \mathcal{O}(n)$ (Lemma 4.4). With these assumptions, finally replacing in expression (4.12) gives a total number of elementary operations necessary to compute the filter attractor bounded by

$$\mathcal{O}(n^3) \cdot \mathcal{O}(n^4) = \mathcal{O}(n^7)$$

In the case that the period length is larger than the previous assumption, it can be proposed a heuristic consisting in imposing a polynomial maximum number of networks to compute. Since we know (Proposition 4.5) the transient length is at most quadratic on the size of the input network, after a sufficiently large number of computed networks, we can stop to compute additional networks. It is very likely the last networks produced belong to

the filter attractor, although the relation (4.10) has not happened with the current computed networks.

Chapter 5

Filtering dynamic cycles

We already saw in last section that the iterated composition of Gauss-Saidel operator on disjunctive Boolean networks stabilizes or reaches a set of networks known as the filter attractor, after a quadratic number of applications of this operator. In this section we are interested in studying when this procedure has the intended meaning explained in Subsection 1.7, that is, when this procedure filters out the limit cycles from the dynamics of networks in the filter attractor. If this is the case, it holds that networks in the filter attractor have only fixed points with parallel dynamics, which necessarily are also fixed points from the input network (see Subsection 1.7).

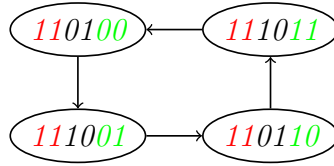
An overview of this section is the following: in Subsection 5.1, we review Lemma 5.1, which relates properties of attractors, with structural properties (specifically, the index of imprimitivity seen in Subsection 3.1) of the considered disjunctive network. Then, we will need to relate structural properties from the input network to structural properties of networks in the respective filter attractor, for which the concept of Induced Length of a cycle with respect to a block-sequential schedule (Definition 5.2) will be useful. In terms of this concept we formulate the Cycle-filter condition (Definition 5.4) for a pair network-schedule; if some pair network-schedule complies this condition, Theorem 5.5 (see Subsection 5.2) assures that networks in the filter attractor do not exhibit limit cycles with parallel dynamics. In Subsection 5.3 it is studied the complexity of checking that a pair network-schedule meets the Cycle-filter condition, which results to be polynomial (Theorem 5.7). To finalize with a related matter, in Subsection 5.4 it is studied the complexity of checking the condition required by Theorem 5.8 (introduced in [39]) valid for sequential schedules.

5.1 The Cycle-filter condition

The next result is an important tool hereinafter since it relates, for a disjunctive network D and a block-sequential schedule S , structural properties of digraph D^S with dynamics $STG(D, S)$. But first we need to introduce some notation. Recall that $k(D) \in \mathbb{N}$, for a strongly connected digraph D , is the greatest common divisor of the lengths of the directed cycles of D . Let D be a disjunctive Boolean network updated with a block-sequential update

schedule S , and let C be any non-trivial strongly connected component of D^S . For an attractor $A \subseteq \{0, 1\}^n$ of the dynamics $STG(D, S)$, ie, A is a set of one -fixed point- or more -limit cycle- vector states forming a cycle in the digraph $STG(D, S)$, we define the *period of A projected to C* , which we will denote as $p(A, C)$, as the smallest positive integer verifying $y_i(t) = y_i(t + p(A, C))$, for every $i \in C$ and $\vec{y}(t) \in A$. It is clear that these projected periods are upper bounded by the respective period of the attractor: $p(A, C) \leq |A|$.

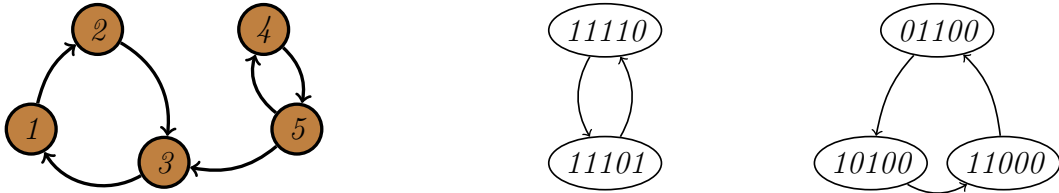
Example 15 *In the figure we can appreciate an attractor of period 4 (ie, a limit cycle) of a network D of size 6 and some block-sequential update schedule S . The network D^S has three non-trivial strongly connected components, C_1, C_2, C_3 , each of them with 2 nodes, and they are associated respectively to the first 2 (red) nodes, the intermediate (black) and the last 2 (green) nodes in the states of the figure. Thus, we conclude that $p(A, C_1) = 1$, $p(A, C_2) = 2$ and $p(A, C_3) = 4$.*



A limit cycle of period 4 in $STG(D, S)$

Lemma 5.1 ([37]) *Let D be a disjunctive Boolean network updated with a block-sequential update schedule S and let C be a non-trivial strongly connected component of D^S . Then, for every attractor A in the dynamics of $STG(D, S)$, the period $p(A, C)$ of A projected to C divides $k(D^S(C))$, and there exists an attractor in which C cycles with period $k(D^S(C))$. In particular, $k(D^S(C)) = 1$ if and only if the state of C is fixed in all attractors of $STG(D, S)$.*

Example 16 *The following disjunctive network D has two strongly connected components, one is comprised of nodes 1, 2 and 3, and the other consists of the nodes 4 and 5. We refer to these components as C_1 and C_2 , respectively. It is clear that the indexes of imprimitivity of each component are $k(D(C_1)) = 3$ and $k(D(C_2)) = 2$. In order to apply Lemma 5.1, we consider in the hypothesis $S = \pi$, the parallel schedule, so the non-trivial s.c.c. of D^S are also the s.c.c. of $D^\pi = D$. The lemma states that for every attractor in the dynamics of $STG(D, S)$ -for example, the limit cycle \mathcal{L} of period 2 deployed at the bottom right-, the periods obtained when projecting to the components C_1 or C_2 , divide the respective index $k(D(C_i))$. For example, the period $p(\mathcal{L}, C_1)$ is equal to 1 (the states are fixed in these nodes), and $p(\mathcal{L}, C_1) | k(D(C_1))$; moreover, the period $p(\mathcal{L}, C_2)$ is equal to 2, and $p(\mathcal{L}, C_2) | k(D(C_2))$. The same is true for the limit cycle of period 3.*



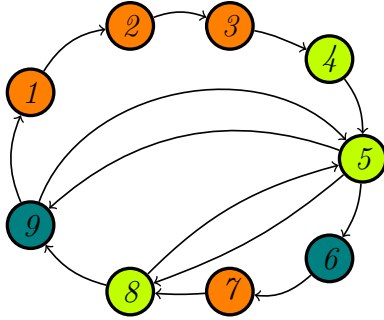
Two limit cycles of D

D

Lemma 5.1 allows to affirm, for example, that a disjunctive network D , updated with parallel schedule π , does not have limit cycles if and only if every s.c.c. of D is primitive. This fact provides means to determine if networks in the filter attractor show limit cycles. Then we would like to find a way to relate the primitivity of s.c.c.'s in the filter attractor, to the input network handed over to the algorithm. The next concept will be useful to achieve this goal.

Definition 5.2 (Induced length) *Let $C \subseteq D = (V, A)$ be a cycle (or a simple directed circuit), and let S be a block-sequential update schedule for network D with $m \geq 1$ blocks $(B_j)_{j=0}^{m-1}$. Let $0 \leq i_C \leq m - 1$ be the instant such that B_{i_C} is the last block of those induced by S having nodes from the circuit C . The induced length of C with respect to S , denoted by $\mathcal{IL}(C, S)$, correspond to the quantity of nodes of C , that belong to block B_{i_C} .*

Example 17 *In the image below at left there is a disjunctive network D of nine nodes, and a block-sequential update schedule S of three blocks for this network, where nodes in the first block B_0 are green (nodes 8, 4, 5), nodes in the second block B_1 are blue (nodes 9, 6) and those in the third block B_2 are orange (nodes 1, 2, 3, 7). In the table below at right we can see several cycles of G , each of them with its respective last block and induced length:*



G

<i>Cycle C</i>	<i>Instant i_C</i>	<i>$\mathcal{IL}(C, S)$</i>
(1, 2, 3, 4, 5, 9)	2	3
(1, 2, 3, 4, 5, 8, 9)	2	3
(5, 6, 7, 8)	2	1
(5, 6, 7, 8, 9)	2	1
(1, 2, 3, 4, 5, 6, 7, 8, 9)	2	4
(8, 5)	0	2
(9, 5)	1	1
(9, 5, 8)	1	1

Notice that the case of circuit (8, 5) is the same of all those circuits connecting nodes pertaining to the same block. Here the induced length of the circuit is equal to the usual cycle length.

Note 4 *The induced length of a cycle can be characterized in an equivalent manner which results to be more complicated but will be useful for the proofs. Let $C \subseteq D = (V, A)$ be a cycle (or a simple directed circuit), and let S be a block-sequential update schedule for this network with $m \geq 1$ blocks $(B_j)_{j=0}^{m-1}$. Let $0 \leq i_C \leq m - 1$ be the instant such that B_{i_C} is the last block of those induced by S having nodes from the circuit C . The characterization is as follows. Assume that in the subdigraph induced in D by B_{i_C} , the circuit C reduces to n_C maximal directed paths, where each j -th path connects $(\ell_j + 1)$ different nodes ($1 \leq j \leq n_C$ and $0 \leq \ell_j \leq |C| - 1$). The induced length of C with respect to S correspond to*

$$\mathcal{IL}(C, S) = \sum_{j=1}^{n_C} \ell_j + n_C$$

The following proposition shows that the induced length, allows to predict the length of certain circuits that are present in the networks belonging to the filter attractor.

Proposition 5.3 *Let $D = (V, A)$ be a disjunctive Boolean network, and a block-sequential update schedule S for this network with $m \geq 1$ blocks $(B_j)_{j=0}^{m-1}$. If there is a simple circuit C in D , such that $\mathcal{IL}(C, S) = L$, then there is a circuit of length equal to L in every network in $\mathcal{A}(D, S)$.*

PROOF. Let s be the update function of S and assume there is a simple circuit C such that $\mathcal{IL}(C, S) = L$. Then if C reduces to n_C maximal directed paths in $D(B_{i_C})$ (where B_{i_C} is the last block having nodes from the circuit C , $0 \leq i_C \leq m - 1$), where each j -th path connects $(\ell_j + 1)$ different nodes, it holds that $L = \sum_{j=1}^{n_C} \ell_j + n_C$. If $n_C = 1$ and the unique maximal path connects $|C| = 1 + \ell_1$ different nodes, this means that all the nodes connected by C belong to B_{i_C} , then C is preserved and it belongs to the networks in $\mathcal{A}(D, S)$, so $L = \ell_1 + n_c = |C| - 1 + 1 = |C|$, and the proposition holds in this case. Let P_j be one of these paths, for which its ending node will be referred as $e_j \in B_{i_C}$. Since P_j corresponds to a sequence of arcs from C , there is a path in D from e_j to i_{j+1} , where $i_{j+1} \in B_{i_C}$ is the initial node from the next maximal path P_{j+1} induced in $D(B_{i_C})$ by circuit C (if $n_C = 1$, then i_{j+1} is simply i_j , the starting node from path P_j). Call this path as $P_{e_j, i_{j+1}}^D$. It holds that, for every node $a \in V$ connected by $P_{e_j, i_{j+1}}^D$ with $a \neq e_j, i_{j+1}$, $s(a) < s(e_j) = s(i_{j+1})$ (if not, P_{j+1} would not be the next maximal path induced by C in $D(B_{i_C})$).

As before, we refer to the network obtained after $i \geq 0$ applications of GAUSS-SEIDEL over D (and schedule S) as D^i . If we take the subdigraph induced by the first $(i_C + 1)$ blocks, $D(B_0 \cup \dots \cup B_{i_C})$, and the schedule S' consisting in the schedule S restricted to the first $(i_C + 1)$ blocks, by Lemma A.5 we obtain that for every $i \geq 0$, there is a path $P_{e_j, i_{j+1}}^{D^i}$ in $D(B_0 \cup \dots \cup B_{i_C})^i$, which starts from e_j to i_{j+1} , and connects nodes belonging to the set of nodes connected by $P_{e_j, i_{j+1}}^D$. By Lemma A.9, this path $P_{e_j, i_{j+1}}^{D^i}$ is also in D^i . Also, it holds that $|P_{e_j, i_{j+1}}^{D^{i+1}}| \leq |P_{e_j, i_{j+1}}^{D^i}|$, for all $i \geq 0$. Now, given that the length of any path is positive, the previous inequality and the fact that $|P_{e_j, i_{j+1}}^D| \leq |C| - |\sum_j P_j|$, there is an integer $\bar{i} \leq |C| - |\sum_j P_j|$ verifying

$$|P_{e_j, i_{j+1}}^{D^{\bar{i}+1}}| = |P_{e_j, i_{j+1}}^{D^{\bar{i}}}|$$

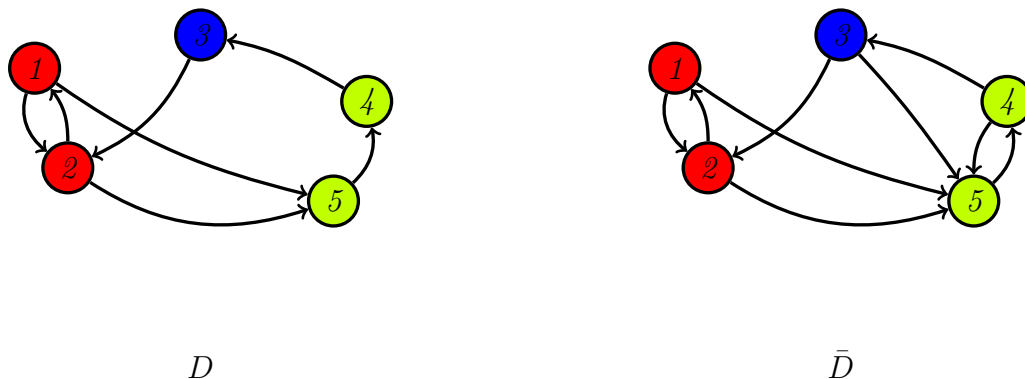
Consider the node incident to i_{j+1} in $P_{e_j, i_{j+1}}^{D^{\bar{i}}}$, namely, the node $u_{\bar{i}}$ meeting that $(u_{\bar{i}}, i_{j+1}) \in P_{e_j, i_{j+1}}^{D^{\bar{i}}}$. Necessarily, this node is one of the nodes connected by $P_{e_j, i_{j+1}}^D$: but if $u_{\bar{i}} \neq e_j$ ($u_{\bar{i}} \neq i_{j+1}$ because C is a simple circuit), then $s(u_{\bar{i}}) < s(i_{j+1})$, and by the same reasoning in the proof of Lemma A.5, it is obtained that $|P_{e_j, i_{j+1}}^{D^{\bar{i}+1}}| < |P_{e_j, i_{j+1}}^{D^{\bar{i}}}|$. This shows that necessarily $u_{\bar{i}} = e_j$, or in other words, $P_{e_j, i_{j+1}}^{D^{\bar{i}}} = (e_j, i_{j+1})$. Since $s(e_j) = s(i_{j+1})$, the arc (e_j, i_{j+1}) is preserved in subsequent applications of GAUSS-SEIDEL. Since this arc does not become -in ulterior iterations- to some of the previous arcs that ended on i_{j+1} , the networks D^i with $i < \bar{i}$ cannot be part of the attractor. Therefore, the attractor is reached in some iteration later to \bar{i} , and since (e_j, i_{j+1}) is preserved at each iteration, it holds that (e_j, i_{j+1}) is part of all the networks that constitute the attractor $\mathcal{A}(D, S)$. In this way, each path $P_{e_j, i_{j+1}}^D$ connecting the maximal path P_j with maximal path P_{j+1} ends up as just one arc (e_j, i_{j+1}) , finally forming,

after $\mathcal{O}(|C| - |\sum_j P_j|)$ iterations, a closed path which connects nodes in B_{i_C} of length equal to

$$\sum_{j=1}^{n_C} (\ell_j + 1) = \sum_{j=1}^{n_C} \ell_j + n_C = \mathcal{IL}(C, S) = L$$

□

Remark 2 Proposition 5.3 is not an equivalence, that is, in any network belonging the filter attractor $\mathcal{A}(D, S)$ there may be a circuit of length L such that none circuit of D has induced length equal to L . The following example shows one of such cases. Let D be the disjunctive network defined by the digraph in the figure, and let S be the following block-sequential update schedule of 3 blocks for the network D : $B_0 = \{1, 2\}$ (the red nodes in the figure), $B_1 = \{3\}$ (the blue node in the figure), $B_2 = \{4, 5\}$ (the green nodes in the figure). After 2 applications of GAUSS-SEIDEL on (D, S) , the network \bar{D} is obtained, which is a structural fixed point for GAUSS-SEIDEL. Every simple circuit of D has induced length equal to 2: $\mathcal{IL}((1, 2), S) = 2$, $\mathcal{IL}((2, 5, 4, 3), S) = 2$, and $\mathcal{IL}((2, 1, 5, 4, 3), S) = 2$. However in \bar{D} there are circuits $C_1 = (4, 5, 3)$, $C_2 = (5, 4, 3, 2)$ and $C_3 = (5, 4, 3, 2, 1)$ of lengths 3, 4, and 5, respectively.



We can now state a condition on the input network and schedule that ensures networks in the filter attractor do not have limit cycles. The proof of this property is on the next subsection.

Definition 5.4 (Cycle-filter condition) Let $D = (V, A)$ be a disjunctive Boolean network, and consider a block-sequential update schedule S for this network. The pair (D, S) is **cycle-filter** if it satisfies the following.

- (i) Every non-trivial strongly connected component of digraph D contained in just one updating block induced by schedule S is primitive.
- (ii) Every non-trivial strongly connected component of digraph D belonging to more than one updating block induced by schedule S , verifies that the greatest common divisor of the induced lengths of its simple circuits equals 1.

5.2 Sufficiency of Cycle-filter

Theorem 5.5 *Let $D = (V, A)$ be a disjunctive Boolean network, and S a block-sequential update schedule for this network. If (D, S) is cycle-filter then the networks in $\mathcal{A}(D, S)$ do not have limit cycles when updated with the parallel schedule π .*

PROOF. Let (D, S) be a pair digraph-schedule which is cycle-filter and let \bar{D} be a network in $\mathcal{A}(D, S)$, the filter attractor of the pair (D, S) . By applying inductively Lemma A.8, every non-trivial s.c.c. C of D produces at most one non-trivial s.c.c. \bar{C} in \bar{D} . Whenever a non-trivial s.c.c. C of D is studied in isolation from the rest of the digraph, the Lemma A.9 is being used implicitly, that is, if C belongs to the first n blocks of S , we use the schedule S' resulting in restricting S to the first n blocks and the subdigraph $D(C)$ of D induced by the nodes of C . The conclusions drawn for $D(C)$ are also valid for D thanks to Lemma A.9.

Let C be a non-trivial strongly connected component in D , and suppose first C is completely contained in just one block induced by S , implying C is preserved in every application of GAUSS-SEIDEL, and thus, C is a s.c.c. in \bar{D} . We first show by contradiction that no new circuits are added to C . On the contrary, if this were the case, the characterization 2.2 says that a circuit \bar{C} in C appearing in any of the applications of GAUSS-SEIDEL is equivalent to the existence of a circuit C^* in D connecting the nodes connected by \bar{C} and some other nodes that are not in the block of C (as \bar{C} was not originally in C). This means that C belongs to more than one block, which is absurd. Since (D, S) is cycle-filter, C is primitive, and in this way, it holds that every s.c.c. of D contained in just one updating block ends up in $\bar{D} \in \mathcal{A}(D, S)$ as a primitive s.c.c.

Now let C be a non-trivial s.c.c. of D belonging to more than one updating block induced by S . If C consists of only one simple circuit \mathcal{C} , as (D, S) is cycle-filter then \mathcal{C} has induced length with respect to S equal to 1. Then by Proposition 5.3, the s.c.c. $\bar{C} \in \bar{D}$ generated by $C \in D$ has a loop and therefore \bar{C} is primitive. If C is comprised of several circuits, since (D, S) is cycle-filter the greatest common divisor of its induced lengths is 1, or in other words, the set of induced lengths of cycles in C is setwise coprime. By Proposition 5.3, the s.c.c. $\bar{C} \in \bar{D}$ generated by $C \in D$, has a circuit of length L for every circuit in C of induced length L . Therefore, the set of lengths of cycles in \bar{C} is also setwise coprime, implying that $k(\bar{D}(\bar{C})) = 1$ and \bar{C} is primitive. This shows that, if (D, S) is cycle-filter, every non-trivial s.c.c. \bar{C} of any network \bar{D} in $\mathcal{A}(D, S)$ is primitive, which implies by Lemma 5.1 (employing the parallel schedule π in the hypothesis of the lemma) that the state of every s.c.c. \bar{C} of \bar{D} is fixed in all attractors of $STG(\bar{D}, \pi)$, meaning that the dynamic of $STG(\bar{D}, \pi)$ does not have limit cycles. This shows the condition sufficiency. □

Remark 3 *The condition 5.4 is not necessary in general. That is, it may happen that none of the networks in $\mathcal{A}(D, S)$ exhibits limit cycles when updated with the parallel schedule π , but the pair (D, S) is not cycle-filter. Let D and S be the same disjunctive network and block-sequential update schedule defined in Remark 2. The pair (D, S) is not cycle-filter, since D is strongly connected, the nodes in D belong to more than one updating block induced*

by schedule S and the induced lengths with respect to S of its circuits ($\mathcal{IL}((1, 2), S) = 2$, $\mathcal{IL}((2, 5, 4, 3), S) = 2$, and $\mathcal{IL}((2, 1, 5, 4, 3), S) = 2$) are even. However, the network \bar{D} is a primitive strongly connected digraph, then Lemma 5.1 allows to affirm that \bar{D} does not have limit cycles with parallel dynamics.

Despite the condition cycle-filter is not necessary in the general case, it is possible to find some cases when this property is actually necessary. The following theorem shows one of such cases.

Theorem 5.6 *Let $D = (V, A)$ be a disjunctive Boolean network, and a block-sequential update schedule S for this network, such that every strongly connected component of D contained in more than one block consists of a unique simple circuit. If the networks in $\mathcal{A}(D, S)$ do not have limit cycles when updated with the parallel schedule π then (D, S) is cycle-filter.*

PROOF. We now show the condition is necessary with the assumption specified on the theorem assertion. Let \bar{D} be a network in $\mathcal{A}(D, S)$, the filter attractor of the pair (D, S) . Just as it was done in the proof of Theorem 5.5, whenever a non-trivial s.c.c. C of D is studied in isolation from the rest of the digraph, the Lemma A.9 is being used implicitly: if C belongs to the first n blocks of S , we use the schedule S' resulting in restricting S to the first n blocks and the subdigraph $D(C)$ of D induced by the nodes of C . The conclusions drawn for $D(C)$ are also valid for D thanks to Lemma A.9. Also, by applying inductively Lemma A.8, every non-trivial s.c.c. C of D produces at most one non-trivial s.c.c. \bar{C} in \bar{D} . Assume (D, S) is not cycle-filter, namely (D, S) does not satisfy (i) or (ii) from Definition 5.4.

In the first case, there is in D an imprimitive s.c.c. C contained in just one block induced by S , implying C is preserved in every application of GAUSS-SEIDEL, and thus, C is a s.c.c. in \bar{D} . No new circuits are added to C (by the same argument put in the proof of Theorem 5.5), then the primitivity of C is not modified and there is in every $\bar{D} \in \mathcal{A}(D, S)$ an imprimitive strongly connected component.

In the second case, there is a s.c.c. $C \in D$ belonging to more than one updating block induced by S such that it consists of a unique simple circuit with induced length with respect to S equal to $d > 1$, implying by Proposition 5.3 that in the s.c.c. $\bar{C} \in \bar{D}$ generated by $C \in D$ there is a circuit of length $d > 1$. If we suppose the s.c.c. $\bar{C} \in \bar{D}(\bar{C})$ has more than one simple circuit, necessarily there is in \bar{C} a node with in-degree greater than one, which violates Lemma A.6. Then the s.c.c. $\bar{C} \in \bar{D}$ has only one simple circuit of length $d > 1$, meaning that $k(\bar{D}(\bar{C})) = d > 1$ and \bar{C} is imprimitive.

This shows that, if (D, S) is not cycle-filter, for every network \bar{D} in $\mathcal{A}(D, S)$ there exists some imprimitive s.c.c. $\bar{C} \in \bar{D}$, and finally by Lemma 5.1 there is an attractor in which \bar{C} cycles (updated with the parallel schedule π) with period $k(\bar{D}(\bar{C})) > 1$. \square

5.3 Complexity of checking Cycle-filter

We address now the problem of determining the complexity of checking Definition 5.4, the Cycle-filter condition. As first observation, given a pair (D, S) , we need to calculate the strongly connected components of the digraph $D = (V, A)$. This is not difficult since there are several algorithms to perform this task in linear time, for example, Kosaraju’s algorithm [82] and Tarjan’s algorithm [84], which work in $\mathcal{O}(|V| + |A|)$. Once we have a s.c.c. C , we need to know the index of imprimitivity of C . We present here an algorithm extracted from [9], [20] that will be very useful hereafter, which calculates this index in $\mathcal{O}(|A|)$. This algorithm receives as input a strongly connected digraph D , with vertex set V , and adjacency lists $(L(v))_{v \in V}$ storing the set of arcs: for each vertex $v \in V$, $L(v)$ corresponds to the set of vertices w for which (v, w) is an arc of D . The algorithm assumes that a spanning directed tree T of D with root r has been previously determined (for example, by means of a depth-first search); this tree allows to get a distances vector d , such that, for each vertex $a \in V$, $d(a)$ is the length of the unique directed chain in T from r to a (with $d(r) = 0$). In this way, the algorithm INDEX returns the index of imprimitivity of D .

```

INDEX( $V, (L(v))_{v \in V}, d$ )
1   $\delta = 0$ 
2  for  $a \in V$ 
3      for  $b \in L(a)$ 
4           $\delta = \text{gcd}\{\delta, d(a) - d(b) + 1\}$ 
5  return  $\delta$ 

```

The correctness proof of this algorithm can be revised in [13]. The algorithm uses the convention that $\text{gcd}\{0, 0\} = 0$. By last observations, we can see that is not difficult to check the part (i) of being cycle-filter: in linear time, we find the strongly connected components, and then in linear time on every component, we can compute the respective index of imprimitivity. In fact, the most used algorithms to find the s.c.c.’s (including Kosaraju’s and Tarjan’s algorithms), perform a depth-first search, which obtains a depth-first spanning directed forest. If we make use of, for example, the Kosaraju’s algorithm, this algorithm gets one root for every depth-first tree of the forest (by visiting nodes in order of decreasing finishing timestamps), each of these trees corresponding to every strongly connected component. Having these roots, it is easy to compute in linear time $\mathcal{O}(|A|)$ the distances vector d required later by INDEX (by traversing the spanning tree from the root). Notice that all these algorithms, either DFS, Kosaraju, Tarjan, and INDEX, represent the digraph as a collection of adjacency lists. Finally, to determine whether a component “belongs to more than one updating block induced by schedule S ”, it is enough to do a single pass over the nodes of the component, evaluating the update function s of schedule S on each node.

It remains to be seen the complexity of checking part (ii) of the definition. This introduces a decision problem in itself that we define here.

CYCLE-FILTER

Input: A strongly connected graph $D = (V, A)$; a block-sequential update S for D

Question: Does D verify the set of induced lengths with respect to S of its circuits is setwise coprime?

We show next that the above problem can be solved in polynomial time. A brute force strategy is to find all simple circuits and compute the induced length of each one. There is a work (Johnson, 1975 [51]) which proposed an algorithm to find all simple circuits of a directed graph D in time bounded by $\mathcal{O}((n + a)(c + 1))$, where D has n vertices, a arcs and c simple circuits. Then the induced length of a simple circuit can be easily obtained in linear time on the length of the circuit. The problem is that the number of simple circuits in a directed graph can grow faster with n than exponential ([51]). The difficulty due to the existence of certain cycles verifying its induced lengths do not have common divisors, can be avoided by the same algorithm INDEX, which determines whether all the cycles have a common divisor by means of a spanning DFS tree. The key that allows to employ the INDEX algorithm, is the Proposition 5.3, which ensures that a cycle in the initial network having an induced length L , implies the existence of a cycle in every network of the attractor with length equal to L . Thus, if we could add the cycles existing in the filter attractor associated to cycles of certain induced length existing in the initial network, then we could test the primitivity of networks in the attractor by applying the INDEX algorithm.

This is the idea implemented by the next algorithm, called TEST-CYCLE-FILTER, whose pseudo-code is written below. This algorithm assumes that, for each $v \in V$, the sets $S^{in}(v), S^{out}(v) \subseteq V$ have been initialized as follows: $S^{in}(v)$ corresponds to the set of nodes incident to v (or in the case of $S^{out}(v)$, the nodes at which v is incident), that are belonging to blocks previous to the block of v (the same for $S^{out}(v)$). By this reason, for every node v in the first block, $S^{in}(v) = S^{out}(v) = \phi$. These sets are easily computed in one pass over the arcs of D taking $\mathcal{O}(|A|)$ time.

Before presenting the algorithm TEST-CYCLE-FILTER, we describe the sub-routine ADD-EDGES(D, i, \hat{D}_i, S) used by the algorithm. This sub-routine receives as input the input digraph D , the blocks $(B_\ell)_{\ell=0}^{m-1}$ of schedule S , the instant $0 \leq i \leq m - 1$ and the subdigraph \hat{D}_i induced in D by the block B_i . The purpose of ADD-EDGES is to add, to the subdigraph $D(B_i)$, the same arcs that would be added to D if we were to execute the filter (the iterated composition of GAUSS-SEIDEL) over (D, S) . If we refer to the subdigraph induced in D by blocks previous to B_i as $D_{prev}^i = D(B_0 \cup \dots \cup B_{i-1})$, we see from the code below that ADD-EDGES(D, i, \hat{D}_i, S) adds an arc (w, v) to the subdigraph \hat{D}_i , where w, v belong to B_i , if there exists a path P in D from w to v , such that the intermediate nodes of P belong to blocks previous to block B_i . This path is comprised by some arc (w, y) , where y is some node in $S^{out}(w)$, some path $P_{y,x}$ from y to x in D_{prev}^i , where x is some node in $S^{in}(v)$ and the arc (x, v) . Given that for every node v in the first block B_0 , $S^{in}(v) = S^{out}(v) = \phi$, when ADD-EDGES processes the first block it does nothing and returns the subdigraph \hat{D}_i without changes.

```

ADD-EDGES( $D, i, \hat{D}_i, S$ )
1  if  $i = 0$  return  $\hat{D}_i$ 
2   $D_{prev}^i = D(B_0 \cup \dots \cup B_{i-1})$ 
3  for  $v, w \in B_i$ 
4      if  $(S^{in}(v) \neq \phi \wedge S^{out}(w) \neq \phi)$ 
5          for  $x \in S^{in}(v), y \in S^{out}(w)$ 
6              if Is there a path from  $y$  to  $x$  in  $D_{prev}^i$  ?
7                   $\hat{D}_i = \hat{D}_i \cup (w, v)$ 
8  return  $\hat{D}_i$ 

```

Procedure TEST-CYCLE-FILTER makes use of two other sub-routines that we now explain. The first of them is NON-TRIVIAL-SCC(D), which receives a digraph D as argument and computes the non-trivial strongly connected components of D . For this, it uses any of the existing algorithms that compute the strongly connected components of a digraph (Tarjan is the most used in practice due to its efficiency) and we discard all those trivial components consisting in only one node without loop. Finally NON-TRIVIAL-SCC(D) returns a set $\{(C_1, r_1), \dots, (C_J, r_J)\}$ where each pair (C_j, r_j) is formed by some non-trivial component C_j of the input graph D , and the respective root r_j of C_j (which can be any node of the set C_j). The trivial strongly connected components are not only discarded from the output of NON-TRIVIAL-SCC(D), but they are also erased from input graph D , therefore, after executing NON-TRIVIAL-SCC(D) the input graph D only has non-trivial strongly connected components. The last sub-routine is named REPLACE-NODE(D, v_1, v_2), which receives as argument a digraph D and two nodes v_1, v_2 of D , and it replaces, in every arc of D where the node v_1 appears, the node v_1 by node v_2 . This can be done easily in one single pass over the arcs of D , and still more quickly if we have in and out adjacency lists for every node of D . Finally we erase node v_1 from digraph D . Given that TEST-CYCLE-FILTER performs REPLACE-NODE(\hat{D}, r_j, r_1) for each $j = 2, \dots, J$ (or for each non-trivial strongly connected component different from the first one) we have that, in line 8 of TEST-CYCLE-FILTER, the digraph \hat{D} is strongly connected.

```

TEST-CYCLE-FILTER( $D = (V, A), S$ )
    Let  $B_0, \dots, B_{m-1}$  the blocks of schedule  $S$ 
1  for  $i = 0, \dots, m - 1$ 
2       $\hat{D}_i = D(B_i)$ 
3       $\hat{D}_i = \text{ADD-EDGES}(D, i, \hat{D}_i, S)$ 
4   $\hat{D} = \hat{D}_0 \cup \dots \cup \hat{D}_{m-1}$ 
5   $\{(C_1, r_1), \dots, (C_J, r_J)\} = \text{NON-TRIVIAL-SCC}(\hat{D})$ 
6  for  $j = 2, \dots, J$ 
7      REPLACE-NODE( $\hat{D}, r_j, r_1$ )
8  if INDEX( $\hat{D}$ ) = 1 return TRUE
9  else return FALSE

```

Given the last algorithm which solves CYCLE-FILTER problem (Lemma A.7), we can propose an upper bound of time complexity of this problem, which is done in the next result:

Theorem 5.7 *CYCLE-FILTER problem is solvable in time $\mathcal{O}(m \cdot |V|^4)$, where the instance*

of the problem is $(D, S) = ((V, A), S)$, and m is the number of blocks of S .

PROOF. Let $D = (V, A)$ be a strongly connected digraph, and S be a block-sequential update for D . The fact that algorithm $\text{TEST-CYCLE-FILTER}(D, S)$ solves the instance (D, S) of problem CYCLE-FILTER is proved on Lemma A.7. If we call \hat{A}^* to the set of arcs of \hat{D}^* , where \hat{D}^* refers to the set \hat{D} after pre-processing with NON-TRIVIAL-SCC , we see, by arguments said before, that $\text{INDEX}(\hat{D}^*)$, and each execution of $\text{REPLACE-NODE}(\hat{D}^*, r_j, r_1)$ is $\mathcal{O}(|\hat{A}^*|)$, which in turn is $\mathcal{O}(|V|^2)$. If we had in and out adjacency lists for every node of \hat{D}^* , each operation $\text{REPLACE-NODE}(\hat{D}^*, r_j, r_1)$ indeed would cost $\mathcal{O}(\deg_{\hat{D}^*}^-(r_j) + \deg_{\hat{D}^*}^+(r_j))$, then the total $(J - 2 + 1)$ $\text{REPLACE-NODE}(\hat{D}^*, r_j, r_1)$ operations would cost $\mathcal{O}(|\hat{A}^*|) = \mathcal{O}(|V|^2)$.

If we now refer to \hat{D} as this set before pre-processing with NON-TRIVIAL-SCC , and \hat{A} corresponding to its respective arc set, we see that $\text{NON-TRIVIAL-SCC}(\hat{D})$ requires to compute the s.c.c.'s of \hat{D} , which can be done (for example, by Tarjan's algorithm) in $\mathcal{O}(|\hat{A}| + |V|) = \mathcal{O}(|V|^2 + |V|)$. If the total number of s.c.c.'s of \hat{D} is NC , then discarding trivial s.c.c.'s from output set is $\mathcal{O}(NC) = \mathcal{O}(|V|)$, and erasing these trivial components from \hat{D} is time $\mathcal{O}(|\hat{A}|) = \mathcal{O}(|V|^2)$.

The tasks of computing induced subdigraphs (each one of subdigraphs \hat{D}_i and D_{prev}^i) takes time $\mathcal{O}(|A|)$. It only remains to see the time required by $\text{ADD-EDGES}(D, i, \hat{D}_i, S)$ in checking if there is a path from y to x in $D_{prev}^i = (V_i, A_i)$ (where $V_i = B_0 \cup \dots \cup B_{i-1}$ and $A_i \subseteq A$, $i = 0, \dots, m-1$). We need to estimate how many node pairs origin-destination (y, x) we have to check. If we realize that $(y, x) \in O^i \times D^i$ when it is executed $\text{ADD-EDGES}(D, i, \hat{D}_i, S)$, where

$$O^i = \bigcup_{v \in B_i} S^{out}(v) \quad \text{and} \quad D^i = \bigcup_{v \in B_i} S^{in}(v)$$

and bounding the cardinal of these sets as follows $|O^i| = \sum_{v \in B_i} |S^{out}(v)| \leq \sum_{v \in V} |S^{out}(v)| \leq |A|$ and analogously $|D^i| \leq |A|$, we conclude that the number of pairs to check is $|O^i \times D^i| = \mathcal{O}(|A|^2) = \mathcal{O}(|V|^4)$. Finally, we need to know the time required to check if there is a path in D_{prev}^i for each pair (y, x) . One alternative is to run a breadth-first-search with root y in $D_{prev}^i = (V_i, A_i)$, this method tells us what nodes in V_i are reachable from y , using an adjacency list representation of D_{prev}^i and taking time $\mathcal{O}(|V_i| + |A_i|)$. However, this search must be performed for each distinct origin y . Another alternative is to run firstly some all-pairs shortest paths algorithm (with weight equal to 1 for each arc) on D_{prev}^i inside $\text{ADD-EDGES}(D, i, \hat{D}_i, S)$, in this way checking if there is a path from y to x for each pair (y, x) takes time $\mathcal{O}(1)$. For example, Johnson's algorithm (see [18]) uses the adjacency-list representation of D_{prev}^i and takes time $\mathcal{O}(|V_i|^2 \lg |V_i| + |V_i| |A_i|)$. The Floyd-Warshall algorithm runs in time $\mathcal{O}(|V_i|^3)$ but using an adjacency matrix representation ([18]). Therefore, we conclude the asymptotic time cost of executing $\text{TEST-CYCLE-FILTER}(D, S)$, when using Johnson's algorithm, is bounded by the number of pairs (y, x) to check, giving a total time of $\mathcal{O}(m \cdot |V|^4)$, where m is the number of blocks of S . \square

5.4 Complexity of checking positive circuits

We finalize this section by studying the complexity of checking the condition required by Theorem 5 in [39]. In that paper, this issue was not addressed. The reader can safely skip this subsection without losing the main argument of this work. First, let us introduce the conventions used here. A Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is *increasing monotonic on input i* if

$$\forall \vec{x} \in \{0, 1\}^n, \quad f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \leq f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

and *decreasing monotonic on input i* if

$$\forall \vec{x} \in \{0, 1\}^n, \quad f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \geq f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

A Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is said to be a *sign-definite function*, if for each $i = 1, \dots, n$, it is either increasing monotonic or decreasing monotonic on input i . Now, let N_F be a Boolean network defined by a global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, where $F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$, and for each $i = 1, \dots, n$, $f_i : \{0, 1\}^n \rightarrow \{0, 1\}$. We say that F is a *sign-definite global function*, if for all $i = 1, \dots, n$, the local transition function f_i is a sign-definite function.

Given a sign-definite global function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, where $F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$, we denote by $I^+(j)$ and $I^-(j)$ the set of indices where f_j is increasing monotonic and decreasing monotonic respectively. Hence, for every Boolean network N_F such that F is a sign-definite global function, we can define a *sign function* $w_F : E_F \rightarrow \{-1, 1\}$ (E_F is the set of arcs of the digraph G^F associated to the network N_F), where

$$w_F(i, j) = \begin{cases} -1 & i \in I^-(j) \\ +1 & i \in I^+(j) \end{cases}$$

We shall say that an arc $(i, j) \in E_F$ is positive if $w_F(i, j) = 1$ and negative otherwise. We will say that a path is positive if the number of its negative arcs is even, and negative otherwise. The same will be said about closed paths, either simple closed paths (cycles, with intermediate nodes no repeated) or not necessarily simple closed paths (circuits). The pair (G^F, w_F) will be called *graph with sign* of N_F . We write now the theorem mentioned before:

Theorem 5.8 ([39]) *Let N_F be a Boolean network in which the global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ complies that all its simple circuits are positive. Let $N^{n-1} = \mathcal{S} \circ \dots \circ \mathcal{S}(N_F)$ be the network obtained from composing the operator \mathcal{S} $n - 1$ times (applied on N_F), where \mathcal{S} is the Gauss-Seidel operator associated to the sequential schedule $(1) \dots (n)$. Then, $N^{n-1} = \mathcal{S}(N^{n-1})$, and the only attractors of network N^{n-1} are fixed points, which are reached in at most n updates of network N^{n-1} .*

We see the alluded condition is that all the simple circuits in the digraph associated to the Boolean network are positive, and this condition is posed on the graph with sign defined before. This question can be put on any arbitrary sign function, and it is itself a decision problem. The complement of this problem can be formulated as follows:

NEGATIVE-CYCLE

Input: A directed graph $G = (V, E)$; a sign function $w : E \rightarrow \{-1, 1\}$

Question: Does the digraph G have a negative cycle (simple circuit)?

In [58], it is demonstrated that NEGATIVE-CYCLE can be solved in polynomial time. In that work, the problem is addressed in the context of sign solvability of linear systems (the problem is referred there as ODD-CYCLE). Here we will show an alternative proof of this fact, which makes use of results from graph databases, and that could mean a contribution in the sense it solves a problem with results from another field. Now we present the necessary definitions to state the next result. A *database graph* (or *db-graph*) $G = (V, E, \psi, \Sigma, \lambda)$ is a directed, labelled graph, where V is the set of nodes, E is the set of edges, and ψ is an *incidence function* mapping E to $V \times V$. Note that this function allows a db-graph to have multiple edges between a given pair of nodes, and there is no problem in this framework to represent graphs which there is only one arc connecting every pair of nodes. The labels of G (rather E) are drawn from Σ , which is a finite set of symbols called the *alphabet*, and λ is an *edge labelling function* mapping E to Σ .

Let $G = (V, E, \psi, \Sigma, \lambda)$ be a db-graph and $w = (v_1, e_1, \dots, e_{n-1}, v_n)$, where $v_i \in V, 1 \leq i \leq n$, and $e_j \in E, 1 \leq j \leq n-1$, be a walk (not necessarily a path) in G . We call the string $\lambda(e_1) \cdots \lambda(e_{n-1})$ the *walk label* of w , denoted by $\lambda(w) \in \Sigma^*$. Let R be a regular expression over Σ . We say that the walk w *satisfies* R if $\lambda(w) \in L(R)$, ie, the walk label $\lambda(w)$ belongs to the language associated with the regular expression R . The *query* Q_R on db-graph G is defined as the set of pairs (x, y) such that there is a walk from x to y in G which satisfies R . If $(x, y) \in Q_R(G)$, we say that (x, y) *satisfies* Q_R .

Now if we put the following decision problem

REGULAR-PATH

Input: A db-graph $G = (V, E, \psi, \Sigma, \lambda)$; nodes $x, y \in V$; a regular expression R over Σ

Question: Does G contain a directed walk $w = (e_1, \dots, e_k)$ from x to y such that w satisfies R , that is, $\lambda(e_1) \cdots \lambda(e_k) \in L(R)$?

it holds that REGULAR-PATH is indeed in P [70]. The name of the decision problem corresponds to the name in the original reference [70], where a “path” corresponds to a “walk” in our terminology. We have enough elements to prove the next result:

Proposition 5.9 NEGATIVE-CYCLE *is solvable in polynomial time*

PROOF. For an instance of the NEGATIVE-CYCLE problem $G = (V, E), w : E \rightarrow \{-, +\}$, we create an associated db-graph $G' = (V, E, \psi, \Sigma, \lambda)$, where ψ is simply the proper incidence function necessary to represent the connectivity of G , and $\Sigma = \{-, +\}$ (we take the sign function w as the labelling function). We take the regular expression $R' = +^* - +^*(+^* -$

$+^* - +^*)^*$ ”, whose associated language $L(R')$ consists of all strings with an odd quantity of “-”. For any pair of nodes $x, y \in V$, the question “is there a walk from x to y in G' which satisfies R' ?”, which we will denote as “Is $(x, y) \in Q_{R'}(G')$?”, as mentioned before is solvable in polynomial time. Then the following algorithm runs also in polynomial time (on the size of G):

NEGATIVE-CIRCUIT($G' = (V, E, \psi, \Sigma, w)$)

```

1  for  $v \in V$ 
2      if Is  $(v, v) \in Q_{R'}(G')$ ?
3          return TRUE
4  return FALSE

```

It is not difficult to see that NEGATIVE-CIRCUIT returns TRUE if and only if there is in G' a closed walk whose walk label has an odd quantity of “-”. The last part of the proof consists in realizing that a signed digraph G' has a negative closed walk if and only if it has a negative (simple) cycle (something that is no longer true when considering, for example, an even quantity of “-”). It is clear that a negative cycle is a negative walk. For the other direction, let $W_1 = (v_1, \dots, v_n)$ be a negative walk that is not a cycle. Then, there are at least two vertices repeated in W_1 . Choose i as large as possible such that $v_i = v_k$, for some $i < k \leq n$. By the election of i , $W_2 = (v_i, v_{i+1}, \dots, v_{k-1}, v_k)$ is a cycle (it has no repeated nodes). If W_2 is a negative cycle, the property holds. If W_2 is not negative, since an odd integer cannot be written as a sum of two even integers, then $W_3 = (v_1, \dots, v_{i-1}, v_{k+1}, \dots, v_n)$ is a negative walk smaller (in length) than W_1 . We can apply this reasoning again over W_3 , and repeating this procedure must finish producing a negative cycle since G' is finite. \square

Chapter 6

Filter convergence to fixed points

In this section, we address the problem of determine what structural properties of the input network guarantee the filter attractor is comprised of only one network, that is, when we can assure the filter attractor is a structural fixed point. Recall that Theorem 5.8 states that, with sequential schedules and positive circuits, the filter attractor is always a fixed point. This section is put after all the previous ones because the arguments hereinafter employ tools introduced on those sections.

6.1 Schedules of two blocks

Let D be a disjunctive network and s an update function of two instants or blocks $(B_\ell)_{\ell=0}^1$. In Section 4.2 we saw that the filter of two blocks can be analyzed through a linear system which involves the adjacency matrix of $D(B_0)^T$. As a first idea, we know the presence of several networks in the attractor relates with the presence of strongly connected components where the arcs (u, v) verifying $s(u) < s(v)$ can cycle with some periodicity $p > 1$, phenomena observed when we have imprimitive strongly connected components. Then, we could say that, for update schedules of two blocks, a sufficient condition to obtain a structural fixed point is that all the strongly connected components of $D(B_0)$ are primitive. Indeed, we can establish a less demanding condition to ensure the required property, what is done by the following result.

Proposition 6.1 *Let $D = (V, A)$ be a disjunctive network, and a block-sequential update schedule S for this network, with only two blocks B_0 and B_1 . Suppose that for any arc (i, j) with $i \in B_0$ and $j \in B_1$, the non-trivial strongly connected components in $D(B_0)^T$, the subdigraph of D induced by B_0 with reversed arcs, that are encountered first by walks (of $D(B_0)^T$) starting from vertex i are primitive. Then, $|\mathcal{A}(D, S)| = 1$, or in other words, the filter attractor is a structural fixed point.*

PROOF. Let s be the update function of schedule S . First suppose that D has no arcs (u, v) where $u \in B_0$ and $v \in B_1$, then the condition in the statement holds trivially in this case.

Since $s(u) \geq s(v)$ for every arc in D , GAUSS-SEIDEL adds to D^S every arc of D without changes and then returns. Therefore D is a structural fixed point reached after one execution of GAUSS-SEIDEL.

Assume now that there is an arc $(i, a) \in A$ such that $s(i) < s(a)$, meaning that $\vec{x}(a, D)_i = 1$ ($\vec{x}(a, D)$ is defined in Subsection 4.2). Let $M \in \mathcal{M}_{|B_0| \times |B_0|}(\{0, 1\})$ be the adjacency matrix of $D(B_0)^T$, the subdigraph induced in D by the first block B_0 , with inverted arcs, ie:

$$M_{ij} = 1 \iff (j, i) \in A$$

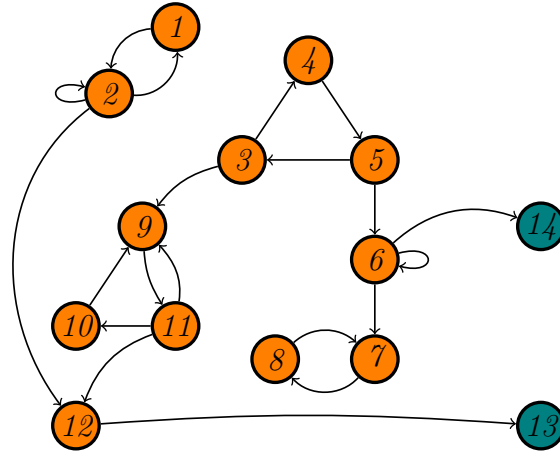
Then, by the Lemma 4.2, for every $n \in \mathbb{N}$, $\vec{x}(a, D^n) = \vec{x}(a, D) \cdot M^n$, where D^n is the graph obtained after n iterated executions of GAUSS-SEIDEL over D (with schedule S). From this equality and the Lemma 3.1 we deduce that

$$\vec{x}(a, D^n)_k = 1 \text{ iff } \vec{x}(a, D)_i = 1 \text{ and there is in } D(B_0)^T \text{ a walk of length } n \text{ from } i \text{ to } k \quad (6.1)$$

Let $(C_\ell)_{\ell=1}^r$ be the set of strongly connected components in $D(B_0)^T$ that are encountered first by walks starting from i . For every C_ℓ , it holds that in a number $n_\ell \leq |B_0|$ of executions of GAUSS-SEIDEL, $\vec{x}(a, D^{n_\ell})_{k_\ell} = 1$, for some vertex k_ℓ belonging to C_ℓ . By hypothesis, every component C_ℓ is primitive, then by Lemma 4.3 after a number n'_ℓ at most quadratic (on the size of the component, $|C_\ell|$) of executions of GAUSS-SEIDEL, it holds that $\vec{x}(a, D^{n_\ell+n'_\ell})_{k_\ell} = 1$, for every vertex k_ℓ belonging to C_ℓ . Since every C_ℓ is strongly connected, the last property is kept in ulterior executions. In particular, for those vertices $k'_\ell \in C_\ell$ from which there are walks starting from C_ℓ and carrying to other strongly connected components, it holds that $\vec{x}(a, D^h)_{k'_\ell} = 1$, for every $h \geq n_\ell + n'_\ell$.

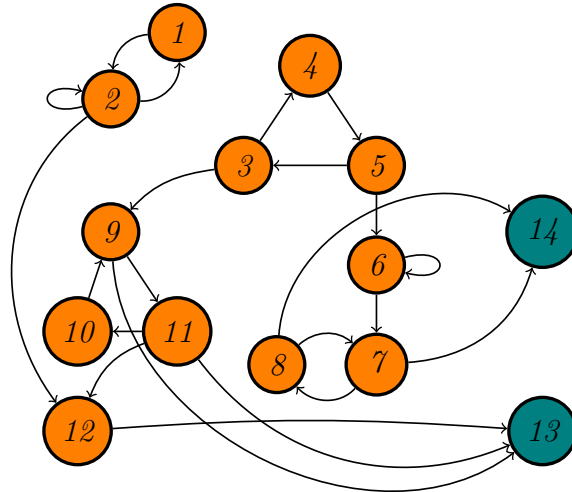
Now, for every node $v \in B_0$ reachable in $D(B_0)^T$ from C_ℓ , there is a walk from C_ℓ to v of length $n_v^\ell \leq |B_0|$, which implies by the equivalence (6.1) that $\vec{x}(a, D^{n_\ell+n_v^\ell+h})_v = 1$, for every $h \geq n_\ell + n'_\ell$ and for every C_ℓ . Finally, it holds that, after a number $\bar{n} = \max_{1 \leq \ell \leq r} (n_\ell + n'_\ell) + \max_{v \in B_0, 1 \leq \ell \leq r} n_v^\ell$, which is at most $\mathcal{O}(|B_0|^2)$, it holds that $\vec{x}(a, D^{\bar{n}})_v = 1$, or $(v, a) \in A(D^{\bar{n}})$, for every $v \in B_0$ reachable from any C_ℓ in $D(B_0)^T$. Since this is valid for every $(i, a) \in A$ such that $s(i) < s(a)$, the filter attractor consists in a structural fixed point. \square

Example 18 *Let D be the disjunctive Boolean network in the figure, and the block-sequential update schedule S (with update function s) of two blocks where the orange nodes (1 to 12) belong to the first block B_0 , and nodes 13 and 14 belong to the second block B_1 . It can be appreciated that the arcs (u, v) such that $s(u) < s(v)$ are the arcs $(6, 14)$ and $(12, 13)$. Both arcs verify the condition of the hypothesis: for the arc $(6, 14)$ the first non-trivial s.c.c. found first by walks starting from 6 in $D(B_0)^T$ is the component formed by the loop on vertex 6, which is primitive. In the other hand, the arc $(12, 13)$ complies the condition since the s.c.c.'s found first by walks starting from 12 in $D(B_0)^T$ are the two primitive components formed by 9, 10, 11, and 1, 2. It does not matter for the condition if the components formed by nodes 3, 4, 5, or 8, 7, are imprimitive. The filter on (D, S) converges after 7 iterations of GAUSS-SEIDEL to a structural fixed point.*



D

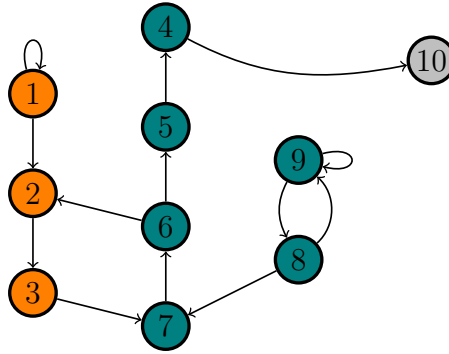
Remark 4 *The condition established by Proposition 6.1 is not necessary as the following example shows. Let D be the disjunctive Boolean network in the figure -which is very similar to that in Example 18-, and the block-sequential update schedule S equal to the schedule of previous example. It can be appreciated that the arcs (u, v) such that $s(u) < s(v)$ are the arcs $(7, 14)$, $(8, 14)$, $(9, 13)$, $(11, 13)$ and $(12, 13)$. None of these arcs verify the condition, the arcs incident to 14, have as the first s.c.c. encountered by walks starting from 7 or 8 in $D(B_0)^T$ to that formed by vertices 7 and 8, which is imprimitive. In the other hand, arcs $(9, 13)$, $(11, 13)$ have as their nearest s.c.c. in $D(B_0)^T$ to the imprimitive component comprised by 9, 10, 11; arc $(12, 13)$ fails to comply the condition by the same component (although the other component encountered first, that formed by vertices 1, 2, is primitive). However, after 4 applications of GAUSS-SEIDEL, the filter onto (D, S) converges to a structural fixed point.*



D

6.2 The general case

The general case for schedules of more than two blocks does not result to be a direct generalization of the case with two blocks. An example of this fact is given by the network D and schedule S in the image below. The schedule has three blocks, the first B_0 to be updated with (orange) nodes 1, 2, 3, the second B_1 to be updated with (blue) nodes 4, 5, 6, 7, 8, 9, and the third B_2 to be updated with the (gray) node 10.



D

We see there are two arcs (u, v) such that $s(u) < s(v)$: $(3, 7)$ and $(4, 10)$. We see also that the nearest non-trivial s.c.c. from node 3 in $D(B_0)^T$ is that formed by node 1, which is primitive, and the nearest non-trivial s.c.c. from node 4 in $D(B_1)^T$ is that formed by nodes 9, 8, which is also primitive, then a naive generalization of the condition holds in this example. However, after 2 executions of GAUSS-SEIDEL a new non-trivial imprimitive s.c.c. appears consisting in the cycle joining nodes 6, 7. Thus, after 2 executions of GAUSS-SEIDEL the arc $(4, 10)$ remains cycling at this new component, which finally leads to the filter applied to (D, S) , after 6 applications of GAUSS-SEIDEL, converges to a filter attractor of size 2.

With the previous example it seems we need to know the strongly connected components that will be appearing as GAUSS-SEIDEL is running, or how the existing components change. For this goal, the concept of induced length of a cycle seen in Section 5.1 turns useful again, since by Proposition 5.3, it informs us about the existence of cycles with certain length in the attractor networks.

Having said the above, the main result of this section is introduced, which gives a sufficient condition ensuring the filter attractor is a structural fixed point, for schedules of two or more blocks. This result is an inductive generalization of Proposition 6.1, and it makes use of Lemma A.9 from Section A.6 of the Appendix for the inductive step.

Theorem 6.2 *Let $D = (V, A)$ be a disjunctive network, and a block-sequential update schedule S for this network. Suppose that for any arc (i, j) with $s(i) = \ell_i < s(j) = \ell_j$, the non-trivial strongly connected components in $D(B_0 \cup B_1 \cup \dots \cup B_{\ell_j-1})^T$, the subdigraph of D induced by $B_0 \cup B_1 \cup \dots \cup B_{\ell_j-1}$ with inverted arcs, that are encountered first by walks (of $D(B_0 \cup B_1 \cup \dots \cup B_{\ell_j-1})^T$) starting from vertex i , satisfy the greatest common divisor*

of induced lengths of cycles is equal to 1. Then, $|\mathcal{A}(D, S)| = 1$, or in other words, the filter attractor is a structural fixed point.

PROOF. Let s be the update function of schedule S . We will refer to the network obtained after $\ell \in \mathbb{N}$ iterations of GAUSS-SEIDEL over network D and schedule S as D^ℓ . The proof is by induction on the quantity m of blocks of schedule S . The base case is when $m = 2$: here it happens that, since the cycles considered by the hypothesis are completely contained in $D(B_0)^T$, the induced length of these cycles match with their usual length, and the condition of the hypothesis is equivalent to the non-trivial strongly connected components are primitive. Thus, the base case is given by Proposition 6.1.

Suppose now the assertion is true for schedules of $m = k$ blocks, and for demonstrating that is also true for $m = k + 1$. Let $(B_j)_{j=0}^k$ be the set of blocks of a block-sequential update schedule S for network D having $k + 1$ update instants, which satisfies the condition of the hypothesis. Let S' be the schedule consisting in the restriction of S to the first k blocks, then the blocks of S' are $(B_j)_{j=0}^{k-1}$, and consider the subdigraph $D' = D(V')$ induced in D by the first k blocks of S ($V' = B_0 \cup \dots \cup B_{k-1}$). It holds that S' is a block-sequential schedule of k blocks for network D' which also satisfies the condition in the hypothesis, therefore, by inductive hypothesis, $|\mathcal{A}(D', S')| = 1$, and assume that $a \in \mathbb{N}$ is the smallest positive integer such that $D^a \in \mathcal{A}(D', S')$.

The last assertion implies, by Lemma A.9, that the filter applied onto (D, S) , from the $(a + 1) - th$ iteration and the following, only changes arcs starting from nodes in the first k blocks to nodes in the $(k + 1) - th$ block (since arcs in $D(V')$ are preserved as $\mathcal{A}(D(V'), S')$ is a structural fixed point). This allows to study the filter applied onto (D, S) , from the $(a + 1) - th$ iteration and the following, with the same construction of Lemma 4.2, using the schedule \tilde{S} of two blocks $(\tilde{B}_j)_{j=0}^1$: $\tilde{B}_0 = V'$ and $\tilde{B}_1 = B_k$ and the network D^a (the network obtained after a iterations of GAUSS-SEIDEL onto (D, S)). This construction simulates the evolution of the filter applied onto (D, S) from $(a + 1) - th$ iteration onwards.

The arcs (u, v) in D^a starting from \tilde{B}_0 to \tilde{B}_1 do not necessarily satisfy the condition in the hypothesis (if all of them satisfy it, the conclusion is direct from applying Proposition 6.1 to (D^a, \tilde{S})), but if this was the case -ie, there is a scc C_u whose induced lengths of cycles are not setwise coprime, which is reachable from u through a path in $D(V')^T$ without other non-trivial components-, the arc $(u, v) \in A(D^a)$ was diffused from an arc $(w, v) \in A(D)$ which indeed meets the requirement, that is, there is a walk in $D(V')^T$ from w to u , and the first non-trivial s.c.c. $C_{w,u} \in D(V')^T$ traversed by this walk complies the g.c.d. of induced lengths of its cycles is 1. This walk corresponds to a walk $W_{u,w}$ from u to w in $D(V')$.

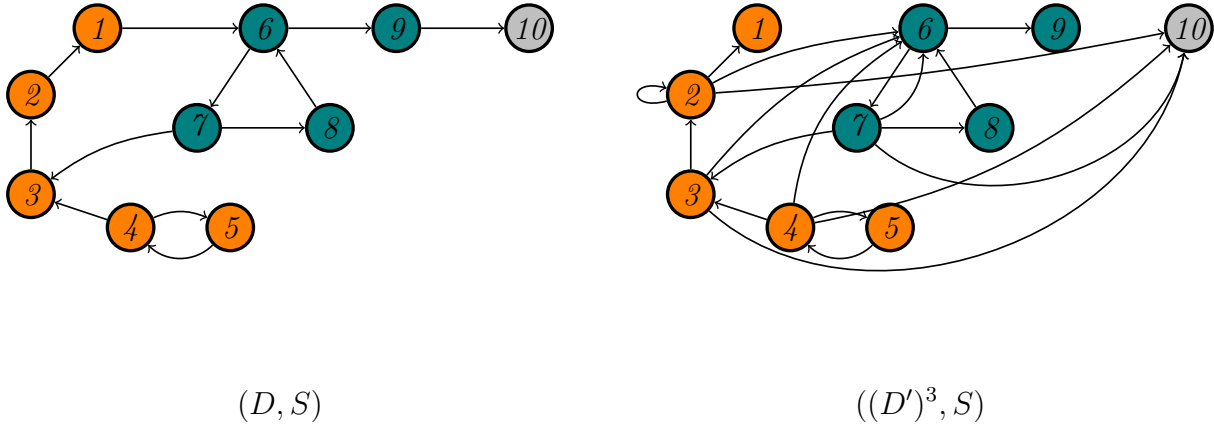
If every arc (a, b) of $W_{u,w}$ satisfies $s(a) \geq s(b)$, this walk is preserved in every application of GAUSS-SEIDEL, and necessarily $W_{u,w}$ is also a walk in D^a . Suppose there is an arc (a, b) of $W_{u,w}$ such that $s(a) < s(b) = \ell_b < k$, by the hypothesis, the first non-trivial strongly connected C_a of $D(B_0 \cup \dots \cup B_{\ell_b-1})^T$ that is reached verifies that the set of induced lengths of cycles is setwise coprime. Then, when $D(B_0 \cup \dots \cup B_{\ell_b-1})^T$ stabilizes as a structural fixed point (by the inductive hypothesis), by Proposition 5.3 C_a becomes a primitive component, and after an at most quadratic number of applications (Lemma 4.3) the procedure produces an arc (v, b) for each vertex $v \in C_a$. This shows that in network $D^a(V')$ necessarily there

is a fixed walk from u , and therefore from C_u , to w . We refer to the corresponding walk in $D^a(V')^T$ as $\widetilde{W}_{w,u}$.

In network D^a , the arc $(w, v) \in A(D)$ has produced some arc $(x, v) \in A(D^a)$, where x is a node from the component $\bar{C}_{w,u} \in D^a$, which is the component produced by $C_{w,u} \in D(V')^T$ (recall that by Lemma A.8, any strongly connected component produces at most one strongly connected at each execution of GAUSS-SEIDEL). By Proposition 5.3, $\bar{C}_{w,u}$ is primitive, therefore after at a number q at most quadratic of applications of GAUSS-SEIDEL, the procedure has produced an arc $(y, v) \in A(D^{a+q})$ for each vertex $y \in \bar{C}_{w,u}$. After this fact, in a number $\bar{n} \leq |V'|$, as in the proof of Proposition 6.1, it holds that $\vec{x}(v, D^{\bar{n}+a+q})_z = 1$, or $(z, v) \in A(D^{\bar{n}+a+q})$, for every $z \in \tilde{B}_0$ reachable in $D^a(\tilde{B}_0)^T$ from $\bar{C}_{w,u}$, in particular, for $z \in C_u$, thanks to walk $\widetilde{W}_{w,u}$.

Since this is valid for every $(u, v) \in A(D^a)$ such that $s(u) < s(v)$ (no matter if it does or does not meet the hypothesis, as it was shown), the filter attractor $\mathcal{A}(D^a, \tilde{S}) = \mathcal{A}(D, S)$ consists in a structural fixed point. This shows the statement of the theorem is valid for block-sequential update-schedules of $k + 1$ blocks, which concludes the proof. \square

Example 19 Let D be the disjunctive network of size 10 of the figure, and S be the block-sequential update schedule of 3 blocks $(B_\ell)_{\ell=0}^2$ for D , where the nodes of the first block are 1, 2, 3, 4, 5 (orange nodes), those on the second block are 6, 7, 8, 9, and the third block only has the node 10.



The arcs $(u, v) \in A(D)$ such that $s(u) < s(v)$ are $(9, 10)$ and $(1, 6)$. The first of them satisfies the hypothesis in Theorem 6.2, since the non-trivial strongly connected component in $D(B_0 \cup B_1)^T$ that is encountered first by walks starting from vertex 9, is that formed by nodes 6, 7, 8, 1, 2, 3, which has the cycle $(6, 7, 8)$ of induced length (with respect to S) 3 and the cycle $(6, 7, 3, 2, 1)$ of induced length 2. However, the arc $(1, 6)$ does not meet the hypothesis, since the non-trivial strongly connected component in $D(B_0)^T$ that is encountered first by walks starting from vertex 1, is that formed by nodes 4, 5, which has a greatest common divisor of induced lengths of cycles equal to 2. The filter applied onto (D, S) converges, after 7 iterations of GAUSS-SEIDEL, to an attractor of size 2.

If we now consider the network D' which consists of network D plus a loop on vertex 2, we see that (D', S) indeed meets the hypothesis, since now the non-trivial strongly connected

component in $D(B_0)^T$ that is encountered first by walks starting from vertex 1, is that formed by node 2, which complies the requirement. In terms of the proof of Theorem 6.2, in the network $(D')^3$ obtained after 3 iterations of GAUSS-SEIDEL, the subdigraph $(D')^3(B_0 \cup B_1)$ has stabilized into a structural fixed point, but for example the arc $(4, 10) \in A((D')^3)$, which was diffused from $(9, 10) \in A(D)$ through the walk $W_{4,9} = 4, 3, 2, 1, 6, 9$ in $D(B_0 \cup B_1)$, does not verify the hypothesis. That arc would finish cycling in the s.c.c. C_4 (reachable from vertex 4 in $(D')^3(B_0 \cup B_1)^T$) formed by nodes 4, 5; however the arc $(1, 6) \in W_{4,9}$ meets the hypothesis, since the first non-trivial strongly connected component C_1 of $D(B_0)^T$ that is reached from 1 is that formed by node 2, which is primitive, what assures that in network $(D')^3(B_0 \cup B_1)^T$ there is a walk $\widetilde{W}_{9,4} = 9, 6, 2, 3, 4$. This walk permits to diffuse arcs that ends at 10 and starts from the primitive component $\widetilde{C}_{9,4} \in (D')^3$ formed by vertices 2, 3, 6, 7, 8, what ultimately generates a fixed point in the component formed by vertices 4, 5 (and not a cycle of length 2). The filter applied onto (D', S) converges after 5 iterations of GAUSS-SEIDEL to a structural fixed point.

Conclusion

7.1 Main theorem

We review now a result that is a summary of all the development displayed in previous sections, and it can be considered as the main contribution of this work.

Theorem 7.1 *Let N_F be a Boolean disjunctive network of size n , where $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is its global transition function, D is the digraph associated to F , and S be a block-sequential update schedule for N_F . The next statements hold:*

- (i) *If the digraph associated to F_S is weakly connected, then, from any initial condition, updating according to S , the transient length is at most $\mathcal{O}(n^2)$.*
- (ii) *If the digraph associated to F is weakly connected and every strongly connected component has a loop or a complete subdigraph of order greater or equal than 3, then, for every network in $\mathcal{A}(N_F, S)$, updating according to the parallel schedule and starting from any initial condition, the transient length is at most $\mathcal{O}(n)$.*
- (iii) *If S has 2 blocks B_0 and B_1 , the digraph $D(B_0)$ is weakly connected and $|\mathcal{A}(N_F, S)| \in \mathcal{O}(n)$, the filter attractor $\mathcal{A}(N_F, S)$ can be computed in time at most $\mathcal{O}(n^7)$.*
- (iv) *If (N_F, S) is cycle-filter, which is checkable in polynomial time, then every network in $\mathcal{A}(N_F, S)$, updated in parallel, only has fixed points as attractors.*
- (v) *Suppose that for every arc (i, j) with $s(i) = \ell_i < s(j) = \ell_j$, it holds that the non-trivial strongly connected components in $D(B_0 \cup B_1 \cup \dots \cup B_{\ell_j-1})^T$, the subdigraph of D induced by $B_0 \cup B_1 \cup \dots \cup B_{\ell_j-1}$ with reversed arcs, that are encountered first by walks (of $D(B_0 \cup B_1 \cup \dots \cup B_{\ell_j-1})^T$) starting from vertex i , satisfy the greatest common divisor of induced lengths of cycles is equal to 1. Then, $|\mathcal{A}(D, S)| = 1$, or in other words, the filter attractor is a structural fixed point.*

PROOF. Assertion (i) corresponds to what is stated by Theorem 3.9. Assertion (iii), that is, the time necessary for computing $\mathcal{A}(N_F, S)$ is given by Proposition 4.5 and discussion in Subsection 4.3. The properties about cycle-filter condition -assertion (iv)- holds from Theorem 5.5, Theorem 5.7 and discussion in Subsection 5.3. The last assertion corresponds to the implicance given by Theorem 6.2, so (v) holds.

To see assertion (ii), let S a schedule with block set $(B_j)_{j=0}^{m-1}$, and let C be a strongly connected component of D , the digraph associated to F , having a loop. It is apparent, by definition of algorithm GAUSS-SEIDEL, that the loops are preserved in every application of this procedure, then, the s.c.c. \bar{C} generated by C (recall that by Lemma A.8 and Lemma

A.9, C generates at most one s.c.c. \bar{C} in the networks in $\mathcal{A}(N_F, S)$ in every network in $\mathcal{A}(N_F, S)$ has the same loop of C . Suppose now C has a complete subdigraph W of order $k \geq 3$. If the nodes of W are updated at the same time according to S , W is preserved and the s.c.c. \bar{C} generated by C in every network in $\mathcal{A}(N_F, S)$ has the same complete subdigraph W . If the nodes of W are updated at different times (or belong to different blocks), let $t_l \in \{0, \dots, m-2\}$ be the smallest time at which some node of W is updated, and let $a \in V \cap W$ be any node of W that does not belong to block B_{t_l} . Since W is complete, we know there is cycle of length 2 formed by nodes a and some node $b \in B_{t_l}$. After applying Gauss-Seidel operator, a loop is added to node a , and this loop is preserved in ulterior applications. Therefore, we see that if W has $k' < k$ nodes belonging to blocks different from block B_{t_l} , then the s.c.c. \bar{C} generated by C in every network in $\mathcal{A}(N_F, S)$ has at least k' loops. We conclude that the initial assumption implies that every s.c.c. of the digraph of any network in $\mathcal{A}(N_F, S)$, has a loop or a complete subdigraph of order greater or equal than 3, then, by Theorem 3.12, for every network in $\mathcal{A}(N_F, S)$, starting from any initial condition, in at most $\mathcal{O}(n)$ updates according to the parallel schedule, the state vector gets to some fixed point \square

7.2 Applications

Theorem 7.1 enumerates a list of facts that may be very useful to propose a general filtering algorithm, that is, given a disjunctive Boolean network N , the algorithm could find a block-sequential update schedule σ optimal in some sense (eg, minimal number of blocks) that guarantees networks in $\mathcal{A}(N, \sigma)$ indeed do not have limit cycles with the parallel schedule, and such that $\mathcal{A}(N, \sigma)$ can be efficiently computed. Also it could have the nice property that the filter attractor is a structural fixed point. Theorem 7.1 allows us to note that, when we restrict to disjunctive networks, the complexity of computing the filter by iterating the Gauss-Seidel operator beside related problems, remains polynomial under reasonable assumptions, which suggests that this technology may be feasible to implement in practical applications (and to work even with large networks). Theorem 4.6 also gives intuition of when the filter attractor $\mathcal{A}(N, \sigma)$ acquires a size that turns impractical to manage.

Disjunctive Boolean networks do not have the negation connective in their Boolean functions, then they trivially meet that all their simple circuits are positive. Then, Theorem 5.8 from the paper of Goles and Salinas (2010) ensures that, for a disjunctive network and sequential schedule (D, S) , the networks in the filter attractor $\mathcal{A}(D, S)$, when updated with parallel schedule, do not have limit cycles. This fact can be seen as an application of Theorem 5.5. Let C be a non-trivial strongly connected component of digraph D contained in just one updating block. Since S is sequential, C is comprised of only one node with loop, then C necessarily is primitive. Now, let C be a non-trivial strongly connected component of digraph D belonging to more than one updating block induced by schedule S . Since S is sequential, there is only one node in the last block of those containing component C . Therefore, necessarily there is in C a cycle whose induced length equals 1. Hence, we see that, for every disjunctive network D and sequential schedule S , the pair (D, S) is Cycle-filter, and Theorem 5.5 allows to conclude that the networks in the filter attractor $\mathcal{A}(D, S)$, when updated with parallel schedule, do not have limit cycles. In the other hand, the other statement

of Theorem 5.8 claiming that, for a disjunctive network and sequential schedule (D, S) , it holds that $|\mathcal{A}(D, S)| = 1$, can be derived analogously from Theorem 6.2.

It is interesting, from a theoretical standpoint, that these properties can be derived from results from linear algebra and studying exclusively the topology of the associated digraph of the network, which is a simple representation of these networks. The fact that dynamical evolution of network states, studied in Subsection 3.2, and the stabilization of some operator applied onto the networks, studied in Section 4, could be studied using the same theoretical results may be an indication of some kind of deeper relationship between these phenomena.

7.3 Outlook and future work

Results from this work confirm that capacity from iterated Gauss-Seidel of removing limit cycles, remains when we extend to block-sequential schedules, at least in certain types of networks. Disjunctive networks are an example of nested canalizing functions [53] [66], which exhibit more robust dynamics than random networks, that is, less attractors and shorter limit cycles, and the majority of regulatory functions in many published models of gene regulatory and signaling networks are nested canalizing [75]. All the more, nested canalizing functions are a subclass of canalizing functions [55] [52], in which at least one of the input variables is able to determine the function output regardless of other values of other variables. This concept evokes the concept of “canalization” introduced by Waddington [89] to refer to the ability of a genotype to develop the same phenotype regardless variations on environmental conditions. In the other hand, there is evidence that canalizing functions are common in higher vertebrate gene regulatory systems [55], and a large-scale study of the literature on transcriptional regulation in eukaryotes demonstrated a bias towards canalizing rules [41].

A potential application of the dynamics in disjunctive networks is to interpret it as the dynamics of some network epidemic model [26]. The simplest epidemic model is the SI model [10], in which there are two states a node can adopt: before the node has caught the disease, it is susceptible, and once it has caught the disease, it becomes infectious. Infectious nodes has some contagion probability of infecting each of its susceptible neighbors. Thus, the dynamics of some disjunctive network can be interpreted as the dynamics of a SI epidemic model over the same digraph (where the nodes with state equal to 0 are the susceptible ones), with contagion probability equal to one. Attractors, particularly fixed points, of these networks can provide information of the stationary state obtained from a given initial condition of the network (or initial distribution of infected nodes), for diseases with high contagion probability. Last paragraphs suggest that disjunctive networks may not only be a blackboard toy, but also a functional modeling tool in itself, despite its conceptual simplicity.

As a conjecture for future work remains the idea of extending the statement of Theorem 3.9 for every possible disjunctive Boolean network, not only those where the associated digraph is weakly connected. The argument is the following. If we have a digraph with several weakly connected components (or disconnected components, if we see the underlying undirected graph), Theorem 3.9 is valid for each of these components, therefore the transient of each one is quadratic on the size of each respective component. As the different components

evolve in parallel with respect to other components, the transient length of the complete network is simply the maximum over the set of transient lengths of all the components, and this maximum would still be quadratic on the size of the whole network. Since all the other results requiring the hypothesis of the weakly connected digraph are based on Theorem 3.9, whether the conjecture is true, all these other results would drop this hypothesis too.

A natural question, motivating a possible line of future research, is to ask what about more expressive networks, if it is possible to extend the results and techniques developed in this work to more general networks. A potential tool to realize this generalization is that of binary decision diagrams (BDDs) [63] [1] [74], one of the most used data structure for representation of Boolean functions. The restricted class of reduced ordered BDDs (OBDDs) obtained extended usage because these diagrams provide a canonical representation and allow efficient manipulation [14]. For a set of Boolean functions, an OBDD representing the functions in this set consists in a directed acyclic graph, that offers a description for the computation of a Boolean function. The computation of Gauss-Seidel operator may be formulated in terms of some basic operations on Boolean functions that can be executed efficiently on OBDDs, as Equivalence tests and Substitution [22]. It should be noted that if superpolynomial behaviors (Theorem 4.6) are observed when the network has only the disjunctive connective, intractable magnitudes are likely to appear with more expressive networks.

Another potential line of research is a possible interpretation of the filtering procedure in terms of biological meaning or other considerations. Notice conditions that guarantee the removal of limit cycles relate to the presence of circuits in the graph associated to the network, meeting certain assumptions. Thomas [79] defined the regulatory circuits as simple circular chains of oriented interactions, which can be positive or negative depending on the parity of the number of negative interactions (analogously to definitions in Subsection 5.4), and they would necessary to generate non-trivial dynamical behaviour. In particular, positive circuits would be necessary to generate alternative cellular states -or multiple attractors-, while negative circuits would be necessary to generate homeostasis or oscillatory behaviour [85]. Several efforts have been made to mathematically formalize these rules [80] [83] [25] [56]. Other idea related to the meaning of cycles in a Boolean (or qualitative) network model derived from a differential equation model is the Glass-Kauffman hypothesis [33] [31]. Assuming the system modeled has a negative regulatory circuit, this principle can be summarized as follows: if the Boolean network has no limit cycles, the differential equation system does not have limit cycles; if the Boolean network has limit cycles, the differential equation model may or may not have limit cycles. A mathematical formulation and proof of this hypothesis has been given by Glass and Pasternack [34] [32] for piece-wise linear systems.

Other related topics and tools to study filters are, for example, the representation of the problem as a query on a database. Proof of proposition 5.9 already suggested a potential connection with database theory. The output of the filter could be seen as the result of a query (fact inference) formulated in a logic programming language (such as Datalog, or Prolog) over a set of facts (the initial network), where the output is computed by least fixpoint iteration: some operator is iterated until one term is equal to its predecessor [15]. The problem of requiring the output of the filter only has fixed points may be relaxed by imposing only a subconfiguration of states remains fixed, regardless of the values of the other nodes, which is known as an alliance, a concept introduced by Goles, Montalva & Ruz (2012) [36]. The

problem of characterizing the whole family of outputs of the filter on a given network can be addressed by employing the equivalence classes that group schedules producing the same dynamics developed by Aracena et al [5] [6]. Finally, the relevance of the usual complexity measures employed in computer science should be evaluated in the context of gene regulation: due to current technology restrictions, the size of networks used in real modeling applications does not increase asymptotically, and maybe the difficulties encountered when working with these networks are not captured by worst case analysis.

Appendix A

Proofs

A.1 Proofs for Section 1

Lemma A.1 *Let N_F be a Boolean network defined by a global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$, with component functions f_i ($F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$), and let S be a block-sequential update schedule for this network. Then it holds that the component functions f_i^S of F_S , the global transition function of network N_F with schedule S , can be characterized as follows:*

$$\forall \vec{x} \in \{0, 1\}^n, f_i^S(\vec{x}) = f_i(x_1^i(\vec{x}), \dots, x_n^i(\vec{x})) \text{ where } x_j^i(\vec{x}) = \begin{cases} x_j & s(i) \leq s(j), \\ f_j^S(\vec{x}) & s(i) > s(j) \end{cases}$$

PROOF. Let S be a block-sequential update schedule of m blocks $(B_j)_{j=0}^{m-1}$ with update function $s : V \rightarrow \{0, \dots, m-1\}$, and define for each $k \in \{0, \dots, m-1\}$, the following vectors (whose value depends on any vector $\vec{x} \in \{0, 1\}^n$):

$$\vec{y}_k = F_{[B_k]} \circ \dots \circ F_{[B_1]} \circ F_{[B_0]}(\vec{x})$$

It is clear that $\vec{y}_{m-1} = F_S(\vec{x})$, and from definition of $F_{[W]}$ (for a set W of nodes updated in parallel) it holds that each j -th component of \vec{y}_k is given by

$$(\vec{y}_k)_j = \begin{cases} f_j(\vec{y}_{k-1}) & j \in B_k, \\ (\vec{y}_{k-1})_j & \sim \end{cases} \text{ for } k \geq 1$$

$$(\vec{y}_0)_j = \begin{cases} f_j(\vec{x}) & j \in B_0, \\ \vec{x}_j & \sim \end{cases}$$

In a block-sequential schedule S , each node j is updated exactly once. This only date

within an update sequence, corresponds for each node j to time $t_j = s(j) \in \{0, \dots, m-1\}$. Considering the values taken by different components $(\vec{y}_k)_j$, then it holds that

$$(\vec{y}_k)_j = \begin{cases} (\vec{y}_{k-1})_j = \dots = (\vec{y}_1)_j = (\vec{y}_0)_j = x_j & 0 \leq k < s(j), \\ (\vec{y}_{k-1})_j = \dots = (\vec{y}_{s(j)+1})_j = (\vec{y}_{s(j)})_j = f_j(\vec{y}_{s(j)-1}) & s(j) \leq k \leq m-1 \end{cases}$$

Thus, it is obtained that $F_S(\vec{x})_i = (\vec{y}_{m-1})_i = f_i(\vec{y}_{s(i)-1})$, and the argument of $f_i(\cdot)$, $\vec{y}_{s(i)-1}$, corresponds, according to the above, to the following

$$(\vec{y}_{s(i)-1})_j = \begin{cases} x_j & s(i) - 1 < s(j), \\ f_j(\vec{y}_{s(j)-1}) & s(i) - 1 \geq s(j) \end{cases} = \begin{cases} x_j & s(i) \leq s(j), \\ f_j(\vec{y}_{s(j)-1}) & s(i) > s(j) \end{cases}$$

In this way, it is deduced that $f_i(\vec{y}_{s(i)-1})$ and $(\vec{y}_{s(i)-1})_j$, correspond respectively to $f_i^S(\vec{x})$ and to $x_j^i(\vec{x})$, which proves the assertion of the lemma. □

A.2 Proofs for Section 2

PROOF OF LEMMA 2.1. Let AG be the digraph of order n associated to the Boolean network N_G . Since N_G is a conjunctive network, each i -th local transition function is given by $g_i(\vec{x}) = \bigwedge_{j \in N_{AG}^-(i)} x_j$, except maybe for those nodes $k \in V$ whose local transition function is constant (where $N_{AG}^-(k) = \phi$). Let define the functions of the disjunctive network N_F so that the digraph associated to N_F is the same for N_G , ie:

$$f_i(\vec{x}) = \bigvee_{j \in N_{AG}^-(i)} x_j, \quad \text{for all } i \in V \text{ such that } N_{AG}^-(i) \neq \phi$$

$$f_i(\vec{x}) = \neg g_i(\vec{x}), \quad \text{for all } i \in V \text{ such that } N_{AG}^-(i) = \phi$$

With global transition function $F(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x}))$ of network N_F defined as above, it suffices to prove the assertion of the lemma for parallel schedule π , since the dynamics of any transition function F of a disjunctive network with schedule S is simulatable using transition function F_S and parallel schedule π , where F_S is also disjunctive (thanks to the fact F is disjunctive and the definition of F_S). So, if $i \in V$ verifies $N_{AG}^-(i) \neq \phi$ then:

$$f_i^\pi(\vec{x}) = f_i(\vec{x}) = \bigvee_{j \in N_{AG}^-(i)} x_j = \bigvee_{j \in N_{AG}^-(i)} \neg \neg x_j = \neg \bigwedge_{j \in N_{AG}^-(i)} \neg x_j = \neg g_i(\neg \vec{x}) = \neg g_i^\pi(\neg \vec{x})$$

If $i \in V$ is a node with constant value, or such that $N_{AG}^-(i) = \phi$, then $f_i^\pi(\vec{x}) = f_i(\vec{x}) = \neg g_i(\vec{x}) = \neg g_i(\neg \vec{x}) = \neg g_i^\pi(\neg \vec{x})$, which concludes the proof. □

A.3 Proofs for Section 3

PROOF OF LEMMA 3.10. Let ℓ be the length of the shortest circuit C_ℓ in D . In the following we refer to the digraph associated with matrix M^ℓ as D^ℓ . First we show that the strongly connected components of D^ℓ correspond exactly to the imprimitivity sets $(V_i)_{i=1}^{k(D)}$ (Lemma 3.4). Let S be a strongly connected component of D^ℓ , and let $p, q \in S$ two nodes from D (not necessarily distinct), then there is a walk $W_{p,q}$ in D^ℓ of length $|W_{p,q}| = H$. By Lemma 3.1, there exists a walk $\tilde{W}_{p,q}$ in D of length $|\tilde{W}_{p,q}| = H \cdot \ell$, and since ℓ is multiple of k , $|\tilde{W}_{p,q}| \equiv_{k(D)} 0$. Let $a \in V$ be any arbitrary fixed node of digraph D . If we examine the definition of the imprimitivity sets (3.1), we see that, if $p \in V_i$ (recall that $(V_i)_{i=1}^{k(D)}$ is a partition of V), there is a walk in D from a to p , $W_{a,p}$, of length $|W_{a,p}| \equiv_{k(D)} i$. If we add this walk with $\tilde{W}_{p,q}$, we obtain a walk in D from a to q , $W_{a,q}$, of length $|W_{a,q}| \equiv_{k(D)} i$, and thereby, necessarily q belongs to V_i . In other words, for every strongly connected component S from D^ℓ , there exists $i \in \{1, \dots, k(D)\}$ such that $S \subseteq V_i$.

We now show the other inclusion: let V_i ($1 \leq i \leq k(D)$) be a set as defined in Lemma 3.4, we claim this set is a strongly connected component of D^ℓ . To prove this, we first shall see that there is a circuit C in D^ℓ that contains every node in V_i (at least once). Since D is strongly connected, there is a closed directed walk \mathcal{C} that contains every node in V . To construct C , we take a node $x_i \in V_i$, and move within the circuit \mathcal{C} thrusting forward ℓ steps each time. Each node encountered with this method belongs to V_i as each of these nodes is at a distance from x_i (in the circuit \mathcal{C}) that is a multiple of ℓ , and hence, multiple of $k(D)$. Notice that with this procedure, a closed walk (starting and ending on x_i) is obtained after $lcm(\ell, |\mathcal{C}|)/\ell$ iterations (of thrusting forward ℓ steps). We claim that this closed walk contains all the nodes in V_i . Suppose there is a node $x_j \in V_i$ that is not reachable from x_i with the procedure mentioned above, ergo the walk in \mathcal{C} that starts on x_i and ends on x_j , W_{x_i, x_j} , has a length not multiple of ℓ : $|W_{x_i, x_j}| \equiv_{\ell} N \neq 0$, but by definition of congruence modulo ℓ , there exists $z \in \mathbb{Z}$ satisfying that

$$|W_{x_i, x_j}| = z \cdot \ell + N = z \cdot \frac{\ell}{k(D)} \cdot k(D) + N \iff |W_{x_i, x_j}| \equiv_{k(D)} N \neq 0$$

and this contradicts (iv) from Lemma 3.4. Thereby this closed walk in D contains every node in V_i separated by ℓ steps, and this means that in D^ℓ there is a circuit C containing every node in V_i . This is equivalent to the subgraph induced in D^ℓ by V_i , $D^\ell(V_i)$, is strongly connected. It remains to prove that the set V_i is a maximal strongly connected component in D^ℓ : assume there is a node $x_r \in V_r$ ($r \neq i$) such that $V_i \cup x_r$ is a strongly connected component in D^ℓ . This obviously contradicts the proven assertion in the previous paragraph, and thus we have proved that V_i is a strongly connected component of D^ℓ .

Finally, to conclude that every strongly connected component in D^ℓ is primitive, notice

that as every arc in D starts from a node in V_i and ends on a node in V_{i+1} , for some i , $1 \leq i \leq k(D)$ (according to (iii) in Lemma 3.4), and as there are $k(D)$ sets V_i and ℓ nodes in the shortest circuit C_ℓ ($k(D) \leq \ell$), necessarily for each $i \in \{1, \dots, k(D)\}$ there is at least one node in $C_\ell \cap V_i$. But each node belonging to the circuit C_ℓ has a loop in D^ℓ , then by the above each strongly connected component in D^ℓ has at least one node with a loop and therefore is primitive. \square

The proof of the next result can be reviewed at Andrews (Corollary 2-2 [3]):

Lemma A.2 (Bezout's Identity) *In order that there exist integers x and y satisfying the equation*

$$ax + by = c$$

it is necessary and sufficient that $d|c$, where $d = \gcd(a, b)$.

A.4 Proofs for Section 4

Lemma A.3 *The algorithm GAUSS-SEIDEL, which receives as input a disjunctive network D and a block-sequential update schedule S , is correct, meaning that the network returned corresponds to the network outputted by Gauss-Seidel operator associated to schedule S applied onto D .*

PROOF. It is enough to prove that the set of arcs A' returned by GAUSS-SEIDEL is exactly A^S , the set of arcs of the network D^S outputted by Gauss-Seidel operator. For this we shall use the characterization of A^S already proved in Lemma 2.2 (where s is the update function of schedule S):

$$A^S = \{(j, i) \mid \text{there is in } D \text{ a path } \{v_0, v_1, \dots, v_l\} \text{ from } j = v_0 \\ \text{to } i = v_l \text{ such that } s(v_0) \geq s(v_1) \wedge \forall 1 \leq k < l, s(v_k) < s(v_{k+1})\}$$

For readability we put here the pseudo-code of GAUSS-SEIDEL:

GAUSS-SEIDEL($D = (V, A), S$)

```

1   $A' = \phi, (A' \subseteq A)$  // initialization
2   $I = \phi, (I \subseteq V)$ 
   Let  $B_0, \dots, B_{m-1}$  be the blocks of  $S$ , and let  $s$  be the update function
3  for  $i = 0$  to  $m - 1$ 
4      for  $j \in B_i$ 
5          for  $(u, j) \in A$  // we go over every arc incident to node  $j$ 
6              if  $s(u) \geq s(j)$ 
7                   $A' = A' \cup \{(u, j)\}$ 
8              else
9                  for  $(t, u) \in A'$ 
10                      $A' = A' \cup \{(t, j)\}$ 
11          $I = I \cup \{j\}$ 
12 return  $(I, A')$ 

```

Let $e = (u, v) \in A'$ be an arc in the network returned by GAUSS-SEIDEL. This arc was added to A' in line 7 or line 10 of the algorithm. Suppose it was added in line 7: in this case $e = (u, v) \in A$ and $s(u) \geq s(v)$, so we can take $v_0 = u, v_1 = v_l = v$ and hence, $e = (u, v) \in A^S$. Suppose now it was added in line 10: in this situation we know there is a node w such that, $(u, w) \in A'$ and $(w, v) \in A$, with $s(w) < s(v)$. The same previous reasoning can be applied to arc (u, w) : it was added to A' in line 7 or line 10. If it was added in line 7, we deduce that $(u, w) \in A$ and $s(u) \geq s(w)$, therefore, we can take $v_0 = u, v_1 = w, v_l = v (l = 2)$, and thus (u, v) verifies the condition to be in A^S . If (u, w) was added in line 10, the previous reasoning can be repeated (since A' is finite the reasoning cannot be repeated indefinitely) until obtaining a path in A from u to v meeting all the conditions such that $(u, v) \in A^S$.

Now let $(u, v) \in A^S$ be an arc in the network D^S calculated by Gauss-Seidel operator, by characterization there is in D a path v_0, v_1, \dots, v_l from $u = v_0$ to $v = v_l$ such that $s(v_0) \geq s(v_1) \wedge \forall 1 \leq k < l, s(v_k) < s(v_{k+1})$. Recall the network $D = (V, A)$ is the input received by GAUSS-SEIDEL, returning the network (V, A') . By GAUSS-SEIDEL definition, the arc (u, v_1) is added to A' in line 7, and the other arcs are added in line 10. All the more, we know nodes v_k where $1 \leq k \leq l$ are added to I in increasing order on k (due to the order in which the blocks are revised by the algorithm), which in turn implies that arcs (v_{k-1}, v_k) , where $1 \leq k \leq l$, are also added to I in increasing order on k . In this manner, the first arc from the path that connects u with v to be added to A' is (u, v_1) in line 7. Later, every arc (v_{k-1}, v_k) , with $2 \leq k \leq l$, is processed in line 10: the algorithm searches in A' all arcs of the form (t, v_{k-1}) , and adds to A' the arc (t, v_k) . When $k = 2$, since the arc (u, v_1) is already in A' , the arc (u, v_2) is added to A' . When $k = 3$, since (u, v_2) is in A' , the arc (u, v_3) is added to A' , and so are added iteratively to A' the arcs (u, v_k) until $k = l$. As $v_l = v$, in the last block of the schedule the arc (u, v) is added to A' : thus, $(u, v) \in A'$. \square

PROOF OF LEMMA 4.2. Let s be the update function of schedule S . Suppose it is not true, that there exists $i \in B_0$ such that $(\vec{x}(a, D) \cdot M)_i = 1$ and $x(a, D^S)_i = 0$. Assume the arcs in A whose starting node is i are the following:

$$\{(i, j)\}_{j \in H}, \quad H \subseteq V$$

Since $(\vec{x}(a, D) \cdot M)_i = 1$, necessarily there exists $j \in H \cap B_0$ satisfying $x(a, D)_j = 1$. This is because if it was that $\forall k \in H \cap B_0, x(a, D)_k = 0$, it would hold that

$$(\vec{x}(a, D) \cdot M)_i = \sum_{k \in B_0} x(a, D)_k M_{ki} = \sum_{k \in B_0 \cap H} \underbrace{x(a, D)_k}_{=0} M_{ki} + \sum_{k \in B_0/H} x(a, D)_k \underbrace{M_{ki}}_{=0} = 0$$

which would be absurd. Since there exists $j \in H \cap B_0$ satisfying $x(a, D)_j = 1$, from definition of $\vec{x}(a, D)$, we know $(j, a) \in A \wedge s(j) < s(a)$. Observe since $j \in H$, this means $(i, j) \in A$. When GAUSS-SEIDEL is applied over D and S , the first block processed is B_0 , implying the arc (i, j) is processed (when the node j is processed) before the arc (j, a) (which is processed when the node a is processed). From what was said before: $s(i) = s(j) < s(a)$, then (i, j) is processed at line 7 from GAUSS-SEIDEL, and (j, a) is processed at line 10: thus, the arc (i, j) is added to A' , and since $(i, j) \in A'$ when (j, a) is processed, then the arc (i, a) is added to A' . With this, $x(a, D^S)_i = 1$ which contradicts the initial assumption.

Suppose now there exists $i \in B_0$ such that $(\vec{x}(a, D) \cdot M)_i = 0$ and $x(a, D^S)_i = 1$. Since $x(a, D^S)_i = 1$, it holds (Lemma A.3) that $(i, a) \in A'(s(i) < s(a))$. At this point we can use the characterization proved in Lemma 2.2 to state that there is in D a path (v_0, v_1, \dots, v_l) from $i = v_0$ to $a = v_l$ such that:

$$s(v_0) \geq s(v_1) \tag{A.1}$$

$$\forall 1 \leq k < l, s(v_k) < s(v_{k+1}) \tag{A.2}$$

Since there are only two blocks, from first condition we see $v_1 \in B_0$ (and therefore, it is different from a), and from second condition, it is deduced $l = 2$. With this we have $(i, v_1) \in A$, meaning that $M_{v_1, i} = 1$. Now, since $(v_1, a) \in A$ y $s(v_1) < s(a)$, this allows to say that $x(a, D)_{v_1} = 1$. Considering the foregoing:

$$(\vec{x}(a, D) \cdot M)_i = \sum_{k \in B_0} x(a, D)_k M_{ki} \geq x(a, D)_{v_1} M_{v_1, i} = 1$$

which is absurd. □

PROOF OF LEMMA 4.3. Let s be the update function of schedule S . First suppose that D has no arcs (u, v) where $u \in B_0$ and $v \in B_1$. Since $s(u) \geq s(v)$ for every arc in D , GAUSS-SEIDEL adds to D^S every arc of D without changes and then returns. Therefore D is a structural fixed point reached after one execution of GAUSS-SEIDEL.

Assume now that there is an arc $(i, a) \in A$ such that $s(i) < s(a)$, meaning that the vector $\vec{x}(a, D)$ is not null. Let $M \in \mathcal{M}_{q \times q}(\{0, 1\})$ be the adjacency matrix of $D(B_0)^T$, the subdigraph induced in D by the first block with reversed arcs, ie:

$$M_{ij} = 1 \iff (j, i) \in A$$

Then, by Lemma 4.2, for every $n \in \mathbb{N}$, $\vec{x}(a, D^n) = \vec{x}(a, D) \cdot M^n$, where D^n is the digraph obtained after n iterated executions of GAUSS-SEIDEL over D (with schedule S). By Theorem 3.2, M is irreducible and primitive, and Theorem 3.5 guarantees that M^n is positive for all $n \geq (q-1)^2 + 1$, and since $\vec{x}(a, D)$ is not null, $\vec{x}(a, D^n)$ is positive for all $n \geq (q-1)^2 + 1$. The above argument says us that every arc $(i, a) \in A$ such that $s(i) < s(a)$, in at most $n \in \mathcal{O}(q^2)$ executions of GAUSS-SEIDEL, generates in the digraph D^n q arcs (l, a) for every $l \in B_0$, and this is kept in D^m , with $m \geq n$.

Arcs connecting nodes in the same block are always preserved by GAUSS-SEIDEL. Assume that there is an arc $(j, k) \in A$ such that $s(k) < s(j)$ -this arc (j, k) is preserved without changes by GAUSS-SEIDEL- and an arc $(i, a) \in A$ such that $s(i) < s(a)$. Since $D(B_0)$ is strongly connected there is a directed path in $D(B_0)$ from k to i of length $p \in \mathcal{O}(q)$. By Lemma 3.1, $M_{i,k}^p > 0$, and given that $x(a, D)_i > 0$, from Lemma 4.2 it follows that $x(a, D^p)_k > 0$, namely there is in D^p an arc (k, a) , where $s(k) < s(a)$. By the definition of GAUSS-SEIDEL algorithm, in D^{p+1} is added the arc (j, a) , where $s(j) = s(a) = 1$, meaning that GAUSS-SEIDEL is adding an arc that connects nodes belonging to the second block B_1 . These arcs are not described by the vectors $\vec{x}(a, D)$ (because the components of these vectors depict nodes in the first block), but what matters here is that these arcs are added in the iteration $p + 1 \in \mathcal{O}(q)$ of GAUSS-SEIDEL, and these arcs remain in the digraphs D^m , with $m \geq p + 1$.

The foregoing demonstrates that in at most $n \in \mathcal{O}(q^2)$ executions of GAUSS-SEIDEL over D with schedule S , every arc of D^n is preserved and no new arcs are added, so $D^{n+1} = \text{GAUSS-SEIDEL}(D^n, S) = D^n$ and thus D^n is a structural fixed point. \square

PROOF OF LEMMA 4.4. Let s be the update function of S . First suppose that D has no arcs (u, v) where $u \in B_0$ and $v \in B_1$. Since $s(u) \geq s(v)$ for every arc (u, v) in D , GAUSS-SEIDEL adds to D^S every arc of D without changes and then returns. Therefore D is a structural fixed point reached after one execution of GAUSS-SEIDEL. Notice that in this case, as each vector $\vec{x}(a, D)$ is null, the condition holds trivially.

Assume now that there is an arc $(i, a) \in A$ such that $s(i) < s(a)$, meaning that the vector $\vec{x}(a, D)$ is not null. Let $M \in \mathcal{M}_{q \times q}(\{0, 1\})$ be the adjacency matrix of $D(B_0)^T$, the subdigraph induced in D by the first block with reversed arcs, ie:

$$M_{ij} = 1 \iff (j, i) \in A$$

Then, by Lemma 4.2, for every $n \in \mathbb{N}$, $\vec{x}(a, D^n) = \vec{x}(a, D) \cdot M^n$, where D^n is the digraph obtained after n iterated executions of GAUSS-SEIDEL over D (with schedule S). By Theorem 3.2, M is irreducible and imprimitive, therefore if we name ℓ to the length of the shortest circuit in D , by Proposition 3.11 there exist $t \in \mathcal{O}(q)$, $p \in \mathcal{O}(\ell)$, such that

$$\vec{x}(a, D^{t+p}) = \vec{x}(a, D^t) \tag{A.3}$$

Arcs connecting nodes in the same block are always preserved by GAUSS-SEIDEL. Assume that there is an arc $(j, k) \in A$ such that $s(k) < s(j)$ -this arc (j, k) is preserved without changes by GAUSS-SEIDEL- and an arc $(i, a) \in A$ such that $s(i) < s(a)$. Since $D(B_0)$ is strongly connected there is a directed path in $D(B_0)$ from k to i of length $p \in \mathcal{O}(q)$. By Lemma 3.1, $M_{i,k}^p > 0$, and given that $x(a, D)_i > 0$, from Lemma 4.2 it follows that $x(a, D^p)_k > 0$, namely there is in D^p an arc (k, a) , where $s(k) < s(a)$. By the definition of GAUSS-SEIDEL algorithm, in D^{p+1} is added the arc (j, a) , where $s(j) = s(a) = 1$, meaning that GAUSS-SEIDEL is adding an arc that connects nodes belonging to the second block B_1 . These arcs are not described by the vectors $\vec{x}(a, D)$ (because the components of these vectors depict nodes in the first block), but what matters here is that these arcs are added in the iteration $p + 1 \in \mathcal{O}(q)$ of GAUSS-SEIDEL, and these arcs remain in the digraphs D^m , with $m \geq p + 1$.

The foregoing paragraph and equality (A.3) show that in at most $\mathcal{O}(q)$ executions of GAUSS-SEIDEL over D with schedule S , this iterative procedure outputs a network in the filter attractor $\mathcal{A}(D, S)$, and the size p of this set is $\mathcal{O}(\ell)$.

Also Proposition 3.11 states that $p = 1$, that is, the filter attractor is a structural fixed point if and only if for every $a \in B_1$ such that $\vec{x}(a, D)$ is non-null, $\vec{x}(a, D)$ has at least one positive component associated to a node in V_i , for every imprimitivity set V_i from node partition $(V_i)_{i=1}^{k(D(B_0))}$ given by Lemma 3.4. This is equivalent to, for each node $a \in B_1$ such that there is an arc $(u, a) \in A$ with $u \in B_0$, there is an arc $(u_i, a) \in A$ such that $u_i \in V_i$, for every set V_i from node partition $(V_i)_{i=1}^{k(D(B_0))}$. \square

Matrix computation of Gauss-Seidel operator

Let $G = (V, E)$ be a Boolean disjunctive network of n nodes, and A_G its adjacency matrix. Lets denote by \hat{c}_j the j -th column of A_G , that is, we have that $A_G = [\hat{c}_1 \hat{c}_2 \cdots \hat{c}_n]$. Let S be an update schedule for G with B_0, B_1, \dots, B_k its sequence of update blocks. We define the *projection of A_G over block B_ℓ* , denoted by $A_{G[\ell]}$, as an $n \times n$ matrix defined as follows. Let \hat{c}'_j be the j -th column in $A_{G[\ell]}$, then we have that

- $\hat{c}'_j = \hat{c}_j$ if $j \in B_\ell$, and
- $\hat{c}'_j = \hat{e}_j$ if $j \notin B_\ell$, where \hat{e}_j is the j -th column of the $n \times n$ identity matrix

Notice that $A_{G[\ell]}$ has a 1 in its position (i, j) if and only if

- $(i, j) \in V$ and j is updated in block ℓ , or
- $i = j$ and i is not updated in block ℓ .

Lets understand the usefulness of last definition. Let $G = (V, E)$ be the disjunctive Boolean network of size n given by the global transition function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and adjacency matrix $A_G \in \mathcal{M}_{n \times n}(\{0, 1\})$. Let S be the block-sequential update schedule with m blocks B_0, \dots, B_{m-1} . For an arbitrary initial condition consisting in the row vector $\vec{x} = (x_1, \dots, x_n)$, to compute the state vector $F_S(\vec{x})$ obtained after a complete update sequence we can compute first the vector $F_{[B_0]}(\vec{x})$, then vector $F_{[B_1]}(F_{[B_0]}(\vec{x}))$ and so on. Recall that

each of these vectors consists in the parallel update of those nodes belonging to the respective block, while the other nodes remain unchanged. It was already said in Section 3 that the parallel update of disjunctive functions is simulatable with the product of the adjacency matrix, therefore, we can write the following

$$F_{[B_0]}(\vec{x})_i = \begin{cases} f_i(\vec{x}) & i \in B_0, \\ x_i & \text{otherwise} \end{cases} = \begin{cases} \vec{x} \cdot \hat{c}_i & i \in B_0, \\ \vec{x} \cdot \hat{e}_i & \text{otherwise} \end{cases} = (\vec{x} \cdot A_{G[0]})_i$$

In this way, since the state vector $F_S(\vec{x})$ obtained after a complete update sequence corresponds to the composition $F_{[B_{m-1}]} \circ \dots \circ F_{[B_1]} \circ F_{[B_0]}(\vec{x})$, using previous idea it can be computed as $F_S(\vec{x}) = \vec{x} \cdot A_{G[0]} \cdots A_{G[m-1]}$.

We are now ready to prove the result that shows how to compute the disjunctive network returned by Gauss-Seidel operator as a matrix product. In what follows, given two matrices M_1 and M_2 , we denote by $M_1 \cdot M_2$ the Boolean matrix product.

Lemma A.4 *Let $G = (V, E)$ be a disjunctive Boolean network and S be a block-sequential update schedule of $m \geq 1$ blocks. Let M^S be the adjacency matrix of the network G^S obtained after executing GAUSS-SEIDEL onto (G, S) . Then, it holds the following equality:*

$$M^S = A_{G[0]} \cdot A_{G[1]} \cdots A_{G[m-1]}$$

PROOF. Let s be the update function of schedule S and $(B_j)_{j=0}^{m-1}$ the respective set of blocks. We will first prove the following property

The j -th column of the product $A_{G[0]} \cdots A_{G[i]}$ (subject to $s(j) \geq i + 1$) corresponds to the j -th column of the identity matrix:

$$(A_{G[0]} \cdots A_{G[i]})_{\bullet j} = I_{\bullet j}, \quad \forall j \text{ s.t. } s(j) \geq i + 1, \quad \forall 0 \leq i \leq m - 1$$

We proceed by induction on i . The base case for $i = 0$ is direct from definitions:

$$(A_{G[0]})_{\bullet j} = I_{\bullet j} \quad \text{if } s(j) \neq 0 \iff s(j) \geq 1 = 0 + 1$$

Now we assume the inductive hypothesis for some $i \geq 0$. Let b be a node such that $s(b) \geq i + 2$, and a be any node:

$$\begin{aligned}
(A_{G[0]} \cdots A_{G[i+1]})_{ab} &= \sum_{k \in V} (A_{G[0]} \cdots A_{G[i]})_{ak} \underbrace{(A_{G[i+1]})_{kb}}_{= I_{kb}} \\
&\hspace{15em} (s(b) \neq i+1) \\
&= \sum_{s^{(k)} \geq i+1} \underbrace{(A_{G[0]} \cdots A_{G[i]})_{ak}}_{= I_{ak}} I_{kb} + \sum_{s^{(k)} \leq i} (A_{G[0]} \cdots A_{G[i]})_{ak} \underbrace{I_{kb}}_{= 0} \\
&\hspace{15em} \text{(IH)} \hspace{15em} (k \neq b) \\
&= \sum_{s^{(k)} \geq i+1} I_{ak} I_{kb} \\
&= I_{ab} I_{bb} = I_{ab}
\end{aligned}$$

which shows the case for $i+1$ is true and the property holds. The second part of the demonstration consists in proving the next property:

If we name $M(A_i) \in \mathcal{M}_{|V| \times |V|}(\{0, 1\})$, $i = 0, \dots, m-1$, to the adjacency matrix of the set of arcs A after the iteration i and before the $(i+1)$ -th iteration of the main for loop of GAUSS-SEIDEL(G, S), then the j -th column of the product $A_{G[0]} \cdots A_{G[i]}$ (subject to $s(j) \leq i$) corresponds to the j -th column of $M(A_i)$:

$$(A_{G[0]} \cdots A_{G[i]})_{\bullet j} = M(A_i)_{\bullet j}, \quad \forall j \text{ s.t. } s(j) \leq i, \quad \forall 0 \leq i \leq m-1$$

We will prove this proposition by induction on i . When we write A_G we refer to the adjacency matrix of G . We put here the code of GAUSS-SEIDEL for readability:

GAUSS-SEIDEL($G = (V, E), S$)

```

1   $A = \phi, \quad (A \subseteq E) \quad //$  initialization
2   $I = \phi, \quad (I \subseteq V)$ 
   Let  $B_0, \dots, B_{m-1}$  be the blocks of  $S$ , and let  $s$  be the update function
3  for  $i = 0$  to  $m-1$ 
4     for  $j \in B_i$ 
5         for  $(u, j) \in E$            // we go over every arc incident to node  $j$ 
6             if  $s(u) \geq s(j)$ 
7                  $A = A \cup \{(u, j)\}$ 
8             else
9                 for  $(t, u) \in A$ 
10                     $A = A \cup \{(t, j)\}$ 
11              $I = I \cup \{j\}$ 
12 return  $(I, A)$ 

```

Let us examine the basis when $i = 0$. By definition of the algorithm, we see that in the first iteration of the main for loop, every arc (u, j) incident to some node j belonging to the first

block B_0 is processed, and since $s(u) \geq s(j)$ for every one of these arcs, all these arcs are added to A . Hence $M(A_0)_{uj} = 1$ if and only if $(u, j) \in E, (s(j) = 0)$, and this is equivalent to $(A_{G[0]})_{uj} = 1$ if $s(j) = 0$ (or if $s(j) \leq i = 0$), and the basis of induction holds.

Now we assume the inductive hypothesis for some $i \geq 0$. Let b be a node such that $s(b) \leq i + 1$, and a be any node:

$$\begin{aligned}
(A_{G[0]} \cdots A_{G[i+1]})_{ab} &= \sum_{k \in V} (A_{G[0]} \cdots A_{G[i]})_{ak} (A_{G[i+1]})_{kb} \\
&= \sum_{s(k) \geq i+1} \underbrace{(A_{G[0]} \cdots A_{G[i]})_{ak}}_{= I_{ak}} (A_{G[i+1]})_{kb} + \sum_{s(k) \leq i} \underbrace{(A_{G[0]} \cdots A_{G[i]})_{ak}}_{= M(A_i)_{ak}} (A_{G[i+1]})_{kb} \\
&\quad \text{(1st. prop)} \qquad \qquad \qquad \text{(IH)} \\
&= \begin{cases} \sum_{s(k) \geq i+1} I_{ak} \underbrace{I_{kb}}_{= 0} + \sum_{s(k) \leq i} M(A_i)_{ak} I_{kb} & s(b) < i + 1, \\ & (k \neq b) \\ \sum_{s(k) \geq i+1} I_{ak} (A_G)_{kb} + \sum_{s(k) \leq i} M(A_i)_{ak} (A_G)_{kb} & s(b) = i + 1 \end{cases} \\
&= \begin{cases} M(A_i)_{ab} & s(b) < i + 1, \\ (A_G)_{ab} + \sum_{s(k) \leq i} M(A_i)_{ak} (A_G)_{kb} & s(b) = i + 1, s(a) \geq i + 1 \\ \sum_{s(k) \leq i} M(A_i)_{ak} (A_G)_{kb} & s(b) = i + 1, s(a) < i + 1 \end{cases}
\end{aligned}$$

Now we analyze by cases if the equality holds. If $s(b) < i + 1$, this means that node b is processed in some iteration previous to the $(i + 1)$ -th iteration, therefore the arcs in A incident to b do not change in the $(i + 1)$ -th iteration, and $M(A_i)_{ab} = M(A_{i+1})_{ab}$ if $s(b) < i + 1$, so the property holds in this case. If we now assume $M(A_{i+1})_{ab} = 1$ and $s(a) \geq i + 1 = s(b)$, meaning arc (a, b) was added to A in line 7 (at $(i + 1)$ -th iteration), this implies $(a, b) \in E$, or in other words, $(A_G)_{ab} = 1$, hence $(A_{G[0]} \cdots A_{G[i+1]})_{ab} = 1$. In the case when $M(A_{i+1})_{ab} = 1$ and $s(a) < i + 1 = s(b)$, meaning arc (a, b) was added to A in line 10 (at $(i + 1)$ -th iteration), this implies there is a node $k \in V$ such that, $(k, b) \in E$, $(a, k) \in A_i$ ((a, k) belongs to A when executing $(i + 1)$ -th iteration), then k was processed in some iteration previous to $(i + 1)$ -th, and $s(k) < s(b) = i + 1$. Thus, $\sum_{s(k) \leq i} M(A_i)_{ak} (A_G)_{kb} = 1 = (A_{G[0]} \cdots A_{G[i+1]})_{ab}$. The case when $M(A_{i+1})_{ab} = 0$ it is analogous to the cases shown newly.

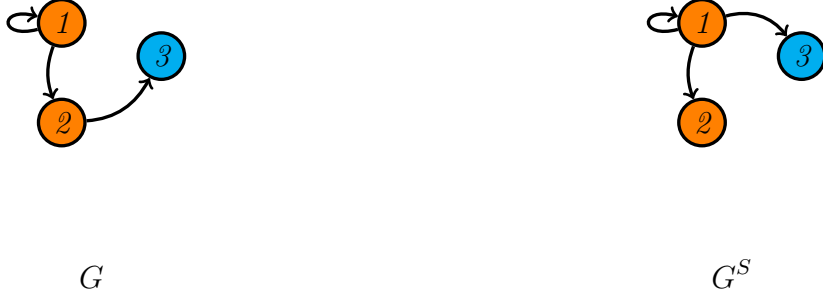
If we take the last property when all iterations have been executed ($i = m - 1$), we have $(A_{G[0]} \cdots A_{G[m-1]})_{\bullet j} = M(A_{m-1})_{\bullet j}$, for all j verifying $s(j) \leq m - 1$, or equivalently, for all j . Since $M(A_{m-1})$ is the adjacency matrix of the set A returned by GAUSS-SEIDEL, and Lemma A.3 assures this algorithm returns the adjacency matrix of network G^S , it holds that:

$$A_{G[0]} \cdot A_{G[1]} \cdots A_{G[m-1]} = M(A_{m-1}) = M^S$$

□

Example 20 Let $G = (V, E)$ be the disjunctive Boolean network given by the digraph in the picture, which defines a global transition function $F : \{0, 1\}^3 \rightarrow \{0, 1\}^3$. Let S be the

block-sequential update schedule of two blocks: $B_0 = \{1, 2\}$ (orange nodes) and $B_1 = \{3\}$ (cyan node). For an initial condition consisting in the row vector \vec{x} , to compute the state vector $F_S(\vec{x})$ we can compute first vector $F_{[B_0]}(\vec{x}) = \vec{x} \cdot A_{G[0]}$, and then vector $F_S(\vec{x}) = F_{[B_1]}(F_{[B_0]}(\vec{x})) = \vec{x} \cdot A_{G[0]} \cdot A_{G[1]}$.



The matrix product $A_{G[0]} \cdot A_{G[1]}$ corresponds, as it should be, to the adjacency matrix of network $G^S = \text{GAUSS-SEIDEL}(G, S)$.

$$A_{G[0]} \cdot A_{G[1]} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

A.5 Proofs for Section 5

Lemma A.5 Let $G = (V, E)$ be a disjunctive Boolean network, and a block-sequential update schedule S for this network with $m \geq 1$ blocks $(B_j)_{j=0}^{m-1}$. Let $a \in V \cap B_{m-1}$ be a node belonging to the last block ($(m-1)$ -th block). Let $v \in V$ be a node such that there is in G a path $P_{a,v}^G$ starting from a to v . If we consider the filter consisting in the system

$$\begin{aligned} G^0 &\equiv G \\ G^{i+1} &\equiv \text{GAUSS-SEIDEL}(G^i, S), \quad i \geq 0 \end{aligned}$$

then, for every $i \geq 0$, there exists a path $P_{a,v}^{G^i}$ in G^i starting from a to v , connecting nodes belonging to the set of nodes connected by $P_{a,v}^G$. Also, it holds that $|P_{a,v}^{G^{i+1}}| \leq |P_{a,v}^{G^i}|$, $\forall i \geq 0$.

PROOF. The proof is by induction on $i \geq 0$. The base case, $i = 0$, is straightforward by the hypothesis. Assume now the lemma holds for $i = k$, and let $P_{a,v}^{G^k}$ be a path in G^k connecting the following sequence of nodes:

$$a = u_0, u_1, \dots, u_{l-1}, u_l = v$$

Suppose first that

$$\forall j \in \{0, \dots, l-1\}, \quad s(u_j) \geq s(u_{j+1})$$

where s is the update function of schedule S . By the Lemma 2.2, each arc starting from u_j to u_{j+1} is preserved after applying the algorithm, so $P_{a,v}^{G^k} = P_{a,v}^{G^{k+1}}$ (and clearly, both paths have the same length).

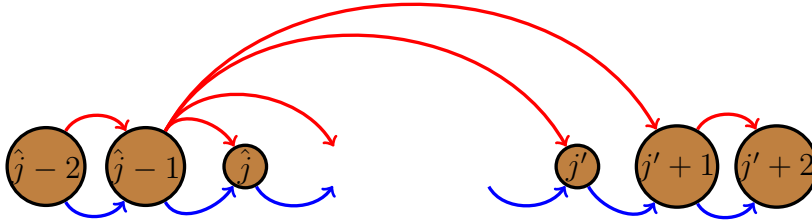
Suppose now there is $j' \in \{1, \dots, l-1\}$ such that $s(u_{j'}) < s(u_{j'+1})$ and $s(u_{j'-1}) \geq s(u_{j'})$ (necessarily $j' > 0$ given that $u_0 = a$ is updated in the last block). By characterization 2.2, it holds that $(u_{j'-1}, u_{j'+1}) \in G^{k+1}$. If we assume that for every $j \neq j'$, $s(u_j) \geq s(u_{j+1})$, all arcs (u_j, u_{j+1}) meeting $j \neq j'$ are preserved, and then there is a path $P_{a,v}^{G^{k+1}}$ in G^{k+1} starting from a to v consisting in the arcs from $P_{a,u_{j'-1}}^{G^k}$, the arc $(u_{j'-1}, u_{j'+1})$, and the arcs from $P_{u_{j'+1},v}^{G^k}$, which verifies

$$|P_{a,v}^{G^{k+1}}| = |P_{a,v}^{G^k}| - 1 \leq |P_{a,v}^{G^k}|$$

Suppose now there is a set of consecutive indexes $J = \{\hat{j}, \hat{j} + 1, \dots, j' - 1, j'\}$ ($\hat{j} > 0$) satisfying that, for every $j \in J$, $s(u_j) < s(u_{j+1})$, and $s(u_{\hat{j}-1}) \geq s(u_{\hat{j}})$. By characterization 2.2, for every $j \in J$ the arc $(u_{j-1}, u_j) \in G^{k+1}$ and also $(u_{\hat{j}-1}, u_{j'+1}) \in G^{k+1}$. If it is further assumed that for every $j \notin J$, $s(u_j) \geq s(u_{j+1})$, then (again by characterization 2.2) $(u_j, u_{j+1}) \in G^{k+1}$ for all $j \notin J$, and thus a path in G^{k+1} starting from a to v consists in the arcs from $P_{a,u_{\hat{j}-1}}^{G^k}$, the arc $(u_{\hat{j}-1}, u_{j'+1})$, and the arcs from $P_{u_{j'+1},v}^{G^k}$, which verifies

$$|P_{a,v}^{G^{k+1}}| = |P_{a,v}^{G^k}| - (j' - \hat{j} + 1) \leq |P_{a,v}^{G^k}|$$

The figure below illustrates this situation, where the blue arcs belong to G^k , while the red arcs pertain to G^{k+1} :



Suppose now the general case in which there are C disjoint sets of consecutive indexes:

$$J^d = \{l^d, l^d + 1, \dots, u^d - 1, u^d\} \quad (\forall d \in \{1, \dots, C\}, 0 < l^d \leq u^d)$$

satisfying that

$$\begin{aligned} \forall j \in J^d, \forall d \in \{1, \dots, C\} \quad & s(u_j) < s(u_{j+1}) \\ \forall j \in \{0, \dots, l\} / \bigcup_{d \in \{1, \dots, C\}} J^d, \quad & s(u_j) \geq s(u_{j+1}) \end{aligned}$$

It is clear that, for every J^d , it does not occur that $l^d - 1 \in J^{d-1}$ or $u^d + 1 \in J^{d+1}$ (this is by definition, as $l^d - 1$ and l^d are consecutive indexes, if $s(u_{l^d-1}) < s(u_{l^d})$, then $l^d - 1 \in J^{d-1} \cap J^d = \phi$), so

$$\forall d, \quad l^d - 1, u^d + 1 \in \{0, \dots, l\} / \bigcup_{d \in \{1, \dots, C\}} J^d$$

By characterization 2.2, and the definition of sets J^d , it holds that for every $d \in \{1, \dots, C\}$, $(u_{l^d-1}, u_{u^d+1}) \in G^{k+1}$ (or in the case when $d = C$ and $u^C = u_l = v$, it holds that $(u_{l^C-1}, u_{u^C}) \in G^{k+1}$), and also it is deduced that

$$\forall j \in \{0, \dots, l\} / \bigcup_{d \in \{1, \dots, C\}} J^d, \quad (u_j, u_{j+1}) \in G^{k+1}$$

Notice that since $u_0 = a$ belongs to the last block, it holds that $s(u_0) \geq s(u_1)$, meaning the first arc from $P_{a,v}^{G^{k+1}}$ always has the node $u_0 = a$ as its starting node. And in any of the cases $s(u_{l-1}) \geq s(u_l)$ or $s(u_{l-1}) < s(u_l)$, the ending node in the last arc from $P_{a,v}^{G^{k+1}}$ is $u_l = v$. Then, proceeding analogously to when there is only one set J of consecutive indexes, it is obtained a path $P_{a,v}^{G^{k+1}}$ in G^{k+1} from a to v , satisfying that

$$|P_{a,v}^{G^{k+1}}| = |P_{a,v}^{G^k}| - \sum_{d=1}^C (u^d - l^d + 1) \leq |P_{a,v}^{G^k}|$$

In any of the previous cases, the nodes connected by $P_{a,v}^{G^{k+1}}$, belong to the set of nodes connected by $P_{a,v}^{G^k}$, since this is true for every k , the nodes connected by $P_{a,v}^{G^i}$, for every $i \geq 0$, are connected by the path $P_{a,v}^G$ too.

□

Lemma A.6 *Let $G = (V, E)$ be a disjunctive network and S be a block-sequential update schedule of $m \geq 1$ blocks. If $\deg_G^-(i) = 1, \forall i \in V$ then*

$$\deg_{G^k}^-(i) = 1, \quad \forall i \in V, \quad \forall k \geq 0$$

where G^k is the network obtained after k applications of GAUSS-SEIDEL onto G and S .

PROOF. Let s be the update function of S and $(B_j)_{j=0}^{m-1}$ the respective set of blocks. The proof is by strong induction on the number $1 \leq n \leq m$ of blocks processed by the algorithm; then, the statement to prove is the following

$$\text{for each node } i \in B_d, \quad 0 \leq d \leq n - 1 \text{ it holds that } \deg_{G^1}^-(i) = 1, \quad \forall n \in \{1, \dots, m\}$$

Let us see the base case ($n = 1$). When the algorithm processes a node $i \in B_0$, and checks for the unique incident arc to i in G , (j, i) , it always holds that $0 = s(i) \leq s(j)$, then, the

algorithm branches to line 7, and (j, i) is added to A . There are no more actions with node i (since there are no more incident arcs to i), so that i has in-degree in G^1 equal to 1, and the next node in B_0 is processed, for which the algorithm executes the same actions. Thus the base case holds.

Assume now the property holds for $n = p$, ie, the algorithm has already processed the blocks B_0, \dots, B_{p-1} , then the property is true in these blocks and let us prove the statement for block B_p . Let $b \in B_p$ be a node, and let (a, b) be the unique incident arc to b in G . If $s(a) \geq s(b)$, the algorithm branches to line 7, the arc (a, b) is added to A , and the processing of node b is finished. If $s(a) < s(b)$, the algorithm goes to lines 9 and 10, and for each arc $(t, a) \in A$, the arc (t, b) is added to A . Since a belongs to one of the first p blocks, by inductive hypothesis there exists only one arc $(t, a) \in A$: hence (t, b) is added to A , the processing of node b is finished and the next node in B_p is processed, for which the algorithm executes the same actions. Thus the property is true for block B_p , and therefore it holds that if the algorithm has processed n blocks (where $n \in \{1, \dots, m\}$), then $\deg_{G^1}^-(i) = 1$ for all $i \in B_d, 0 \leq d \leq n - 1$.

When the algorithm finishes its execution, all blocks have been processed, and since the previous property holds for $n = m$, every node in G^1 has in-degree equal to 1. When the algorithm is applied onto G^1 and S , G^1 verifies that each node has in-degree equal to 1, then by the same argument G^2 verifies this too, and then every node in G^k , with $k \geq 0$, has in-degree equal to 1. \square

Lemma A.7 *The algorithm TEST-CYCLE-FILTER is correct, ie, it solves the decision problem CYCLE-FILTER on an input consisting in a strongly connected digraph $D = (V, A)$, and a block-sequential update S for D .*

PROOF. Let s the update function of schedule S , and $(B_j)_{j=0}^{m-1}$ the respective set of $m \geq 1$ blocks. We first prove the following equivalence:

There is a simple circuit C in D such that $\mathcal{IL}(C, S) = L$, where B_{i_C} is the last block having nodes from circuit C , $0 \leq i_C \leq m-1$, if and only if, after executing $\text{ADD-EDGES}(D, i_C, \hat{D}_{i_C}, S)$, the subdigraph \hat{D}_{i_C} has a simple circuit of length equal to L .

Let us see the sufficiency. Assume C reduces to n_C maximal directed paths in B_{i_C} , where each j -th path connects $(\ell_j + 1)$ different nodes, so it holds that $L = \sum_{j=1}^{n_C} \ell_j + n_C$ (Note 4). Let P_j be one of these paths, for which its ending node will be referred as $e_j \in B_{i_C}$. Since P_j correspond to a sequence of arcs from C , there is a path in D from e_j to i_{j+1} , where $i_{j+1} \in B_{i_C}$ is the initial node from the next maximal path P_{j+1} induced in $D(B_{i_C})$ by circuit C (if $n_C = 1$, then i_{j+1} is simply i_j , the starting node from path P_j). Call this path as $P_{e_j, i_{j+1}}^D$. It holds that,

$$\forall a \in V \text{ connected by } P_{e_j, i_{j+1}}^D \text{ such that } a \neq e_j, i_{j+1}, \quad s(a) < s(e_j) = s(i_{j+1}) \quad (\text{A.4})$$

(if not, P_{j+1} would not be the next maximal path induced by C in B_{i_C}). According to the definitions of TEST-CYCLE-FILTER, $S^{in}(i_{j+1}) \neq \phi$ and $S^{out}(e_j) \neq \phi$. Therefore, by (A.4)

there must be a path from some $y \in S^{out}(e_j)$ to some $x \in S^{in}(i_{j+1})$ in $D(B_0 \cup \dots \cup B_{i_C-1})$, and the arc (e_j, i_{j+1}) is added to \hat{D}_{i_C} . In this way, each path $P_{e_j, i_{j+1}}^D$ connecting the maximal path P_j with maximal path P_{j+1} ($j = 1, \dots, n_C$) produces one arc (e_j, i_{j+1}) in \hat{D}_{i_C} , finally forming -similarly to what happens in the proof of Proposition 5.3- a closed simple path connecting nodes in B_{i_C} of length equal to

$$\sum_{j=1}^{n_C} (\ell_j + 1) = \sum_{j=1}^{n_C} \ell_j + n_C = L$$

For the converse, assume the subdigraph \hat{D}_{i_C} -after executing $\text{ADD-EDGES}(D, i_C, \hat{D}_{i_C}, S)$ - has a circuit C^* of length equal to L . Since ADD-EDGES does not erase edges from D , C^* may be a circuit originally from digraph D : in this case, C^* was a circuit completely contained in block B_{i_C} , then, its length matches with its induced length with respect to S , and the property follows. Now, if C^* was not part of digraph D , C^* was added by procedure ADD-EDGES : assume that $L = o^* + p^*$, where o^* is the number of arcs of C^* that were originally part of D , and p^* the number of arcs added later by $\text{ADD-EDGES}(D, i_C, \hat{D}_{i_C}, S)$. For each one of the arcs (a_i, b_i) added later ($i = 1, \dots, p^*$), necessarily there is in $D(B_0 \cup \dots \cup B_{i_C-1})$ a path P_i from y_i to x_i , for some $x_i \in S^{in}(b_i), y_i \in S^{out}(a_i)$. Replacing in C^* each arc (a_i, b_i) by the path consisting in the arc (a_i, y_i) , the path P_i , and the arc (x_i, b_i) , results in a circuit C^{**} that necessarily belongs to D and that has induced length with respect to S equal to $o^* + p^* = L$.

We need to show also that the pre-processing of \hat{D} does not modify the lengths of simple circuits. For this, we will refer to the digraph \hat{D} after the pre-processing with NON-TRIVIAL-SCC in line 5, as \hat{D}^* , and we simply speak of \hat{D} to refer this digraph before the pre-processing (in line 4). Hereafter, we write r_1 to mean the root of the first non-trivial s.c.c. found by NON-TRIVIAL-SCC (in line 5). The assertion to prove is:

There is a cycle of length L in \hat{D} if and only if there is a cycle of length L in \hat{D}^ .*

If there is a cycle C of length L in \hat{D} , and this cycle connects nodes distinct than any root r_j of some non-trivial s.c.c. C_j , C is preserved (by definition, the nodes removed from \hat{D} in line 5 are not connected by any circuit) and is present in \hat{D}^* , and the required holds. It is apparent that C cannot contain two roots of different s.c.c.'s of \hat{D} , then, if C contains the root r_1 , the arcs adjacent to r_1 are not modified, and the required holds. If C contains some root r_j distinct than r_1 , the arcs of C adjacent to r_j are replaced by arcs adjacent to r_1 in procedure $\text{REPLACE-NODE}(\hat{D}, r_j, r_1)$, without changing the length of C in \hat{D}^* , and this proves one of the implications. Now, if there is a cycle C of length L in \hat{D}^* , this cycle may or may not contain the node r_1 . If it does not contain the node r_1 , C was not created by $\text{REPLACE-NODE}(\hat{D}, r_j, r_1)$, and C is a cycle in \hat{D} too. If C does contain the node r_1 , C may or may not be created by $\text{REPLACE-NODE}(\hat{D}, r_j, r_1)$: in the first case, there was in \hat{D} a node r_j such that, replacing in C the arcs adjacent to r_1 by arcs adjacent to r_j , results in a cycle of length L that belonged to the non-trivial s.c.c. C_j of \hat{D} . In the second case, the cycle C belonged to the s.c.c. C_1 of \hat{D} , which concludes the proof of the assertion.

Now, let us suppose that $\text{TEST-CYCLE-FILTER}(D, S)$ returns TRUE , which is equivalent

to the set of cycle lengths of \hat{D}^* (after executing $\text{NON-TRIVIAL-SCC}(\hat{D})$, where \hat{D}^* is strongly connected) is setwise coprime. By the second assertion proved, this is equivalent to the set of cycle lengths of \hat{D} is setwise coprime. Any circuit from \hat{D} is completely contained in some \hat{D}_i (by definition of \hat{D}), after having executed $\text{ADD-EDGES}(D, i, \hat{D}_i, S)$ (again by definition of \hat{D}), then, by the first assertion proved, the set of cycle lengths of \hat{D} is setwise coprime, is equivalent to the set of cycle induced lengths (with respect to S) of D is setwise coprime, which solves CYCLE-FILTER problem. \square

The proof of the next lemma can be reviewed on [37].

Lemma A.8 ([37]) *Let G be a strongly connected digraph and S a block-sequential update schedule of its nodes. Then, G^S is comprised of one unique non-trivial strongly connected component and possibly some outgoing acyclic subdigraphs.*

A.6 Relation between a block and the previous

Lemma A.9 *Let $G = (V, E)$ be a disjunctive Boolean network, and a block-sequential update schedule S for this network having $m \in \mathbb{N}$ blocks $(B_i)_{i=0}^{m-1}$. Consider the set of nodes belonging the first $n \leq m$ blocks, $V' = B_0 \cup \dots \cup B_{n-1}$, and the digraph induced in G by this set of nodes: $G(V')$. Let S' be a block-sequential update for $G(V')$, such that S restricted to the nodes in V' is equal to S' , ie, if s and s' are the update functions of schedule S and S' respectively, then for all $v \in V'$, $s(v) = s'(v)$. Then, the networks returned by GAUSS-SEIDEL :*

$$G^S = (V, E^S) = \text{GAUSS-SEIDEL}(G, S)$$

$$G(V')^{S'} = (V', E^{S'}) = \text{GAUSS-SEIDEL}(G(V'), S')$$

satisfy $E^{S'} = E^S(V')$, where $E^S(V')$ corresponds to the restriction of E^S to those arcs meeting their starting and ending node belong to V' .

PROOF. Let $(i, j) \in E^S(V')$ be an arc of E^S , with $i, j \in V' = B_0 \cup \dots \cup B_{n-1}$. By characterization 2.2, this is equivalent to the existence of a path $\{v_0, v_1, \dots, v_l\}$ in G from $i = v_0$ to $j = v_l$ such that

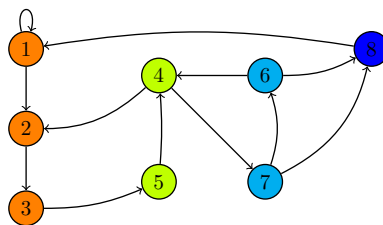
$$\begin{aligned} s(v_0) &\geq s(v_1) \\ \forall 1 \leq k < l, \quad s(v_k) &< s(v_{k+1}) \end{aligned}$$

From the first condition, we know v_1 is in the same block or one updated before than the block of $v_0 = i$. From the second condition, we know every node from v_1 until v_{l-1} belong to blocks updated before than the block containing $v_l = j$. Thus, all the nodes in the path belong to the first n blocks, and by hypothesis, the above conditions are equivalent to

$$\begin{aligned} s'(v_0) &\geq s'(v_1) \\ \forall 1 \leq k < l, \quad s'(v_k) &< s'(v_{k+1}) \end{aligned}$$

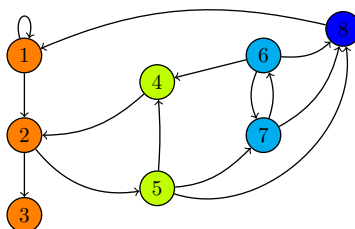
and again by the characterization 2.2, these conditions are equivalent to $(i, j) \in E^{S'}$. \square

Example 21 The following disjunctive network G of size 8 is defined by its associated digraph in the image below. We consider a block-sequential update schedule S of 4 blocks $(B_i)_{i=0}^3$: $(1, 2, 3)(4, 5)(6, 7)(8)$. The node fill colors are put to differentiate the different blocks: first block is orange, second is green, third is cyan and fourth is blue.



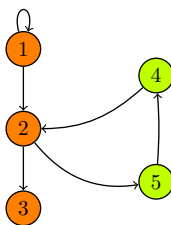
G

If we execute $G^S = (V, E^S) = \text{GAUSS-SEIDEL}(G, S)$, we get the network G^S :



G^S

Now, if we take the subdigraph induced in G by the first two blocks, $G' = G(B_0 \cup B_1) = G(\{1, 2, 3, 4, 5\})$, and consider the schedule S' consisting in restrict S to the first two blocks, $S' = (1, 2, 3)(4, 5)$, we get the following network when executing $\text{GAUSS-SEIDEL}(G', S')$



$G'^{S'}$

It can be seen that this is the same network when we get the subdigraph of G^S induced by the node set $B_0 \cup B_1$.

Appendix B

Software

In this section, we describe briefly the code written during the development of this thesis, both the tasks it performs and interaction with the user. The software was developed in order to acquire intuition about some study objects or to discard misleading ideas, objective which may require a large number of examples for testing and studying. In particular, the software was critical to formulate the Definition 5.4 -the cycle-filter condition-, the results about the linear transient given some structural properties -Theorem 3.12-, the Theorem 3.9 about the quadratic bound for the transient length, the Theorem 6.2 about structural fixed points or anything that had to do with simple visualization of dynamics of (small) networks -for example, the figures in this work displaying state transition graphs-.

The code was written in Java language, due to the portability it grants, and it makes use of LogicalModel, an open source library developed to improve the interoperability between logical modeling tools, motivated by the CoLoMoTo (Common Logical Modelling Toolbox) discussion group (www.colomoto.org). The library can be used as a standalone command line tool for model conversion, after compiling it, and the source code is available at <https://github.com/colomoto/logicalmodel>. LogicalModel relies on JSBML [23] (<http://sbml.org/Software/JSBML>) for SBML-qual import/export, and on a small MDD manipulation toolkit, mddlib (<https://github.com/colomoto/mddlib>). Currently, it provides import and export capabilities between the following formats:

- SBML-qual [16] (System Biology Markup Language Qualitative Models Package) http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Qualitative_Models
- Raw logical functions
- boolsim
- GINML [77] <http://ginsim.org/home> (only export)
- GNA [12] <http://ibis.inrialpes.fr/article122.html> (only export)

The format we use in our applications is that of raw logical functions, because it does not require any additional syntax the user must learn. As a example, if we want to hand over the Boolean network from Example 1 to our application, it can be coded in a .txt file as follows (for the following examples, we assume this file is named as `examp_3.1.txt`):


```
x1: x4
x2: x1 | !x4
x3: x2 & x3
x4: x3 & x5
x5: !x2
```

Variable ID's can be any string (a character chain without spaces), and they must be consistent along function definition (e.g. 'a' and 'A' refer to distinct variables). Constant variables are simply coded with its respective value (e.g. 'A: 0'), but it must be noted that currently mddlib does not provide complete functionality for constant variables in raw function format.

The codes were written, executed and tested in a machine with Java 1.7.0_25 and Ubuntu 12.04.2 LTS.

B.1 Dynamics simulation

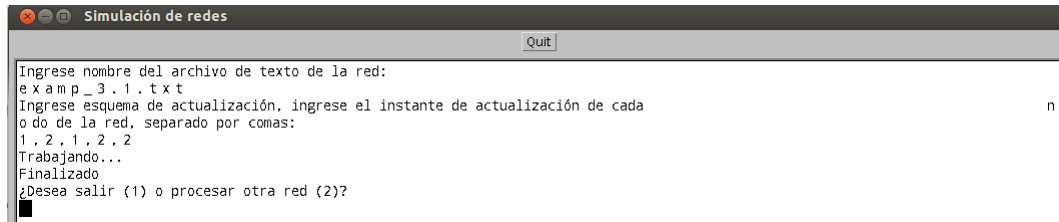
One of the two purposes of the code written in this thesis is to build a simple tool for network dynamics visualization. This resulted in the development of three Java classes: `BlockSequentialUpdater.java`, `Net_simulator.java` and `TestUpdaters.java`. The first of them implements the update for a block-sequential schedule, that is, the computation of the network configuration after a complete update sequence -starting from a given network configuration-. This class inherits from `AbstractUpdater.java`, which implements the general behavior of any updater of the network (in fact, `AbstractUpdater.java` implements the interface given by `LogicalModelUpdater.java`).

The node blocks of the schedule are stored in a variable of type `ArrayList`, where the j -th element of the list corresponds to an integer array storing the nodes updated at instant j . This idea for the implementation was provided by Aurelien Naldi, one of the authors of `LogicalModel` and `GINSim`, in a personal communication with him. This strategy produces a performance gain in comparison with, for example, an array storing the update instant for each node -or the update function-, which implies computing the inverse of the update function each time we compute some transition, since we iterate over the instants (or the blocks).

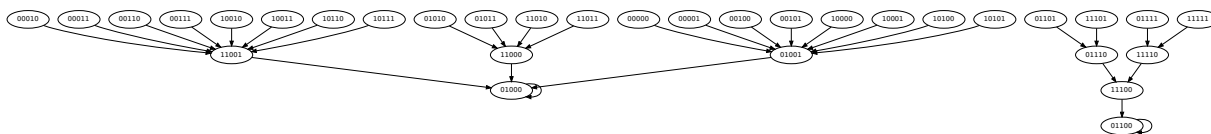
The user enters the schedule writing the sequence of update instants for each node separated by comma, in the same order the variables -or nodes- were specified in the network definition. For example, if we want to enter the schedule given by blocks sequence (1, 3)(2, 4, 5), for the same network in Example 1, we must write '1,2,1,2,2'. Notice that with this format, we restrict the entry only for block-sequential schedules, since the format requires to the user one and just one update instant for each node of the network.

The class `Net_simulator.java` contains the main method. It generates a simple window wrapping a CLI prompt which asks the user for the file storing the Boolean network and the schedule, which is ingressed manually with the conventions mentioned before. This win-

now was generated with Console class, from HSA package (this package was available at http://www.holtsoft.com/java/hsa_package.html, but nowadays is obsolete). The image below is an example of an interaction of the user with the application.



We see that this dialog requires to the user the file (including the root) storing the Boolean network, and the block-sequential schedule, which are coded using the conventions already explained. The application processes the entered data, and it finally outputs a .pdf file with a picture showing the respective state transition graph. Following our example, the output file is named `exam_3.1_1,2,1,2,2.pdf` and its content is the next image.



The picture is generated thanks to GraphViz, an open source graph visualization software (<http://www.graphviz.org/>), and `Net_simulator.java` can use it by means of GraphViz Java API, which can be downloaded from <http://www.loria.fr/~szathmar/off/projects/java/GraphVizAPI/index.php>. Thus, `Net_simulator.java` requires that GraphViz is installed in the machine. The code of `GraphViz.java` allows to select GraphViz directories depending if the code is run on Windows or Linux (currently is set to Linux).

Finally, the class `TestUpdaters.java` is an existing test class from LogicalModel library, which makes use of JUnit (<http://junit.org/>) to write the tests. `TestUpdaters.java` was updated with method `testBlockSequentialUpdater()` to test for the new functionality given by `BlockSequentialUpdater` class. `TestUpdaters.java` was written according to a set of guidelines (shared by A. Naldi) defined by the LogicalModel crew.

The end-user application is given by the executable file `simulador.jar`. This file contains all the classes and libraries necessary to run, and it requires the machine has installed JRE -Java Runtime Environment- (<http://www.java.com/es/download/>).

B.2 Computing the filter

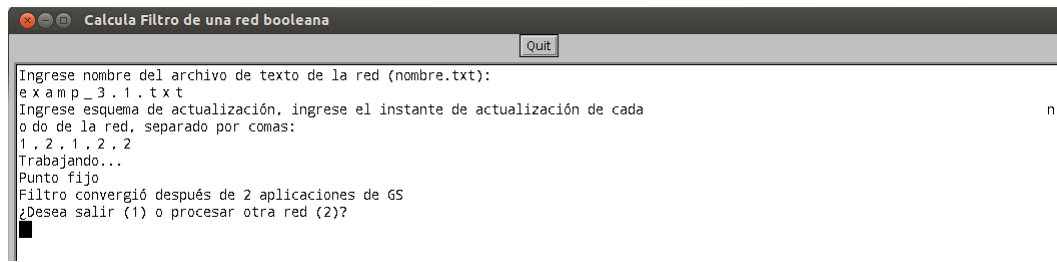
In order to implement the computation of the filter, four Java classes were written: `GS.java`, `Filtro.java`, `interfaz_filtro.java` and `TestGS.java`. The first of them serves to compute the Gauss-Seidel operator output on some Boolean network. It has three instance variables: the list of String's nodes, which stores the node ID's; the map `m_functions`, which stores, for

every node ID, the String coding the respective expression defining the function; and the map `GSeidel_functions`, which stores, for every node ID, the String coding the respective expression defining the function in the network outputted by Gauss-Seidel. This class has getter and setter methods for every one of the instance variables, except for `GSeidel_functions`, which cannot be setted externally.

The method `parse()` from class `GS` is intended to read a network from a `.txt` file and store it in the attributes `nodes` and `m_functions`. The method `getModel()` returns the `LogicalModel` associated to the network stored in the attributes `nodes` and `m_functions`. These methods are based in the methods of class `RawFunctionImport.java`, which serves to import a Boolean network (to an instance of `LogicalModel`) from a text file. The most important method of `GS.java` is `getGaussSeidelNet()`, which computes the output of Gauss-Seidel operator, stores it in the attribute `GSeidel_functions` (as String's) and returns it (as a `LogicalModel`) once finalized. This method receives as arguments an `ArrayList` `updt_sched` with the blocks of the schedule delivered to Gauss-Seidel (stored with the same format explained in last subsection), and an `int` array `updt_function`, which stores the respective update function (of the same schedule). This method passes over each node i in every block, looks for variables j of previous blocks on which the current node depends, and substitutes, in the expression defining variable i , the ID of variable j by the expression defining this variable (in the network outputted by Gauss-Seidel). These substitutions on a node i must be performed simultaneously (otherwise, spurious substitutions are produced), which is achieved by means of class `Pattern`, a representation of regular expressions. Finally, when the expressions of all Gauss-Seidel functions are computed, the `LogicalModel` associated to the output network is computed and returned by this method. The reason of why the operations of Gauss-Seidel are made on String's (and not on MDD's of the functions, for example), is mainly due to the lack of knowledge about manipulating networks with `mddlib`.

The class `TestGS.java` makes use of `JUnit` to test the functionality of class `GS`. The class `Filtro.java` has two instance variables: `operator` of type `GS`, and a list of `LogicalModel`'s named `redes`. The only method of this class is `calcula_Filtro`, which receives the name of the file storing input network, and the schedule (both the blocks and the update function); the method uses object `operator` to compute iteratively Gauss-Seidel and stops when the current network produced is equivalent to some network produced before. The previous networks are stored in variable `redes`. The comparison of networks is performed with the facilities of `LogicalModel` library (in particular, the class `LogicalModelComparator`), which in turn is based on `mddlib`. Additionally, `calcula_Filtro` writes every network produced in the process in a different text file, which name is coded as follows: "iterationNumber_updateFunction_inputNetworkName.txt". Finally, the method `calcula_Filtro` outputs a message informing the convergence of the filter into a fixed point or a cycle, its length and the number of iterations of Gauss-Seidel. The class `interfaz_filtro.java` provides the main method and a simple CLI prompt (produced in the same way seen in last subsection) which asks the user for the file storing the Boolean network and the schedule. This class has a method `generate_scheme()` to internally compute the blocks of the schedule from the update function entered by the user.

As an example, we can run the class `interfaz_filtro.java` with the same network and schedule from example in B.1, and get the image shown below.



```
Ingrese nombre del archivo de texto de la red (nombre.txt):
e x a m p _ 3 . 1 . t x t
Ingrese esquema de actualización, ingrese el instante de actualización de cada
nodo de la red, separado por comas:
1 , 2 , 1 , 2 , 2
Trabajando...
Punto fijo
Filtro convergió después de 2 aplicaciones de GS
¿Desea salir (1) o procesar otra red (2)?
```

The execution of `interfaz_filtro` (or the executable file `calcula_filtro.jar`) produced two files: “1_1,2,1,2,2_examp_3.1.txt” and “2_1,2,1,2,2_examp_3.1.txt”, and since the filter converged after two iterations to a fixed point (according to the message displayed), the two files store the same network specified by the following functions:

```
x1: (x4)
x2: 1
x3: (x2 & x3)
x4: (x2 & x3 & x5)
x5: (!x2)
```

Bibliography

- [1] S.B. Akers. Binary decision diagrams. *IEEE Trans Comp*, 1978.
- [2] R. Albert and H. G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *J. Theor. Biol.*, 2003.
- [3] G.E. Andrews. *Number Theory*. Dover Books on Mathematics Series. Dover Publications, 1994.
- [4] J. Aracena. Maximum number of fixed points in regulatory boolean networks. *Bull. Math. Biol.*, 2008.
- [5] J. Aracena, J. Demongeot, E. Fanchon, and M. Montalva. On the number of different dynamics in boolean networks with deterministic update schedules. *Mathematical Biosciences*, 242(2):188 – 194, 2013.
- [6] J. Aracena, E. Goles, A. Moreira, and L. Salinas. On the robustness of update schedules in boolean networks. *Biosystems*, 2009.
- [7] Julio Aracena, Mauricio González, Alejandro Zuniga, Marco A Mendez, and Verónica Cambiazo. Regulatory network for cell shape changes during drosophila ventral furrow formation. *J. Theor. Biol.*, 2006.
- [8] Anish Arora, Paul Attie, Michael Evangelist, and Mohamed Gouda. Convergence of iteration systems. *Distrib. Comput.*, 7(1):43–53, November 1993.
- [9] Mikhail J. Atallah. Finding the cyclic index of an irreducible, nonnegative matrix. *SIAM J. Comput.*, 11(3):567–570, 1982.
- [10] M. Barthélemy, A. Barrat, R. Pastor-Satorras, and A. Vespignani. Dynamical patterns of epidemic outbreaks in complex heterogeneous networks. *J. Theor. Biol.*, 2005.
- [11] V. Batagelij and A. Mrvar. Pajek - program for large network analysis. *Connections*, 1998.
- [12] G. Batt, B. Besson, P.-E. Ciron, H. de Jong, E. Dumas, J. Geiselmann, R. Monte, P.T. Monteiro, M. Page, F. Rechenmann, and D. Ropers. *Bacterial Molecular Networks*, chapter Genetic Network Analyzer : A tool for the qualitative modeling and simulation

of bacterial regulatory networks, pages 439–462. Humana Press, Springer, New York, 2012.

- [13] R.A. Brualdi and H.J. Ryser. *Combinatorial Matrix Theory*. Cambridge University Press, Cambridge, 1991.
- [14] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans Comp*, 1986.
- [15] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *Knowledge and Data Engineering, IEEE Transactions on*, 1(1):146–166, Mar 1989.
- [16] Claudine Chaouiya, Duncan Berenguier, Sarah M. Keating, Aurelien Naldi, Martijn P. van Iersel, Nicolas Rodriguez, Andreas Dräger, Finja Büchel, Thomas Cokelaer, Bryan Kowal, Benjamin Wicks, Emmanuel Gonçalves, Julien Dorier, Michel Page, Pedro T. Monteiro, Axel von Kamp, Ioannis Xenarios, Hidde de Jong, Michael Hucka, Steffen Klamt, Dennis Thieffry, Nicolas Le Novère, Julio Saez-Rodriguez, and Tomas Helikar. Sbm1 qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC Systems Biology*, 2013.
- [17] E. Clarke, D. Kröning, J. Ouaknine, and O. Strichman. Completeness and complexity of bounded model checking, in verification, model checking, and abstract interpretation lncs 2937. *Venice, Italy: Springer*, 2004.
- [18] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.
- [19] M. I. Davidich and S. Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. (*G. Stolovitzky, Ed.*) *PLoS ONE*, 2008.
- [20] Eric V. Denardo. Periods of connected networks and powers of nonnegative matrices. *Mathematics of Operations Research*, 2(1):20–24, 1977.
- [21] Y. Devloo, P. Hansen, and M. Labbé. Identification of all steady states in large networks by logical analysis. *Bull. Math. Bio.*, 2003.
- [22] Rolf Drechsler and Detlef Sieling. Binary decision diagrams in theory and practice. *International Journal on Software Tools for Technology Transfer*, 3(2):112–136, 2001.
- [23] Andreas Dräger, Nicolas Rodriguez, Marine Dumousseau, Alexander Dörr, Clemens Wrzodek, Nicolas Le Novère, Andreas Zell, and Michael Hucka. Jsbm1: a flexible java library for working with sbml. *Bioinformatics*, 27(15):2167–2168, 2011.
- [24] Elena Dubrova and Maxim Teslenko. A sat-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2011.

- [25] Plahte E., Mestl T., and Omholt S.W. Feedback loops, stability and multistationarity in dynamical systems. *J Biol Syst*, 1995.
- [26] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.
- [27] C. Farrow, J. Heidel, H. Maloney, and J. Rogers. Scalar equations for synchronous boolean networks with biological applications. *IEEE Trans. Neural Networks*, 2004.
- [28] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 2006.
- [29] Jr. Ferrell, J. E. and E. M. Machleder. The biochemical basis of an all-or-none cell fate switch in xenopus oocytes. *JScience*, 1998.
- [30] A. Garg, A. D. Cara, L. Xenarios, L. Mendoza, and G. D. Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 2008.
- [31] L. Glass. *Dynamical Systems and Cellular Automata*, chapter Boolean and continuous models for the generation of biological rhythms, pages 197–206. Academic Press, London, 1985.
- [32] L. Glass and J. Pasternack. Stable oscillations in mathematical models of biological control systems. *J. Math. Biol.*, 1978.
- [33] Leon Glass and Stuart A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39(1):103 – 129, 1973.
- [34] Leon Glass and Joel S. Pasternack. Prediction of limit cycles in mathematical models of biological oscillations. *Bulletin of Mathematical Biology*, 40(1):27 – 44, 1978.
- [35] E. Goles and S. Martínez. Neural and automata networks. *Mathematics and its Applications Ser., Kluwer Academic Publishers, Dordrecht*, 1991.
- [36] Eric Goles, Marco Montalva, and GonzaloA. Ruz. Deconstruction and dynamical robustness of regulatory networks: Application to the yeast cell cycle networks. *Bulletin of Mathematical Biology*, 75(6):939–966, 2013.
- [37] Eric Goles and Mathilde Noual. Disjunctive networks and update schedules. *Advances in Applied Mathematics*, 2012.
- [38] Eric Goles and Lilian Salinas. Comparison between parallel and serial dynamics of boolean networks. *Theoretical Computer Science*, 2008.
- [39] Eric Goles and Lilian Salinas. Sequential operator for filtering cycles in boolean networks. *Advances in Applied Mathematics*, 2010.
- [40] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford University Press, Oxford, sixth edition, 2008. Revised by D. R. Heath-Brown and J. H.

Silverman, With a foreword by Andrew Wiles.

- [41] S. E. Harris, B. K. Sawhill, A. Wuensche, and S. Kauffman. A model of transcriptional regulatory networks based on biases in the observed regulation rules. *Complexity*, 7:23–40, 2002.
- [42] I. Harvey and T. Bossomaier. Time out of joint: Attractors in asynchronous random boolean networks. *In: Proceedings of the Fourth European Conference on Artificial Life. MIT Press, Cambridge*, 1997.
- [43] J. Heidel, J. Maloney, J. Farrow, and J. Rogers. Finding cycles in synchronous boolean networks with applications to biochemical systems. *Int. J. Bifurcat. Chaos*, 2003.
- [44] H.A. Helfgott. Major arcs for goldbach’s theorem. *arXiv:1305.2897v4 [math.NT]*, 2013.
- [45] J. Holladay and R. Varga. On powers of non-negative matrices. *Proc. Amer. Math. Soc.*, 1958.
- [46] S. Huang. Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *J. of molecular medicine*, 1999.
- [47] S. Huang. *Regulation of cellular states in mammalian cells from a genome-wide view.* in Collado-Vides, J. and Hofestadt, R., Eds, ‘Gene Regulation and Metabolism: Post-Genomic Computational Approach’, MIT Press, Cambridge, MA., 2002.
- [48] D. J. Irons. Improving the efficiency of attractor cycle identification in boolean network. *Physica D*, 2006.
- [49] Demongeot J., Goles E., Morvan M., Noual M., and Sene S. Attraction basins as gauges of robustness against boundary conditions in biological complex systems. *PLoS ONE*, 2010.
- [50] Abdul Salam Jarrah, Blessilda Raposa, and Reinhard Laubenbacher. Nested canalizing, unate cascade, and polynomial functions. *arXiv:q-bio/0606013v3 [q-bio.QM]*, 2007.
- [51] Donald B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.
- [52] Winfried Just, Ilya Shmulevich, and John Konvalina. The number and probability of canalizing functions. *Physica D: Nonlinear Phenomena*, 197(3–4):211 – 221, 2004.
- [53] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Random boolean network models and the yeast transcriptional network. *PNAS*, 2003.
- [54] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetics nets. *J. Theor. Biol.*, 1969.
- [55] S.A. Kauffman. *The origins of order: self-organization and selection in evolution.* Oxford University Press, 1993.

- [56] M. Kaufman, C. Soulé, and R. Thomas. A new necessary condition on interaction graphs for multistationarity. *Journal of Theoretical Biology*, 248(4):675 – 685, 2007.
- [57] M.A. Kiwi, R. Ndoundam, M. Tchunte, and E. Goles. No polynomial bound for the period of the parallel chip firing game on graphs. *Theoretical Computer Science*, 136(2):527 – 532, 1994.
- [58] Ladner R. Klee V. and Manber R. Signsolvability revisited. *Linear Algebra and its Applications*, 1984.
- [59] Konstantin Klemm and Stefan Bornholdt. Stable and unstable attractors in boolean networks. *Phys. Rev. E*, 72:055101, Nov 2005.
- [60] C. J. Langmead, S. Jha, and E. M. Clarke. Temporal-logics as query languages for dynamic bayesian networks: Application to d. melanogaster embryo development cmucs-06-159. 2010.
- [61] Andrea S. Lapaugh and Christos H. Papadimitriou. The even-path problem for graphs and digraphs. *Networks*, 14(4):507–513, 1984.
- [62] Dmitriy Laschov and Michael Margaliot. Controllability of boolean control networks via the perron–frobenius theory. *Automatica*, 48(6):1218 – 1223, 2012.
- [63] C.Y. Lee. Representation of switching circuits by binary decision diagrams. *Bell Syst Tech J*, 1959.
- [64] Daniel S Levine and Manuel Aparicio. *Neural networks for knowledge representation and inference / edited by Daniel S. Levine, Manuel Aparicio IV*. Hillsdale, N.J. : Lawrence Erlbaum Associates, 1994. Includes bibliographical references and indexes.
- [65] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences of the United States of America*, 2004.
- [66] Yuan Li, John O. Adeyeye, David Murrugarra, Boris Aguilar, and Reinhard Laubenbacher. Boolean nested canalizing functions: a comprehensive analysis. *arXiv:1204.5203v2 [math.DS]*, 2013.
- [67] William J.R. Longabaugh, Eric H. Davidson, and Hamid Bolouri. Computational representation of developmental genetic regulatory networks. *Developmental Biology*, 283(1):1 – 16, 2005.
- [68] William J.R. Longabaugh, Eric H. Davidson, and Hamid Bolouri. Visualization, documentation, analysis, and communication of large-scale gene regulatory networks. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms*, 1789(4):363 – 374, 2009. `jc:titlejSystem Biology – Genetic Networksj/ce:titlej.`
- [69] Anna Lubiw. A note on odd/even cycles. *Discrete Applied Mathematics*, 22(1):87 – 92, 1988.

- [70] Alberto O. Mendelzon and Peter T. Wood. Finding regular simple paths in graph databases. In *VLDB* [], pages 185–193.
- [71] L. Mendoza and E. Alvarez-Buylla. Dynamics of the genetic regulatory network for arabidopsis thaliana flower morphogenesis. *J. of Theoretical Biology*, 1998.
- [72] Henryk Minc. *Nonnegative matrices*. John Wiley & Sons, New York,, 1988.
- [73] P. Montealegre and E. Goles. Computational complexity of majority automata under different updating schemes. In: *Proceedings of the nineteenth International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2013) – Exploratory Papers*, Edited by Jarkko Kari, Martin Kutrib, Andreas Malcher, 2013.
- [74] Bernard M. E. Moret. Decision trees and diagrams. *ACM Comput. Surv.*, 14(4):593–623, December 1982.
- [75] David Murrugarra and Reinhard Laubenbacher. Regulatory patterns in molecular interaction networks. *Journal of Theoretical Biology*, 288(0):66 – 72, 2011.
- [76] Christoph Müssel, Martin Hopfensitz, and Hans A. Kestler. Boolnet—an r package for generation, reconstruction and analysis of boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.
- [77] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya. Logical modelling of regulatory networks with {GINsim} 2.3. *Biosystems*, 97(2):134 – 139, 2009.
- [78] J. Paulsson, O. G. Berg, and M. Ehrenberg. Stochastic focusing: Fluctuation- enhanced sensitivity of intracellular regulation. *Proc. Natl. Acad. Sci. USA*, 2000.
- [79] Thomas R. On the relation between the logical structure of systems and their ability to generate multiple steady states of sustained oscillations. *Springer Series Synergetics*, 1981.
- [80] Thomas R. Positive feedback circuits are a necessary conditions for positive eigenvalues of the jacobian matrix. *Ber Bungenes Phys Chem*, 1994.
- [81] F. Robert. Discrete iterations. *Springer S. in Computational Mathematics*, 1986.
- [82] M. Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67 – 72, 1981.
- [83] C. Soulé. Graphic requirements for multistationarity. *ComPlexUs*, 2003.
- [84] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.
- [85] Denis Thieffry. Dynamical roles of biological regulatory circuits. *Briefings in Bioinformatics*, 8(4):220–225, 2007.

- [86] Carsten Thomassen. Even cycles in directed graphs. *European Journal of Combinatorics*, 6(1):85 – 89, 1985.
- [87] J. J. Tyson, K. Chen, and B. Novak. Network dynamics and cell physiology. *Nature Rev Mol Cell Biol*, 2001.
- [88] Richard S. Varga. *Matrix Iterative Analysis*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1962.
- [89] C. H. Waddington. Canalisation of development and the inheritance of acquired characters. *Nature*, 150:563–564, 1942.
- [90] K. Yang and Q. Zhao. Balance problem of min-max systems is co-np hard. *Syst. Contr. Lett.*, 2004.
- [91] S. Q. Zhang, M. Hayashida, T. Akutsu, W. K. Ching, and M. K. Ng. Algorithms for finding small attractors in boolean networks. *EURASIP J. Bioinf. Syst. Bio.*, 2007.
- [92] Q. Zhao. A remark on “scalar equations for synchronous boolean networks with biological applications” by c. farrow, j. heidel, j. maloney and j. rogers. *IEEE Trans. Neural Netw.*, 2005.
- [93] Desheng Zheng, Guowu Yang, Xiaoyu Li, Zhicai Wang, Feng Liu, and Lei He. An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks. *PLoS ONE*, 8(4):e60593, 04 2013.