Universidad de Concepción
Dirección de Postgrado
Concepción-Chile

Dinámica de redes discretas con esquemas de actualización deterministas. Aplicación a las redes de regulación génica.

(Dynamics of discrete networks with deterministic updates schedules.
Application to genetic regulatory networks.)

Tesis para optar al grado de Doctor en Ciencias Aplicadas con mención en Ingeniería Matemática.

Luis Miguel Gómez Guzmán

Enero-2015

Dinámica de redes discretas con esquemas de actualización deterministas. Aplicación a las redes de regulación génica.

Luis Gómez Guzmán

Profesor Guia: Dr. Julio Aracena Lucero, Departamento de Ingeniería Matemática, Facultad de Ciencias Físicas y Matemáticas, Universidad de Concepción, Chile.

Co-tutor: Dr. Jaques Demongeot, Laboratorio AGIM, Facultad de Medicina, Universidad de Grenoble, Francia.

Co-tutor: Dr. Lilian Salinas Ayala, Departamento de Informática y Ciencias de la Computación, Facultad de Ingeniería, Universidad de Concepción, Chile.

Director de Programa: Dr. Raimund Bürger, Universidad de Concepción, Chile

COMISIÓN EVALUADORA

Dr. Gilles Bernot, Université de Nice, Francia.

Dr. Alejandro Maass, Universidad de Chile.

Dr. Gonzalo Ruz, Universidad Adolfo Ibañez.

COMISIÓN EXAMINADORA

Firma: _____
Dr. Adrien Richard.
Université de Nice, Francia.

Firma: _____
Dr. Alejandro Maass.
Universidad de Chile, Chile.

Firma: _____
Dr. Gonzalo Ruz.
Universidad Adolfo Ibañez, Chile.

Firma: _____
Dr. Jacques Demongeot.
Université de Grenoble, Francia.

Firma: _____

Dra. Lilian Salinas.
Universidad de Concepción, Chile.

Firma: _____

Dr. Julio Aracena.
Universidad de Concepción, Chile.

Fecha Examen de Grado: _____

Calificación: _____

Concepción–Enero de 2015

Agradecimientos.

En primer lugar, agradezco la gentileza, preocupación y dedicación brindada por mis directores de tesis: Dr. Julio Aracena, Dr. Jacques Demongeot y la Dra. Lilian Salinas. Sin lugar a dudas unos excelentes profesionales, quienes me han brindado su apoyo incondicional en este largo camino, a pesar de las adversidades.

Por el apoyo financiero, agradezco a la Comisión Nacional de Investigación Científica y Tecnológica (CONICYT) a través de los proyectos FONDECYT No. 1090549 y No. 113103, y a través de la Beca de Cotutela Doctoral en el Extranjero. Y a MECESUP a través del proyecto UCO 0713.

A los miembros de la comisión evaluadora y examinadora, por su buena voluntad, interés por el tema. tiempo y dedicación en la revisión de este trabajo.

A mis padres, por su cariño y apoyo durante estos 29 años. A mis suegros por su paciencia durante estos últimos 6 años.

A mi esposa Gabriela, por iluminar estos últimos años de mi vida, por darme su apoyo incondicional permaneciendo conmigo en todas y por darnos lo mas preciado que tenemos, nuestros hijos Amara y Eluney. A mi hija Amara por su alegría inagotable y a mi hijo Eluney por nacer y hacernos felices a pesar de las dificultades.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. English version

A Boolean network is a system of n interacting Boolean variables, which evolve, in a discrete time, according to a predefined rule. They have applications in many areas, including circuit theory, computer science and social systems (Green et al., 2007; Tocci and Widmer, 2001). In particular, from the seminal works of Kauffman (1969, 1993) and (Thomas, 1973; Schaefer, 1978), they are extensively used as models of gene regulatory networks. Despite their simplicity, they provide a useful model in which different phenomena can be reproduced and studied, and indeed, many regulatory models published in the biological literature fit within their framework (Huang, 1999; Shmulevich et al., 2003; Fauré et al., 2006; Bornholdt, 2008). In this context, Boolean networks give a first idea of the qualitative dynamics of a gene regulatory network represented by the temporal evolution of the gene and protein states. They are used to investigate the organizational principles of a network and how this influences its robustness. The reconstruction of gene regulatory networks using Boolean networks has several advantages. They can effectively represent realistic complex biological phenomena (Albert and Othmer, 2003; Ciliberti et al., 2007; Abou-Jaoudé et al., 2009, 2010; Chaves et al., 2010; Veliz-Cuba and Stigler, 2011). Furthermore, the discretization to binary values simplifies the obtained models by reducing the noise level in experimental data as well as simulations in computers are easy to handle.

Since Boolean networks have a finite number of states, the long-run dynamic trajectories always reach a periodic sequence of states, called attractor. When the period is one, the attractor is said to be a fixed point, and when the period is greater than one, it is called limit cycle. In the modeling of genetic regulatory networks, the attractors are associated to distinct types of cells defined by patterns of gene activity. In particular, the limit cycles are often associated with the cell cycle (Huang, 1999; Fauré et al., 2006).

The update schedule in a Boolean network, that is the order in which each node is updated, is of great importance in the dynamics of the network. In general and probably due to the difficulty of really knowing the order (if any) in which events take place in the cell,

regulatory networks are usually studied with synchronous schedule (parallel scheme). One reason for determinism is the need to model some periodical behaviors; when randomness is introduced, attractors became regions of the phase space, but are no longer exact dynamical cycles (Aracena et al., 2009). However, many other update schedules with a certain level of asynchronism have been used in Boolean networks to model different biological systems (Thomas, 1973; Chaves et al., 2005; Mendoza and Alvarez-Buylla, 1998; Albert and Othmer, 2003; Hansson et al., 2005). Other types of deterministic update schedules, introduced by Robert (1986, 1995), and used in the discrete modeling of genetic regulatory networks (see Ruz et al. (2014); Goles et al. (2013); Meng and Feng (2014)) and other dynamical systems are: the sequential update (nodes are updated one by one in a prescribed order) and block-sequential updates (which are sequential over the sets of a partition, but parallel inside of each set).

The change in the update schedule of a Boolean network can yield variations on the dynamical behavior of the network, in particular on the set of attractors. The robustness of Boolean networks against perturbations of their update schedule has been studied mainly from an experimental and statistical point of view (Elena, 2009; Chaves et al., 2005; Demongeot et al., 2008; Fauré et al., 2006; Goles and Noual, 2010; Christoph Schmal and Drossel, 2010).

There are many theoretical and analytical studies about the effect of changing the deterministic update schedule in Boolean networks. Some of the pioneering works in this context were done by Robert (1986, 1995), who projected continuous tools into the discrete problem, and Goles (1980), who studied the dynamics of discrete neural networks with deterministic update schedules. Some other studies were done in Mortveit and Reidys (2001), where it was studied the set of update schedules preserving the whole dynamical behavior of a special class of discrete dynamical networks, called sequential dynamical systems, where the interaction digraph is symmetric or equivalently an undirected graph and the update schedule is sequential. In Goles and Noual (2012) disjunctive networks are classified according to the robustness of their dynamics with respect to changes in the update schedule. In Elena (2009), is given an analytical classification of Boolean networks according to their dynamical behavior. Also, in Salinas (2008) and Aracena et al. (2009), were given equivalence classes of update schedules, based on their associated update digraphs, such that elements in the same class, yield the same dynamical behavior. Furthermore, in Aracena et al. (2013a), was studied how many different dynamics can exist in a Boolean network when the update schedule is changed.

Since the fixed points are invariant to the update schedule, the interesting problem when analyzing the attractors, is focused on the limit cycles. In this context, there are also several theoretical and analytical studies that have been done when different update schedules are used. Most of them show that the limit cycles are very sensitive to changes in the update schedule of the network. In particular, Demongeot et al. (2008) studied the role of the update schedule on the asymptotic behavior of gene regulatory networks, essentially on the occurrence of limit cycles. Ruz and Goles (2013) used the swarm intelligence technique to analyze the ability of the networks to preserve the attractors when the updating schemes are changed

from parallel to sequential. Also, Goles and Salinas (2008) did a comparative analysis on the attractors in Boolean networks with parallel and sequential update schedules. They proved that both schemes cannot have limit cycles in common. Also, Macauley and Mortveit (2009) give limit cycle equivalence classes (isomorphic as directed graphs) in sequential dynamical systems.

One of the major problems in the understanding of the function of many biological complex systems, such as genetic networks or molecular signaling pathways, is the inference of the network with a given update schedule from observed data, as for example a limit cycle. In this sense, the reconstruction of a genetic regulatory network has been so far done considering mainly synchronous update (see for example Shmulevich et al. (2002); Akutsu et al. (1999)). However, there are limit cycles, under sequential or block-sequential schedules, which cannot be yielded with parallel update (Goles and Salinas, 2008).

In this work, we focus our study on different problems related to limit cycles and deterministic updated schedules, which can be divided in two main topics, robustness and inverse problems. In the first one, we are interested in whether a given limit cycle (or the full set of limit cycles) of a given Boolean network is kept when the update schedule is changed. And in the second topic we are interested in, given a dynamical characteristic and Boolean function, determining whether there is an update schedule such that the Boolean function updated under it presents the characteristic in its dynamical behavior. In this context, we explore families of Boolean networks with different types of local activation functions and structural properties of the interaction digraph, to define the sharp delineation of the algorithmic complexity for the several problems that arise.

This thesis is organized as follows. In Chapter 2, we introduce all definitions and notation that are used through the following chapters.

In Chapter 3 we study the robustness of limit cycles of two given Boolean networks that differ only in their update schedules. Since in Aracena et al. (2009), was shown that if two update schedules belong to the same class (equal update digraphs) then the respective Boolean networks have the same dynamical behabior, we are interested in studying different classes yielding the same limit cycle set. In this context, we show that the related decision problems are NP-Hard (that is, intractable computationally problems) and that the information provided by the update digraph is not sufficient to determine whether two given Boolean networks share limit cycles or not. Besides, we give a polynomial algorithm that works as a necessary condition for sharing limit cycles for two given update digraphs: if this algorithm returns TRUE, then no matter the Boolean function used, they will never share any limit cycle. If the algorithm returns FALSE, then we give some sufficient conditions to construct a Boolean function that will have a common limit cycle for the corresponding Boolean networks. Also, we give some sufficient conditions which allows us to, given a Boolean network, construct a non equivalent update schedule that share a given limit cycle of the given Boolean network.

In Chapter 4 we study some problems about the existence of update schedules which yield limit cycles. Here, we prove that the problem of existence is NP-Hard even for AND-

OR functions or symmetric interaction digraph. Nevertheless, we show that the problem is polynomial if both conditions are satisfied. Besides, we prove that such an update schedule exists if and only if the limit cycle set of the network updated in parallel is non empty. Furthermore, this last condition is characterized by a property in the interaction digraph that can be verified in polynomial time. For the problem of deciding the existence of an update schedule that does not generate limit cycles, we prove that the general case is NP-Hard, and that there always exists such an update schedule in the case of OR functions.

In Chapter 5, we are interested in studying the problem of, given a Boolean function and a sequence of global state vectors, whether there exists an update schedule that generates the given sequence as a limit cycle when the Boolean function is updated under it. In first place, we deal with a more basic problem: a single transition. We prove that this problem is NP-Complete even for OR functions. In this case, we give a characterization of the existence of solution in terms of the interaction digraph and an algorithm to test it, this algorithm being polynomial in the symmetric case. About the sequence problem, we prove that it is also NP-Complete even for OR functions. Besides, we show that the symmetric OR case and the case where the length of the limit cycle is equal to two are both polynomial. We also study another related problems and we prove that all of them are NP-Complete even in the OR case.

In Chapter 6, we apply the results of Chapter 3 for studying the robustness of the limit cycles of two genetic regulatory networks when the update schedule is changed: the logical version of the mammal cell cycle network and the fission yeast cell-cycle network, introduced in Fauré et al. (2006) and Davidich and Bornholdt (2008), respectively. Some theoretical and simulation results were given about the robustness of their dynamical behaviors against changes in the update schedule in Ruz et al. (2014) and Goles et al. (2013), respectively. For each network we construct non equivalent update schedules that yield the same attractors as the networks synchronously updated.

Finally, in Appendix A are given some explicit algorithms from the results of Chapters 4 and 5.

1.2. Spanish version

Una red Booleana es un sistema de n variables Booleanas que interactúan y evolucionan, en tiempo discreto, de acuerdo a una regla predefinida. Tienen aplicaciones en diversas áreas, incluyendo la teoría de circuitos, ciencias de la computación y sistemas sociales (Green et al., 2007; Tocci and Widmer, 2001). En particular, de los trabajos seminales de Kauffman (1969, 1993) y (Thomas, 1973; Schaefer, 1978), son usadas extensamente como modelos de redes de regulación génica. A pesar de su simplicidad, ella proporcionan un modelo útil en el cual diferentes fenómenos pueden ser reproducidos y estudiados, y de hecho, muchos modelos regulatorios en la literatura biológica calzan en este marco (Huang, 1999; Shmulevich et al., 2003; Fauré et al., 2006; Bornholdt, 2008). En este contexto, las redes Booleanas dan una primera impresión de la dinámica cualitativa de una red de regulación génica representada por la evolución temporal de los estados de las proteínas. Ellas son usadas para investigar los principios organizacionales de una red y como ésta afecta a su robustez. La reconstrucción de redes génicas regulatorias tiene varias ventajas, puesto que pueden representar efectivamente fenómenos biológicos complejos reales (Albert and Othmer, 2003; Ciliberti et al., 2007; Abou-Jaoudé et al., 2009, 2010; Chaves et al., 2010; Veliz-Cuba and Stigler, 2011). Mas aún, la discretización a estados binarios simplifica los modelos obtenidos puesto que reduce el nivel de ruido de los datos experimentales y también las simulaciones conputacionales son fáciles de manejar.

Puesto que las redes Booleanas tienen un número finito de estados, las trayectorias dinámicas de largo plazo siempre alcanzan una secuencia de estados periódicos, llamada atractor. Cuando el periódo es uno, el atractor se dice punto fijo, y cuando el período es mayor que uno, se llama ciclo limite. En el modelamiento de redes de regulación génica, los atractores están asociados a distintos tipos de células definidos por patrones en la actividad génica. En particular, los ciclos limites son usualmente asociados con ciclos celulares (Huang, 1999; Fauré et al., 2006).

El esquema de actualización de una red Booleana, que es el orden en el cual cada nodo es actualizado, es de gran importancia en la dinámica de la red. En general y debido probablemente a la dificultad de conocer el orden (si es que alguno existe) en el que los eventos tienen lugar en la célula, las redes regulatorias son usualmente estudiadas con un esquema sincrónico (o paralelo). Una razón para el determinismo es la necesidad de modelar algunos comportamientos periódicos; cuando la aleatoriedad es introducida, los atractores forman regiones del espacio fase, pero ya no son ciclos dinámicos exactos (Aracena et al., 2009). Sin embargo, esquemas de actualización con algún nivel de asincronismo han sido utilizados en redes Booleanas para modelar distintos sistemas biológicos (Thomas, 1973; Chaves et al., 2005; Mendoza and Alvarez-Buylla, 1998; Albert and Othmer, 2003; Hansson et al., 2005). Otros tipos de esquemas de actualización deterministas, introducidos por Robert (1986, 1995), y usados en la modelación discreta de redes génicas regulatorias (ver Ruz et al. (2014); Goles et al. (2013); Meng and Feng (2014)) y otros tipos de sistemas dinámicos son: el esquema secuencial (los nodos son actualizados uno por uno en un orden preestablecido) y bloque-secuenciales (que son secuenciales sobre los conjuntos de una partición, pero paralelos

dentro de cada conjunto).

El cambio en el esquema de actualización de una red Booleana puede producir variaciones en el comportamiento dinámico de la red, y en particular en el conjunto de atractores. La robustez de las redes Booleanas contra perturbaciones en el esquema de actualización a sido estudiado principalmente desde un punto de vista experimental y estadístico (Elena, 2009; Chaves et al., 2005; Demongeot et al., 2008; Fauré et al., 2006; Goles and Noual, 2010; Christoph Schmal and Drossel, 2010).

Hay varios estudios teóricos y analíticos acerca del efecto de cambiar el esquema de actualización determinista de una red Booleana. Algunos de los trabajos pioneros en este contexto fueron hechos por Robert (1986, 1995), quien proyectó herramientas del análisis continuo al problema discreto, y Goles (1980), quien estudio la dinámica de redes neuronales discretas con esquemas de actualización deterministas. Otros estudios fueron hechos en Mortveit and Reidys (2001), donde se estudio el conjunto de esquemas de actualización que preservan el comportamiento dinámico completo de una clase especial de redes dinámicas discretas, llamada sistemas dinámicos secuenciales, en donde el dígrafo de interacción es simétrico o equivalentemente un grafo no dirigido, y el esquema es secuencial. En Goles and Noual (2012), las redes disyuntivas son clasificadas de acuerdo a la robustez de su dinámica con respecto a cambios en el esquema de actualización. En Elena (2009), se da una clasificación analítica de las redes Booleanas de acuerdo a su comportamiento dinámico. A su vez, en Salinas (2008) y Aracena et al. (2009), son definidas clases de equivalencia de esquemas de actualización, basado en su dígrafo de actualización, de forma tal que elementos en la misma clase tiene el mismo comportamiento dinámico. Mas aún, en Aracena et al. (2013a), se estudiaron cuantas dinámicas diferentes pueden existir en una red Booleana cuando el esquema de actualización es cambiado.

Puesto que los puntos fijos son invariantes al esquema de actualización, el problema interesante al analizar los atractores esta centrado en los ciclos limites. En este contexto, también hay varios estudios teóricos y analíticos que se han hecho cuando diferentes esquemas de actualización son utilizados. La mayoría de ellos muestra que los ciclos limites son muy sensibles a cambios en el esquema de actualización de la red. En particular, Demongeot et al. (2008) estudiaron el rol del esquema de actualización en el comportamiento asintótico de redes de regulación génica, esencialmente en la ocurrencia de ciclos limites. Ruz and Goles (2013) usaron técnicas de inteligencia de enjambre para analizar la habilidad de las redes Booleanas para preservar los atractores cuando el esquema de actualización es cambiado desde el paralelo al secuencial. También, Goles and Salinas (2008) hicieron una análisis comparativo de los atractores en redes Booleanas con los esquemas paralelo y secuencial y probaron que ambos esquemas no pueden tener ciclos limites en común. Por otra parte, Macauley and Mortveit (2009) dan clases de equivalencias de ciclos limites (isomorfas como grafos dirigidos) en sistemas dinámicos secuenciales.

Uno de los mayores problemas en la comprensión de la función de muchos sistemas biológicos complejos, tales como redes de regulación génica o vías de señalización moleculares, es la inferencia de la red con un esquema de actualización dado a partir de datos observados,

como por ejemplo un ciclo limite. En este sentido, la reconstrucción de una red de regulación génica a sido hasta ahora hecho principalmente considerando el esquema sincrónico (ver por ejemplo Shmulevich et al. (2002); Akutsu et al. (1999)). Sin embargo, hay ciclos limites, bajo esquemas secuenciales o bloque-secuenciales, que no pueden ser generados por el esquema paralelo (Goles and Salinas, 2008).

En este trabajo, nos enfocamos en estudiar diferentes problemas relacionados con ciclos limites y esquemas de actualización deterministas, que pueden ser divididos en dos temas. En el primero, nos interesa saber si un ciclo limite dado (o el conjunto total de ciclos limites) de una red Booleana es mantenido cuando el esquema de actualización es cambiado. Y en el segundo nos interesa, dada una característica dinámica de una función Booleana, determinar si existe un esquema de actualización de forma tal que la función Booleana actualizada bajo este esquema presenta la característica en su comportamiento dinámico. En este contexto, exploramos familias de redes Booleanas con distintos tipos de funciones locales de activación y propiedades estructurales del dígrafo de interacción, para definir la delgada delineación de la complejidad algorítmica de los diversos problemas que surgen.

Esta tesis esta organizada como sigue. En el Capítulo 2, introducimos todas las definiciones y notación que utilizaremos en los capítulos siguientes.

En el Capítulo 3 estudiamos la robustez de los ciclos limites de dos redes Booleanas dadas que solo difieren en su esquema de actualización. Puesto que en Aracena et al. (2009), fue mostrado que si dos esquemas de actualización pertenecen a la misma clase (digrafos de actualización iguales) entonces las redes Booleanas respectivas tienen el mismo comportamiento dinámico, estamos interesados en estudiar distintas clases que generen el mismo conjunto de ciclos limites. En este contexto, mostramos que los problemas de decisión relacionados son todos NP-Hard (esto es, problemas computacionalmente intratables) y que la información entregada por el dígrafo de actualización no es suficiente para determinar si dos redes Booleanas dadas comparten ciclos limites o no. Además, entregamos un algoritmo polinomial que funciona como una condición necesaria para compartir ciclos limites para dos dígrafos de actualización dados: si este algoritmos retorna VERDADERO, entonce sin importar la función Booleana usada, ellos nunca van a compartir ningún ciclo limite. Si el algoritmo retorna FALSO, entonces damos algunas condiciones suficientes para construir una función Booleana que tendrá un ciclo limite en común para las redes Booleanas respectivas. A su vez, damos algunas condiciones suficientes que nos permite, dada una red Booleana, construir un esquema de actualización no equivalente que comparte un conjunto de ciclos limites dado de la red Booleana dada.

En el Capítulo 4 estudiamos algunos problemas relacionados con la existencia de esquemas de actualización que producen ciclos limites. Aquí, probamos que el problema de existencia es NP-Hard incluso para funciones AND-OR o con grafo de interacción simétrico. Sin embargo, mostramos que el problema es polinomial si ambas condiciones se cumplen. A su vez, probamos que tal esquema de actualización existe si y solo si el conjunto de ciclos limites de la red actualizada en paralelo es no vacío. Mas aún, esta ultima condición esta caracterizada por una propiedad del dígrafo de interacción que puede ser verificada en tiem-

po polinomial. Para el problema de decidir la existencia de un esquema de actualización que no genere ciclos limites, probamos que el caso general es NP-Hard y que siempre existe tal esquema en el caso de funciones OR.

En el Capítulo 5, estamos interesados en estudiar, dada una función Booleana y una secuencia de vectores estados globales, si existe un esquema de actualización que genere la secuencia dada como un ciclo limite cuando la función Booleana es actualizada bajo él. En primer lugar, tratamos con un problema mas básico: una transición. Probamos que este problema es NP-Completo incluso para funciones OR. En este caso, damos una caracterización de la existencia de solución en términos del dígrafo de interacción y un algoritmo para testearlo, que se vuelve polinomial en el caso simétrico. Para el problema de la secuencia, también probamos que es NP-Completo para funciones OR. Además, mostramos que el caso OR simétrico y cuando el largo del ciclo limite es dos son ambos polinomiales. También estudiamos otros problemas relacionados y probamos que todos son NP-Completos incluso en el caso OR.

En el Capítulo 6, aplicamos los resultados del Capítulo 3 para estudiar la robustez de los ciclos limites de dos redes de regulación génica cuando el esquema de actualización es cambiado: La versión lógica de las redes del ciclo celular mamífero y del ciclo celular de la levadura de fisión, introducidas en Fauré et al. (2006) y Davidich and Bornholdt (2008), respectivamente. Algunos resultados teóricos y de simulación son dados acerca de la robustez de sus comportamiento dinámico contra cambios en el esquema de actualización en Ruz et al. (2014) y Goles et al. (2013), respectivamente. Para cada red, construimos esquemas no equivalentes que producen los mismos atractores que las redes iteradas sincrónicamente.

Finalmente, en el Apéndice A son dados algunos algoritmos explícitos de los resultados de los Capítulos 4 y 5.

Chapter 2

Definitions and Notation

Let V be a set of n elements. We denote a function $F = (f_v)_{v \in V} : \{0,1\}^n \to \{0,1\}^n$, where each component function, $f_v : \{0,1\}^n \to \{0,1\}$, is a Boolean function, and such that $\forall x \in \{0,1\}^n$, $\forall v \in V \colon F(x)_v = f_v(x)$.

Given $x = (x_v)_{v \in V} \in \{0,1\}^n$ and $u \in V$, we define $\bar{x}^u \in \{0,1\}^n$ as:

$$\forall v \in V \colon \bar{x}^u_v = \begin{cases} x_v & \text{if } v \neq u \\ \neg x_u & \text{if } v = u \end{cases}$$

Where $\forall a \in \{0,1\} : \neg a = 1 \iff a = 0$.

We also define $\bar{x} \in \{0,1\}^n$ as: $\forall v \in V,\ \bar{x}_v = \neg x_v$.

A *Boolean network* $N = (F, s)$ is defined by a finite set V of n elements; n state variables $x_v \in \{0,1\},\ v \in V$; a function $F = (f_v)_{v \in V}$ called *global activation function*, where its component functions f_v are called *local activation functions*, and an *update schedule* defined by a function $s : V \to \{1, \ldots, n\}$ such that $s(V) = \{1, \ldots, m\}$ for some $m \leq n$. A *block* of an update schedule s is a set $B_i = \{v \in V : s(v) = i\}$, $1 \leq i \leq m$. An update schedule s is also denoted by $s = \{v \in B_1\}\{v \in B_2\} \cdots \{v \in B_m\}$. A *synchronous* or *parallel* update is given by an update schedule s such that $\forall v \in V$, $s(v) = 1$. A *sequential* update corresponds to a bijective function. Other kinds of update schedules can be considered as *block-sequential* updates. Block-sequential updates schedules were introduced in Robert (1986).

The iteration of the Boolean network with an update function s is given by:

$$x_v^{k+1} \quad = \quad f_v(x_u^{l_u} \colon u \in V)$$

where $l_u = k$ if $s(v) \leq s(u)$ and $l_u = k + 1$ if $s(v) > s(u)$.

This is equivalent to applying a function $F^s : \{0,1\}^n \to \{0,1\}^n$ in a parallel way, with $F^s(x) = (f_v^s(x))_{v \in V}$ defined by:

$$f_v^s(x) = f_v(g_{v,u}^s(x) \colon u \in V),$$

where the function $g^s_{v,u}$ is defined by $g^s_{v,u}(x) = x_u$ if $s(v) \leq s(u)$ and $g^s_{v,u}(x) = f^s_u(x)$ if $s(v) > s(u)$. Thus, the function F^s corresponds to the dynamical behavior of the network $N = (F, s)$. We note that F^s was called Serial-Parallel operator in Robert (1986), and in the particular case of sequential updates it was called Gauss-Seidel operator.

We say two Boolean networks $N_1 = (F_1, s_1)$ and $N_2 = (F_2, s_2)$ have the same dynamical behavior if $F_1^{s_1} = F_2^{s_2}$.

Since $\{0,1\}^n$ is a finite set, we have two limit behaviors for the iteration of a network:

- *Fixed Point.* We define a fixed point as $x \in \{0,1\}^n$ such that $F^s(x) = x$.
- *Limit Cycle.* We define a cycle of length $p > 1$ as the sequence $[x^k]_{k=0}^p = [x^0, \ldots, x^{p-1}, x^0]$ such that $x^k \in \{0,1\}^n$, x^k are pairwise distinct and $F^s(x^k) = x^{k+1}$, for all $k = 0, \ldots, p-1$ and $x^p \equiv x^0$. The set of limit cycles of N is denoted by $LC(N)$.

Fixed points and limit cycles are called *attractors* of the network.

We say that a node is *frozen* for a limit cycle if its state is constant on it.

Given a digraph G, the node set of G is referred to as $V(G)$, and its arc set as $A(G)$. An arc $(v, v) \in A(G)$ is called a *loop* of G. Given a node $v \in V(G)$, the set of incoming nodes to v is denoted as $N_G^-(v) = \{u \in V(G) \colon (u, v) \in A(G)\}$. Analogously, the set of outgoing nodes from v is denoted as $N_G^+(v) = \{u \in V(G) \colon (v, u) \in A(G)\}$. We denote the maximum in-degree the digraph as $\Delta^-(G) = \max_{v \in V(G)} \left|N_G^-(v)\right|$. The set of reachable nodes from v is defined as $R_G^+(v) = \{u \in V \colon \text{there exists a path from } v \text{ to } u \text{ in } G\}$. The set of nodes that reach v is defined as $R_G^-(v) = \{u \in V \colon \text{there exists a path from } u \text{ to } v \text{ in } G\}$.

Given $U \subseteq V(G)$, $G[U]$ is the digraph obtained from G by removing all nodes in $V(G) \setminus U$ and all arcs incoming to or outgoing from these nodes. $G[U]$ is called the *sub-digraph generated* by U.

Given a strongly connected digraph G, the *index of cyclicity* of G, denoted $\rho(G)$, is defined as the greatest common divisor of the lengths of the cycles of G. If the digraph has non trivial strongly connected components $\{G_i\}_{i=1}^m$, then the index of cyclicity is defined as $\rho(G) = \operatorname{lcm}\{\rho(G_i) \colon 1 \leq i \leq m\}$, or zero if it does not have any cycles. A strongly connected component is not trivial if it has at least one arc. $\rho(G)$ was defined in Jarrah et al. (2010) and it was called the *loop number* of G and it was proved that can be calculated in polynomial time. $\rho(G)$ correspond to the *index of cyclicity* of G as in Schutter and Moor (2000) and to the *index of imprimitivity* of the adjacency matrix of G as in Brualdi and Ryser (1991) and Berman and Plemmons (1994). We say that G is *primitive* if $\rho(G) = 1$. The *gird* of a digraph G, denoted $g(G)$, is defined as the length of the shortest cycle in G. We denote by $\mathcal{L}(C)$ the length of a cycle C of G.

The digraph associated to a Boolean function $F = (f_v)_{v \in V}$, called *interaction digraph*, is the directed graph $G^F = (V, A)$, where $(u, v) \in A$ if and only if f_v depends on x_u, i.e., if there

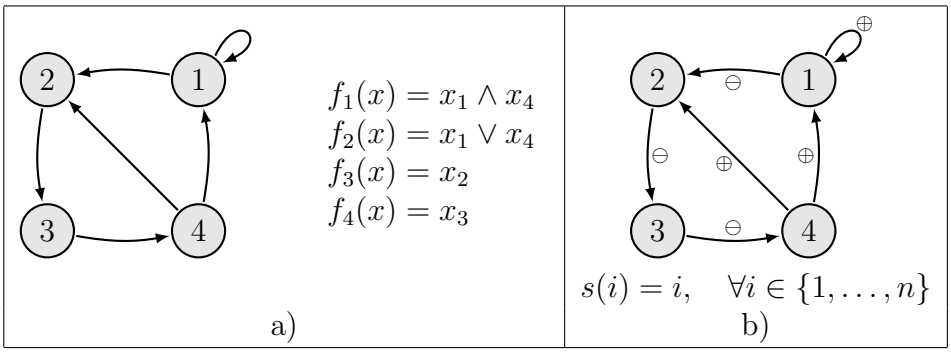


Figure 2.1: a) Digraph associated to a Boolean network. b) Update Digraph associated to a Boolean network and an update schedule.

exists $x \in \{0,1\}^n$ such that $f_v(x) \neq f_v(\bar{x}^u)$. Note that if f_v is constant, then $N^-_{G^F}(v) = \emptyset$. See an example of a interaction digraph in Figure 2.1.

Given $G = (V, A)$ a digraph with node set V of n elements and $s : V \to \{1, \ldots, n\}$ an update schedule, we denote $G_s = (G, \mathrm{lab}_s)$ the labeled digraph, called *update digraph*, where the function $\mathrm{lab}_s : A \to \{\ominus, \oplus\}$ is defined as:

$$\mathrm{lab}_s(u,v) = \begin{cases} \oplus & \text{if } s(u) \geq s(v) \\ \ominus & \text{if } s(u) < s(v) \end{cases}$$

The update digraph associated to a Boolean network $N = (F, s)$ is defined by $G^F_s = (G^F, \mathrm{lab}_s)$ (see an example of update digraph G_s in Figure 2.1). Hence, we define the following equivalence relation between update schedules s and s':

$$s \sim_{G^F} s' \iff G^F_s = G^F_{s'}$$

We denote $[s]_{G^F}$ the equivalence class of s induced by $\sim_{G^F}$. Note that the label on a loop will always be $\oplus$. It was proven in Aracena et al. (2009) that if two different updates schedules have the same update digraph, then they also have the same dynamical behavior.

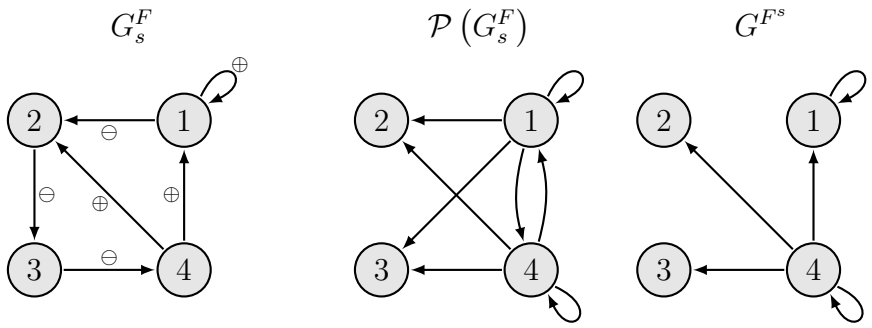
Given an update digraph $G_{\mathrm{lab}} = (G, \mathrm{lab})$, with $G = (V, A)$, we define the operator $\mathcal{P}$ as $\mathcal{P}(G_{\mathrm{lab}}) = (V', A')$, where $V' = V$ and $(u, v) \in A'$ if and only if:

I.- $(u, v) \in A$ and $\mathrm{lab}(u, v) = \oplus$ or,

II.- there exists $w \in V$ such that $(w, v) \in A$, $\mathrm{lab}(w, v) = \ominus$ and $(u, w) \in A'$.

Note that G^{F^s} is a sub-digraph of $\mathcal{P}(G^F_s)$ (See an example in Figure 2.2). $\mathcal{P}(G^F_s)$ is referred to as the *parallel digraph of* G_{lab}.

Given a digraph $G = (V, A)$, a function $\mathrm{lab}\colon A \to \{\oplus, \ominus, \bigcirc\}$ is called a *partial label* of G. In this case, the labeled digraph $G_{\mathrm{lab}} = (G, \mathrm{lab})$ is called a *partial labeled digraph*. The



$$
\begin{aligned}
f_1(x) &= x_1 \wedge x_4 \\
f_2(x) &= x_1 \vee x_4 \\
f_3(x) &= x_2 \\
f_4(x) &= x_3 \\
s &= \{1\}\,\{2\}\,\{3\}\,\{4\}
\end{aligned}
\qquad
\begin{aligned}
f_1^s(x) &= x_1 \wedge x_4 \\
f_2^s(x) &= x_4 \\
f_3^s(x) &= x_4 \\
f_4^s(x) &= x_4
\end{aligned}
$$

Figure 2.2: Example of G_s^F, $\mathcal{P}(G_s^F)$ and G^{F^s}

support of lab is defined by $\mathrm{Sup}\,(G_{\mathrm{lab}}) = \{a \in A\colon \mathrm{lab}\,(a) \neq \bigcirc\}$. If $\mathrm{Sup}\,(G_{\mathrm{lab}}) = A$, we say that G_{lab} is a *total labeled digraph*.

A partial labeled digraph G_{lab} is an *update digraph* if there exists an update schedule s such that

$$\forall a \in \mathrm{Sup}\,(G_{\mathrm{lab}})\,:\ \mathrm{lab}_s\,(a) = \mathrm{lab}\,(a) \tag{2.1}$$

Besides, an update schedule that satisfies (2.1) can be found in polynomial time (Aracena et al., 2011).

Given a partial or total labeled digraph $G_{\mathrm{lab}} = (G = (V, A), \mathrm{lab})$, we define $G_{\mathrm{lab}}^R = (G^R = (V, A^R), \mathrm{lab}^R)$, the *reverse digraph*, as follows:

- $(u, v) \in A^R \iff ((u, v) \in A \wedge \mathrm{lab}(u, v) = \oplus) \vee ((v, u) \in A \wedge \mathrm{lab}(v, u) = \ominus)$.
- $\mathrm{lab}^R(u, v) = \ominus$ if $\mathrm{lab}(v, u) = \ominus$ and $\mathrm{lab}^R(u, v) = \oplus$ otherwise.

Basically, all $\ominus$-arcs keep their labels and get their orientation inverted, all $\oplus$-arcs remain with its labels and orientation at least that the respective $\ominus$-arcs already exists, and all $\bigcirc$-arcs get removed (see an example in Figure 2.3).

A cycle in G_{lab}^R is called *forbidden* if it contains at least one $\ominus$-arc. It was shown in Aracena et al. (2011) that G_{lab} is an update digraph if and only if G_{lab}^R does not contain any forbidden cycles. Besides, this property can be tested in polynomial time.

Given a partial labeled update digraph $G_{\widetilde{\mathrm{lab}}}$, a label lab of G is said to be an *extension* of $\widetilde{\mathrm{lab}}$ if

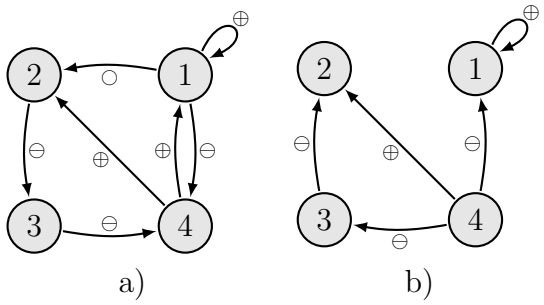


Figure 2.3: In a) a partial labeled digraph and in b) its reverse digraph.

1. $\mathrm{Sup}\left(G_{\widetilde{\mathrm{lab}}}\right) \subseteq \mathrm{Sup}\left(G_{\mathrm{lab}}\right)$.
2. $\forall a \in \mathrm{Sup}\left(G_{\widetilde{\mathrm{lab}}}\right) : \mathrm{lab}\left(a\right) = \widetilde{\mathrm{lab}}\left(a\right)$.
3. G_{lab} is an update digraph.

It was also shown in Aracena et al. (2011) that there always exist an extension of $\widetilde{\mathrm{lab}}$.

Given a finite set U of k elements, we say that a Boolean function $f : \{0,1\}^k \to \{0,1\}$ is *monotonic* on input $v \in U$ if for every $x \in \{0,1\}^k$ such that $x_v = 0$, we have that $f(x) \leq f(\bar{x}^v)$. A loop $(v,v) \in A\left(G^F\right)$ is *monotonic* if f_v is monotonic on input v. In particular, a monotonic function f is said to be an *AND function*, denoted $f\left(x\right) = \bigwedge\limits_{v \in U} x_v$, if and only if $f\left(x\right) = 1 \iff \forall v \in U \colon x_v = 1$. We say that a monotonic function f is an *OR function*, denoted $f\left(x\right) = \bigvee\limits_{v \in U} x_v$, if and only if $f\left(x\right) = 1 \iff \exists v \in U \colon x_v = 1$.

In this way, we say that a function $F \colon \{0,1\}^n \to \{0,1\}^n$ is *monotonic* if each local activation function is monotonic. We say that F is an *AND-OR function* if each local activation function is either an AND or an OR function. In this case, we define $V_{\mathrm{AND}}\left(F\right) \subseteq V\left(G^F\right)$ $\left(V_{\mathrm{OR}}\left(F\right) \subseteq V\left(G^F\right)\right)$ as the nodes that have an AND (OR) local activation function. In particular, we say that F is an *OR function* if each local activation function is an OR function.

An AND-OR function F can be completely described by its interaction digraph, labeling AND and OR nodes differently (in the figures of this paper, white nodes represent OR nodes, and dark gray nodes represent AND nodes). That is, given $G = (V, A)$ a digraph and $\{V_{\mathrm{AND}}, V_{\mathrm{OR}}\}$ a partition of V, we define $F \colon \{0,1\}^{|V|} \to \{0,1\}^{|V|}$ as follows:

$$\forall v \in V \colon f_v\left(x\right) = \begin{cases} \bigwedge\limits_{u \in N_G^-(v)} x_u & \text{if } v \in V_{\mathrm{AND}} \\ \bigvee\limits_{u \in N_G^-(v)} x_u & \text{if } v \in V_{\mathrm{OR}} \end{cases}$$

Note that if $N_G^-(v) = \emptyset$, then

$$f_v(x) = \begin{cases} 1 & \text{if } v \in V_{\text{AND}} \\ 0 & \text{if } v \in V_{\text{OR}} \end{cases}$$

Chapter 3

Robustness of limit cycles with update digraphs in Boolean networks

A Boolean network is said to be robust for a certain dynamical property, if small changes in the network do not affect some characteristic observed. There are several kinds of perturbations in a Boolean network: for instance, perturbations of the states of the nodes in a given global state of the network, changes in the local activation functions, or modifications of the type of update schedule, which is at the center of the present study. The last two ones correspond to changes in the definition of the network and therefore they can yield variations on the set of attractors.

The robustness of Boolean networks against perturbations of their the update schedule has been studied mainly from an experimental and statistical point of view (Elena, 2009; Chaves et al., 2005; Demongeot et al., 2008; Fauré et al., 2006). More recently, Christoph Schmal and Drossel (2010) have studied Boolean networks that follow a reliable trajectory in state space, which can be robust against perturbations in the update schedule. On the other hand, there exist only a few analytic studies on this subject (Salinas, 2008; Gómez, 2009; Noual, 2011). In particular, Goles and Salinas (2008) have done a comparative analysis on the attractors in Boolean networks with parallel and sequential update schedules.

Some analytic works about perturbations of update schedules have been made in a special class of discrete dynamical networks, called sequential dynamical systems, where the interaction digraph is symmetric or equivalently an undirected graph and the update schedule is sequential. For this class of networks, the team of Barrett, Mortveit and Reidys studied the set of sequential update schedules preserving the whole dynamical behavior of the network (Mortveit and Reidys, 2001) and the set of attractors in a certain class of cellular automata (Hansson et al., 2005).

This chapter[1] deals with the robustness of attractors of Boolean networks against changes in the deterministic update schedule, which may range from the parallel update, the most

[1]This chapter was published in Aracena et al. (2013b).

common (Kauffman, 1969), to the sequential update, passing through all the combinations of block-sequential updates. Some of the pioneering works in this context was done by Robert (1986) and Goles (1980).

In Salinas (2008) equivalence classes of deterministic update schedules were defined according to the labeled digraph associated to a given Boolean network (update digraph). It was proved that two schedules in the same class yield the same dynamical behavior of a given Boolean network (Salinas, 2008; Aracena et al., 2009). Besides, it was exhibited that the limit cycles of a Boolean network are very sensitive to changes of the update schedule. In this way, the existence of frozen nodes in a limit cycle could make it more robust. The importance of the frozen nodes of the attractors in the robustness of Boolean networks has been previously studied by Greil et al. (2007); Kauffman (1990).

Here, we study the update schedules preserving a set of given limit cycles of a Boolean network. Because the schedules in the same equivalence class preserve the whole dynamics of a Boolean network, we focus on the problem of determining the distinct equivalence classes whose elements preserve the limit cycles, not necessarily the whole dynamics, of a given Boolean network.

3.1. Motivation

The following result was proven in Aracena et al. (2009).

Theorem 3.1. *Let $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$ be two Boolean networks that differ only in the update schedule. If $[s_1]_{G^F} = [s_2]_{G^F}$, then $F^{s_1} = F^{s_2}$.*

In this way, the equivalence classes of update schedules $[s]_{G^F}$ previously defined are such that elements in a same class yield the same dynamical behavior in Boolean networks which differ only in the update schedule.

Hence, the interesting problem is knowing for a given Boolean network if there exists another non equivalent update schedule such that preserves the limit cycles of the network. Next, we show that the related problem is NP-hard.

Limit Cycle Problem **(LCP):** Given a Boolean network $N = (F, s)$ and $\mathcal{C} \in LC(N)$. Does there exists $\hat{s} \notin [s]_{G^F}$ such that $\mathcal{C} \in LC(F, \hat{s})$?

Theorem 3.2. *LCP is NP-hard.*

Proof. We prove that $\text{SAT}_0 \leq_p \text{LCP}$, where SAT_0 is defined as

SAT_0: Given a normal conjunctive formula (ncf) ϕ with $\phi(\vec{0}) = 1$. Does there exist $x \neq \vec{0}$ such that $\phi(x) = 1$?, where $\vec{0} = (0, 0, \ldots, 0)$.

Note that SAT_0 is obviously NP-complete.

Let ϕ be a ncf in variables $w_1, \ldots, w_n$ such that $\phi(\vec{0}) = 1$, with $\vec{0} \in \{0,1\}^n$. We define V and $F = (f_v)_{v \in V} : \{0,1\}^{n+3} \to \{0,1\}^{n+3}$ as follows:

$$\begin{aligned} \forall i \in \{1,\ldots,n\},\ f_{v_i}(x) &= \neg x_{v_i} \wedge x_{z_2}, \\ f_{z_1}(x) &= \neg x_{z_1} \wedge x_{z_2}, \\ f_{z_2}(x) &= \neg x_{z_1} \wedge \bigwedge_{i=1}^{n} \neg x_{v_i}, \\ f_{v_\phi}(x) &= \phi\left(x_{v_i} : i \in \{1,\ldots,n\}\right) \wedge \neg x_{z_1}. \end{aligned}$$

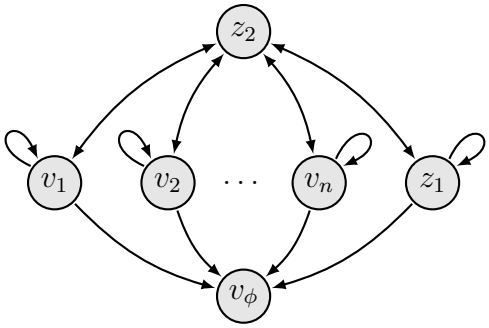


Figure 3.1: Interaction digraph of the transformation defined in Theorem 3.2.

Now, we consider the Boolean network $N = (F, s)$ where $s = \{z_2, v_\phi\}\{v_1, \ldots, v_n, z_1\}$. Clearly $\mathcal{C} = \left[\vec{0}, \vec{1}, \vec{0}\right] \in LC(N)$, where $\vec{1} = (1, 1, \ldots, 1)$, $\vec{0} \in \{0,1\}^{n+3}$. Moreover, from the dynamical behavior of N described in Table 3.1 we have that $LC(N) = \{\mathcal{C}\}$. Hence, if there is another update schedule that share a limit cycle with N, it must be necessarily $\mathcal{C}$. Now, we prove the equivalence,

x				$F^s(x)$			
$x_{v_1}, \ldots, x_{v_n}$	$x_{v_{n+1}}$	$x_{v_{n+2}}$	$x_{v_{n+3}}$	$f^s_{v_1}(x), \ldots, f^s_{v_n}(x)$	$f^s_{v_{n+1}}(x)$	$f^s_{v_{n+2}}(x)$	$f^s_{v_{n+3}}(x)$
$= \vec{0}$	0	$*$	$*$	$\vec{1}$	1	1	1
$= \vec{0}$	1	$*$	$*$	$\vec{0}$	0	0	0
$\neq \vec{0}$	0	$*$	$*$	$\vec{0}$	0	0	$*$
$\neq \vec{0}$	1	$*$	$*$	$\vec{0}$	0	0	0

Table 3.1: Dynamical behavior of the transformation defined in Theorem 3.2 ($*$ means either 0 or 1).

($\Longrightarrow$) Let $\hat{w} \neq \vec{0}$ be such that $\phi(\hat{w}) = 1$. Then, considering the update schedule $\hat{s} = \{z_2\}\{v_i : \hat{w}_i = 1\}\{v_\phi\}\{v_i : \hat{w}_i = 0\}\{z_1\}$, we have that $\mathcal{C} \in LC(F, \hat{s})$ and $\hat{s} \notin [s]_{G^F}$.

($\Longleftarrow$) Let be $\hat{s} \notin [s]_{G^F}$ an update schedule such that $\mathcal{C} \in LC(F, \hat{s})$. We note that:

$$\mathrm{lab}_{\hat{s}}(z_1, v_\phi) = \ominus \implies f^{\hat{s}}_{v_\phi}(\vec{0}) = 0.$$

Therefore, $\mathrm{lab}_{\hat{s}}(z_1, v_\phi) = \oplus$. Since $f^{\hat{s}}_{v_\phi}(\vec{0}) = 1$, necessarily $\phi(x^{\hat{s}}) = 1$, where:

$$\forall i \in \{1, \ldots, n\},\ x^{\hat{s}}_{v_i} = \begin{cases} 0 & \text{if } \mathrm{lab}_s(v_i, v_\phi) = \oplus \\ 1 & \text{if } \mathrm{lab}_s(v_i, v_\phi) = \ominus. \end{cases}$$

Now, since $\hat{s} \notin [s]_{G^F}$, we have that there exists $i \in \{1, \ldots, n\}$ such that $\mathrm{lab}_s(v_i, v_\phi) = \ominus$, meaning that $x^{\hat{s}} \neq \vec{0}$. Indeed, since $f_v(\vec{0}) = 1$ for every $v \in \{v_1, \ldots, v_n, z_1\}$, then $\mathrm{lab}_{\hat{s}}(v, z_2) = \mathrm{lab}_s(v, z_2) = \oplus$ and $\mathrm{lab}_{\hat{s}}(z_2, v) = \mathrm{lab}_s(z_2, v) = \ominus$. Therefore, there exists $i \in \{1, \ldots, n\}$ such that $\mathrm{lab}_s(v_i, v_\phi) = \ominus$.

□

It is easy to check that $\mathcal{C}$ is the only limit cycle of $\hat{N} = (F, \hat{s})$ in the above proof. Hence, we obtain the following Corollary.

Corollary 3.3. *The following problems are NP-hard.*

Limit Cycle Set Problem: *Given a Boolean network $N = (F, s)$. Does there exists $\hat{s} \notin [s]_{G^F}$ such that $LC(N) = LC(F, \hat{s})$?*

Common Limit Cycle Problem: *Given a Boolean network $N = (F, s)$. Does there exists $\hat{s} \notin [s]_{G^F}$ such that $LC(N) \cap LC(F, \hat{s}) \neq \emptyset$?*

On another hand, the following theorem shows that for any given Boolean network, the possibility that another non equivalent update schedule yields a same limit cycle depends on the global activation function F and not only on the associated update digraph. Therefore, it is not possible to define a new equivalence relation between update schedules, by relaxing the condition of equal update digraphs, and such that elements in the same class preserve the set of limit cycles and not necessarily the whole dynamics of the network.

Theorem 3.4. *Let $G = (V, A)$ be a digraph and let s_1, s_2 be two different update schedules such that $G_{s_1} \neq G_{s_2}$. Then, there exists a function $F : \{0,1\}^n \to \{0,1\}^n$, with $G^F = G$, such that $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$ verify $LC(N_1) \neq LC(N_2)$.*

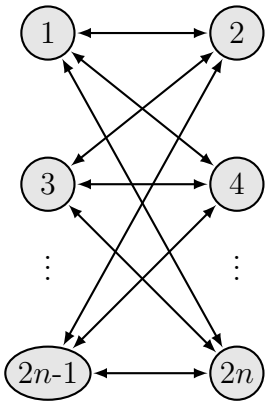


Figure 3.2: Complete bipartite digraph $K_{n,n}$ used in Example 3.1.

Proof. We are going to define F such that $\mathcal{C} = [\vec{0}, \vec{1}, \vec{0}]$ is a limit cycle of N_1 but not of N_2, where $\vec{0},\ \vec{1} \in \{0,1\}^n$.

For each $v \in V$, we define

$$f_v(x) = \bigwedge_{u \in N_G^-(v)} x_u^{\sigma_u}.$$

where $x_u^{\sigma_u} = x_u$ if $\mathrm{lab}_{s_1}(u,v) = \ominus$ and $x_u^{\sigma_u} = \neg x_k$ if $\mathrm{lab}_{s_1}(u,v) = \oplus$.

Hence, on one hand, by induction on the nodes in increasing order according to the value of s_1, we obtain that $\forall\, v \in V$, $f_v(\vec{0}) = 1 \wedge f_v(\vec{1}) = 0$. Thus, $F^{s_1}(\vec{0}) = \vec{1} \wedge F^{s_1}(\vec{1}) = \vec{0}$. Therefore, $\mathcal{C} = [\vec{0}, \vec{1}, \vec{0}]$ is a limit cycle of N_1.

On the other hand, let $v \in V$ such that $\exists\, u \in N_G^-(v)$, $\mathrm{lab}_{s_1}(u,v) \neq \mathrm{lab}_{s_2}(u,v)$. Then, $f_v^{s_2}(\vec{0}) = 0 \neq f_v^{s_1}(\vec{0})$. Therefore, $\mathcal{C}$ is not a limit cycle of N_2. □

As a direct consequence of Theorem 3.4, the existence of shared limit cycles in networks, which differ only in the update schedule, could depend strongly on the global activation function, as shown in Example 3.1.

Example 3.1. Let $N = (F, s)$ be a Boolean network such that $G^F = K_{n,n}$, where $K_{n,n}$ is the complete bipartite digraph as shown in Figure 3.2.

For a given $k \in \{1, \ldots, n\}$, we define $F = (f_1, \ldots, f_{2n})$ as follows:

$$f_i(x) = \begin{cases} 1 & \text{if} \quad \displaystyle\sum_{i \in N_{G^F}^-(i)} x_i \geq k, \\ 0 & \text{if} \quad \displaystyle\sum_{i \in N_{G^F}^-(i)} x_i < k. \end{cases}$$

Hence, if s is the parallel update schedule, then $\mathcal{C} = [x^0, x^1, x^0]$ is a limit cycle of N, where $x_i^0 = 0$ for i even, $x_i^0 = 1$ for i odd and for every i, $x_i^1 = \neg x_i^0$.

Next, we exhibit that the existence of another non equivalent update schedule that preserves the limit cycle $\mathcal{C}$ depends on the value of k.

- For $k = 1$ and $k = n$ we have that for every i, $f_i(x) = \bigvee_{j \in N^-_{G^F}(i)} x_j$ and $f_i(x) = \bigwedge_{j \in N^-_{G^F}(i)} x_j$, respectively. It is easy to check that in either case, there is not any other class of update schedules that yields the limit cycle $\mathcal{C}$.

- For $k = \lceil \frac{n}{2} \rceil$, we have that for every i, f_i is the majority function.

  In the parallel schedule, we have that each node in state one receives exactly n ones and each node in state zero also receives exactly n zeros. Thus, to keep the state of a node in $\mathcal{C}$, we need that each node receives at least $\lceil \frac{n}{2} \rceil$ ones or zeros, respectively. That means that we have to change at most $\lfloor \frac{n}{2} \rfloor$ ones or zeros to zeros or ones, respectively. Hence, if

$$k_n = \begin{cases} \lfloor \frac{n}{2} \rfloor & \text{if } n \text{ is odd,} \\ \lfloor \frac{n-1}{2} \rfloor & \text{if } n \text{ is even.} \end{cases}$$

  then, we need both of the following conditions:

$$|\{i \in I \colon s(i) = 1\}| \leq k_n \quad \text{and} \quad |\{i \in P \colon s(i) = 1\}| \leq k_n,$$

  where $I = \{1, 3, \ldots, 2n-1\}$ and $P = \{2, 4, \ldots, 2n\}$. Therefore,

$$|\{[s]_{G^F} \colon \mathcal{C} \in LC(F, s)\}| \geq 1 + \left(\sum_{i=1}^{k_n} \binom{n}{i} \right)^2.$$

3.2. Necessary conditions to share limit cycles

Now we are interested in studying what kind of information is provided by the update digraph about the possibility of two Boolean networks, that differ only in their update schedule, to share limit cycles.

From the previous section we know that there always exists a global function F such that for any two non-equivalent update schedules s_1 and s_2, $LC(F, s_1) \neq LC(F, s_2)$. Goles and Salinas (2008) proved that in the case where the interaction digraph does not have loops, the dynamical behavior of the network updated in parallel way does not share limit cycles with any sequential update schedule. Also, Aracena et al. (2009) proved that for any update schedule there exists a sequential update schedule what does not share limit cycles. Now, given $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$ two Boolean networks such that $G^F_{s_1} \neq G^F_{s_2}$ and the loops are monotonic, we propose a polynomial test that give us a necessary condition on the update digraphs for N_1 and N_2 to share a limit cycle.

3.2.1. Necessary conditions algorithm

In this section we present Test 1 such that with input a given digraph $G = (V, A)$ and s_1 and s_2 two update schedules on V, returns TRUE if any pair of Boolean networks $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$ such that $G^F = G$ satisfies $LC(N_1) \cap LC(N_2) = \emptyset$.

Test 1

Input: $G = (V, A)$, s_1, s_2
Output: true or false

$M = \emptyset$
$N = V$
while $\exists v \in N$ *such that* $(((N_G^-(v) \setminus \{v\}) \cap N = \emptyset) \vee (\exists u \in M, N_G^-(u) = \{v\})) \vee$
$(\forall u \in (N_G^-(v) \setminus \{v\}) \cap N, ((\mathrm{lab}_{s_1}(u, v) = \oplus \wedge \mathrm{lab}_{s_2}(u, v) = \ominus)) \vee$
$(\forall u \in (N_G^-(v) \setminus \{v\}) \cap N, ((\mathrm{lab}_{s_1}(u, v) = \ominus \wedge \mathrm{lab}_{s_2}(u, v) = \oplus))$ **do**
    $M \leftarrow M \cup \{v\}$
    $N \leftarrow N \setminus \{v\}$
end
if $M = V$ **then**
    return `true`
else
    return `false`
end

This algorithm marks the nodes that should be frozen in a possible shared limit cycle of two Boolean networks which differ only in their update schedule. If every node is marked, i.e. belongs to M, then the only shared attractors are fixed points.

Definition 3.1. Let $N = (F, s)$ and $N' = (F, s')$ be two Boolean networks with different update schedules. We say that a node $v \in V(G^F)$, without a loop or with a monotonic loop, has the *homogeneous labels property* if $\forall\, u \in N_{G^F}^-(v)$, $\mathrm{lab}_s(u, v) = \oplus$ and $\forall\, u \in N_{G^F}^-(v) \setminus \{v\}$, $\mathrm{lab}_{s'}(u, v) = \ominus$.

The following result was proved in Aracena et al. (2009).

Proposition 3.5. *Let $N = (F, s)$ and $N' = (F, s')$ be two Boolean networks with different update schedules and $\mathcal{C} \in LC(N) \cap LC(N')$. If $v \in V(G^F)$ has the homogeneous labels property, then v is a frozen node in $\mathcal{C}$.*

Theorem 3.6. *Let $G = (V, A)$ be a digraph and let s_1 and s_2 be two update schedules on V. If* Test 1*(G, s_1, s_2)=TRUE, then every function F such that $G^F = G$ and the loops are monotonic, satisfies $LC(F, s_1) \cap LC(F, s_2) = \emptyset$. Besides,* Test 1 *runs on time $O(|V|^2)$.*

Proof. We distinguish the following causes for a node to be frozen in a shared limit cycle:

1. A node without loop depending either only on frozen nodes or without inputs.

2. A node with a monotonic loop, such that it does not depend on a node (different from itself) that is not frozen.

3. A node that is the only input of a frozen node.

These ones are checked in line 3 of `Test 1`, consecutively. Hence, if $N = V$, then all nodes are frozen in the limit cycle, which is a contradiction. Besides, all conditions are feasible for being checked in $O(|V|)$ elemental operations. Hence, run time of `Test 1` is $O(|V|^2)$. □

Example 3.2. Let us consider G_{s_1} and G_{s_2} as defined in Figure 3.3a) . Then, nodes 2 and 5 are marked because of the homogeneous label property (Figure 3.3b)). Next, node 3 is marked because it is the only input to a marked node and node and node 4 is marked because all its incoming neighbors are marked (Figure 3.3c)). Finally, node 1 is marked because all its incoming neighbors are marked (Figure 3.3d)).

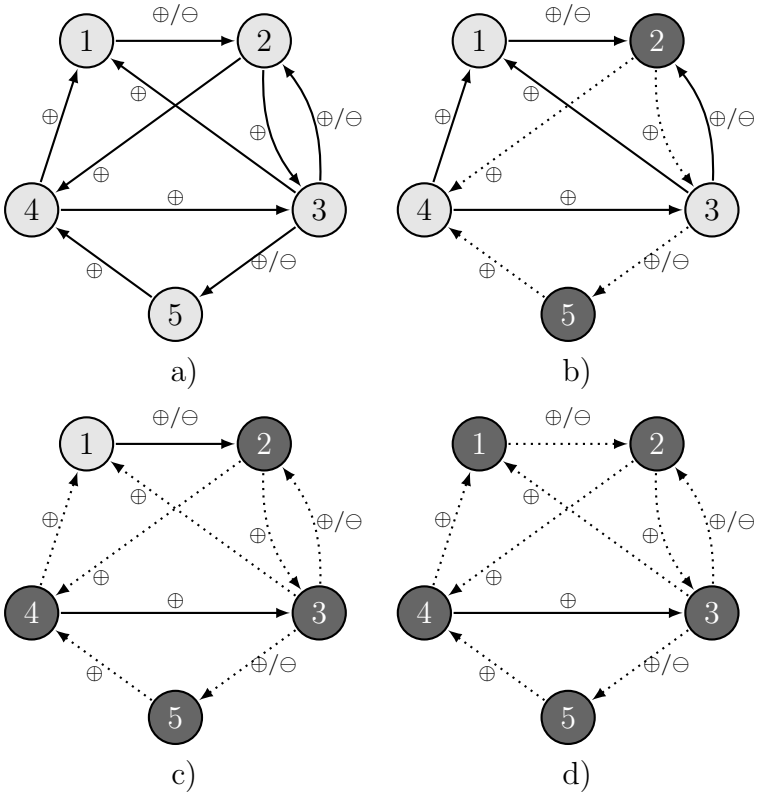


Figure 3.3: Example of how `Test 1` works, as described in Example 3.2.

Thus, we have the following relation between the update digraph structure and the preservation of limit cycles in Boolean networks.

Corollary 3.7. *Let $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$ be two Boolean networks such that $\Delta^-(G^F) = 1$, G^F is connected and $[s_1]_{G^F} \neq [s_2]_{G^F}$. Then, $LC(N_1) \cap LC(N_2) = \emptyset$.*

Proof. We apply `Test 1` to G^F, s_1 and s_2. Since $[s_1]_{G^F} \neq [s_2]_{G^F}$, $\exists v \in V(G^F)$, $u \in N^-_{G^F}(v)$: $\text{lab}_{s_1}(u,v) \neq \text{lab}_{s_2}(u,v)$.

- Given that $\Delta^-(G^F) = 1$, we have that v satisfies the homogeneous labels property, and therefore, is marked.
- Sequentially and forward, all nodes reachable from v (i.e., there exists a path from v to the nodes) are marked, since each one of them has only one input, which is marked.
- Sequentially and backward, all nodes reaching v (i.e., there exists a path from the nodes to v) are marked, since they correspond to the only input of a marked node.
- Since G^F is connected, all nodes are marked. Thus, `Test 1`(G^F, s_1, s_2)=TRUE.

Therefore, $LC(N_1) \cap LC(N_2) = \emptyset$. □

Definition 3.2. Given a digraph $G = (V, A)$, we will say that $G' = (V', A')$ is a source sub-digraph of G if $V' \subseteq V$, $A' = A \cap V' \times V'$ and $\forall v \in V'$, $(u, v) \in A \Rightarrow u \in V'$.

Corollary 3.7 can be simply extended to networks having a source sub-digraph with the properties stated in it as is established in the following result.

Corollary 3.8. *Let $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$ be two Boolean networks and $\mathcal{C} \in LC(N_1) \cap LC(N_2)$. If G' is a source sub-digraph of G^F that satisfies the properties stated in Corollary 3.7, then every node $v \in V(G')$ is frozen in $\mathcal{C}$.*

Example 3.3 shows an application of Corollary 3.8.

Example 3.3. Let us consider $F = (f_1, f_2, f_3, f_4, f_5)$ and $\mathcal{C}$ as defined in Figure 3.4 and let us define $N_1 = (F, s_1)$, $N_2 = (F, s_2)$, where $s_1 = \{1, 2, 3, 4, 5\}$ and $s_2 = \{1, 2, 4, 5\}\,\{3\}$.

The sub-digraph $G' = (V' = \{1, 3\}, A' = \{(1, 3), (3, 1)\})$ of G^F satisfies the conditions of Corollary 3.8 and $LC(N_1) \cap LC(N_2) = \{\mathcal{C}\}$. Hence, we can see that nodes 1 and 3 are indeed frozen in $\mathcal{C}$.

Remark 3.1. Observe that for $N_i = (F, s_i)$, $i = 1, 2$ two given Boolean networks such that $\Delta^-(G^F) = 1$ and G^F is connected,

$$LC(N_1) = LC(N_2) \quad \veebar \quad LC(N_1) \cap LC(N_2) = \emptyset.$$

Indeed, from Theorem 3.1, if $G^F_{s_1} = G^F_{s_2}$, then $F^{s_1} = F^{s_2}$. This implies $LC(N_1) = LC(N_2)$. Otherwise, by Corollary 3.7 $LC(N_1) \cap LC(N_2) = \emptyset$.

This tells us that for a Boolean network $N = (F, s)$, with G^F connected and $\Delta^-(G^F) = 1$, and whose limit cycle set is not empty, the unique equivalence class of update schedules yielding this set is $[s]_{G^F}$. This is not true if any condition on G^F does not hold (see Example 3.4). Indeed, it is sufficient that there exists a node $v \in V(G^F)$ with $|N^-_{G^F}(v)| \geq 2$ for there to be different limit cycle sets in Boolean networks which differ only in the update schedule.

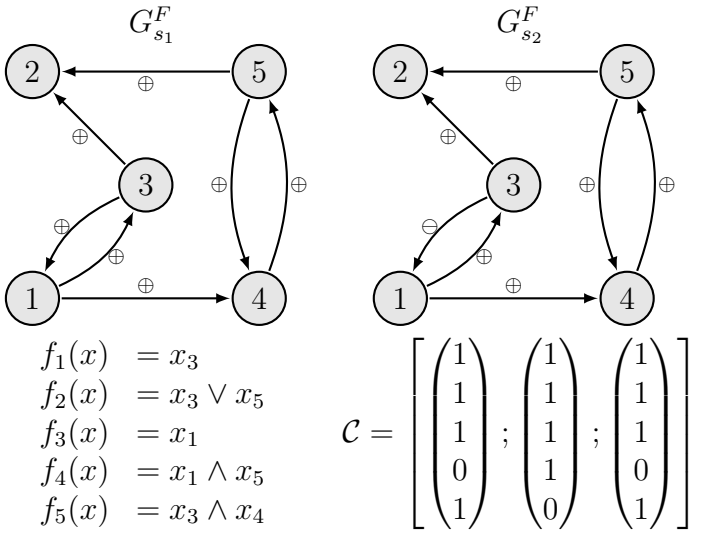


Figure 3.4: Example of two Boolean networks satisfying the conditions of Corollary 3.8, as detailed in Example 3.3.

Example 3.4. Let us consider $F = (f_1, f_2, f_3, f_4, f_5)$ as defined in Figure 3.5 and let us define $N_1 = (F, s_1)$ and $N_2 = (F, s_2)$, where $s_1 = \{1, 2, 3, 4, 5\}$ and $s_2 = \{1, 2\}\,\{5\}\,\{3, 4\}$. We note that G^F is connected, $\Delta^-(G^F) = 2$ and $G^F_{s_1} \neq G^F_{s_2}$.

Each network has six limit cycles, but only three of them are common to both networks. More precisely, $LC(N_1) \cap LC(N_2) = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, where $\mathcal{C}_1 = [x^1, x^2, x^1]$, $\mathcal{C}_2 = [x^3, x^4, x^3]$ and $\mathcal{C}_3 = [x^5, x^6, x^5]$ with:

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^5 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^6 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Therefore, as shown in the previous example, for Boolean networks $N = (F, s)$, where $\Delta^-(G^F) \geq 2$ and $LC(N) \neq \emptyset$, it is not possible to guarantee the existence of another equivalence class $[s']_{G^F}$ different from $[s]_{G^F}$ such that $LC(F, s') = LC(N)$, only knowing the update digraph G^F_s as established in Theorem 3.4. It is necessary to have some additional knowledge about the local activation functions of the network.

3.3. Possibility of sharing limit cycles

In the previous section we proved that given G, s_1 and s_2 such that $G_{s_1} \neq G_{s_2}$ if `Test 1`(G, s_1, s_2)=TRUE, then it is not possible to have a same limit cycle in Boolean net-

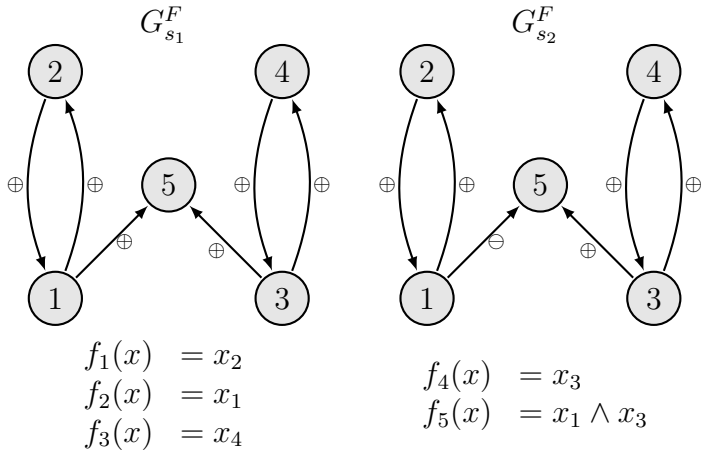


Figure 3.5: Example of two Boolean networks that do not satisfy the conditions of Remark 3.1, as detailed in Example 3.4.

works with interaction digraph G and update schedules s_1 and s_2.

In this section, we study the possibility of sharing limit cycles when `Test 1`(G, s_1, s_2)=FALSE.

Definition 3.3. Given G_{s_1} and G_{s_2} two update digraphs, we denote for every $v \in V$:

$$N_e^-(v) = \{u \in N_G^-(v) : \text{lab}_{s_1}(u,v) = \text{lab}_{s_2}(u,v)\},$$
$$N_d^-(v) = \{u \in N_G^-(v) : \text{lab}_{s_1}(u,v) \neq \text{lab}_{s_2}(u,v)\}.$$

Theorem 3.9. *Let G, s_1 and s_2 be such that $G_{s_1} \neq G_{s_2}$ and* `Test 1`(G, s_1, s_2)*=FALSE and N and M the resulting sets. If the following conditions*

1. $\forall v \in N,\ N_e^-(v) \cap N \neq \emptyset$,
2. $\forall v \in N,\ |N_G^-(v)| = 2 \implies N_G^-(v) \subseteq N_e^-(v)$,
3. $\forall v \in M,\ |N_G^-(v)| = 2 \implies N_G^-(v) \subseteq N_e^-(v) \ \vee \ N_G^-(v) \subseteq N_d^-(v)$,

are satisfied, then $\exists F,\ G^F = G \ \wedge \ LC(F, s_1) \cap LC(F, s_2) \neq \emptyset$.

Proof. Since `Test 1`(G, s_1, s_2)=FALSE, then $N \neq \emptyset$. We will define each local activation function f_v such that $G^F = G$ and $[x^0, x^1, x^0] \in LC(F, s_1) \cap LC(F, s_2)$, where $\forall v \in V \setminus M$, $x_v^0 = 0$, $x_v^1 = 1$ and $\forall v \in M$, $x_v^0 = x_v^1 = 0$. For given $v \in V$, we denote $x_u^{\sigma_u} = x_u$ if $\text{lab}_{s_1}(u,v) = \ominus$ and $x_u^{\sigma_u} = \neg x_u$ if $\text{lab}_{s_1}(u,v) = \oplus$. Next, we define the activation local functions f_c depending on each case:

Case 1: $v \in M, N_G^-(v) \cap M \neq \emptyset$, then

$$f_v(x) = \bigwedge_{u \in N_G^-(v)} x_u.$$

Case 2: $v \in M, N_G^-(v) \subseteq N$. Since $|N_G^-(v)| \neq 1$ and by the hypothesis $N_G^-(v) \subseteq N_e^-(v)$ or $N_G^-(v) \subseteq N_d^-(v)$, then $\exists u_1 \neq u_2, \{u_1, u_2\} \subseteq N_e^-(v) \ \vee \ \{u_1, u_2\} \subseteq N_d^-(v)$. Hence,

$$f_v(x) = (x_{u_1}^{\sigma_{u_1}} \veebar x_{u_2}^{\sigma_{u_2}}) \wedge \bigwedge_{u \in N_G^-(v) \setminus \{x_{u_1}, x_{u_2}\}} x_u.$$

Case 3: $v \in N$. If $N_G^-(v) \cap N \subseteq N_e^-(v) \ \vee \ N_G^-(v) \cap M \neq \emptyset$, then

$$f_v(x) = \bigwedge_{u \in N_e^-(v) \cap N} x_u^{\sigma_u} \ \vee \ \bigwedge_{u \in N_G^-(v) \setminus (N_e^-(v) \cap N)} x_u.$$

Case 4: $v \in N$, $N_G^-(v) \cap M = \emptyset$, $\exists u_1, u_2 \in N_e^-(v) \cap N, u_1 \neq u_2$ and $N_G^-(v) \cap N_d^-(v) \neq \emptyset$, then

$$f_v(x) = x_{u_1}^{\sigma_{u_1}} \vee \left[(x_{u_1}^{\sigma_{u_1}} \veebar x_{u_2}^{\sigma_{u_2}}) \wedge \bigwedge_{u \in N_G^-(v) \setminus \{u_1, u_2\}} x_u \right].$$

Case 5: Otherwise, i.e. $v \in N$ and $\exists u_1 \in N_e^-(v) \cap N$ and $\exists u_2, u_3 \in N_d^-(v) \cap N, u_2 \neq u_3$

$$f_v(x) = x_{u_1}^{\sigma_{u_1}} \vee \left[(x_{u_2}^{\sigma_{u_2}} \veebar x_{u_3}^{\sigma_{u_3}}) \wedge \bigwedge_{u \in N_G^-(v) \setminus \{u_1, u_2, u_3\}} x_u \right].$$

Hence, it is easy to check that the local functions defined in this way satisfy the conditions: $G^F = G \ \wedge \ LC(F, s_1) \cap LC(F, s_2) \neq \emptyset$. □

If there exists at least one node which does not satisfy the conditions of Theorem 3.9, then the proposed limit cycle is not a shared limit cycle for the networks. In this case we propose `Test 2` that receives as input the digraph G, update schedules s_1, s_2 and sets M and N resulting from `Test 1`. If this algorithm returns TRUE then it is easy to see that there exists a function F such that (F, s_1) and (F, s_2) share a limit cycle, i.e. $LC(F, s_1) \cap LC(F, s_2) \neq \emptyset$. The construction of function F is the same that we have seen before. In fact, this algorithm propose a way to eliminate the critical cases marking a new node as frozen.

Test 2

Input: $G = (V, A),\ s_1, s_2,\ N,\ M$
Output: true or false

while
$(\exists v \in N)(\forall u \in N_G^-(v) \cap N), \mathrm{lab}_1(u, v) \neq \mathrm{lab}_2(u, v)\ \vee\ [(\exists v \in M), N_G^-(v) = \{u_1, u_2\},$
$u_1 \in N_e^-(v) \wedge u_2 \in N_d^-(v)]$ **do**
    if $(\forall u \in N_G^-(v) \cap N), \mathrm{lab}_1(u, v) \neq \mathrm{lab}_2(u, v)$ **then**
        $x \leftarrow v$
    else
        $x \leftarrow u_2$
    end
    $M \leftarrow M \cup \{x\}$
    $N \leftarrow N \setminus \{x\}$
    while $\exists v \in N, ((N_G^-(v) \cap N = \emptyset) \vee (\exists u \in M, N_G^-(u) \cap N = \{v\}))$ **do**
        $M \leftarrow M \cup \{v\}$
        $N \leftarrow N \setminus \{v\}$
    end
end
if $N \neq \emptyset$ **then**
    return `true`
else
    return `false`
end

The results proved so far has been summarized in Table 3.2 that shows how the space of Boolean networks gets divided.

<table>
<tr><td rowspan="4">$G_{s_1} = G_{s_2}$

$\forall F\colon F^{s_1} = F^{s_2}$

$\forall F\colon LC(F, s_1) = LC(F, s_2)$</td><td colspan="4">$G_{s_1} \neq G_{s_2}$
$\exists F\colon LC(F, s_1) \neq LC(F, s_2)$</td></tr>
<tr><td rowspan="3">Test 1=TRUE

$\forall F\colon LC(F, s_1) \cap LC(F, s_2) = \emptyset$</td><td colspan="3">Test 1=FALSE</td></tr>
<tr><td colspan="2">Certain Conditions = TRUE</td><td>Certain Conditions = FALSE</td></tr>
<tr><td rowspan="2">$\exists F\colon LC(F, s_1) \cap LC(F, s_2) \neq \emptyset$</td><td>Test 2=TRUE</td><td>Test 2=FALSE</td></tr>
<tr><td>$\exists F\colon LC(F, s_1) \cap LC(F, s_2) \neq \emptyset$</td><td>?</td></tr>
</table>

Table 3.2: Summary of the results of the necessary conditions of Chapter 3.

3.4. Construction of classes preserving limit cycles

Theorem 3.2 and Example 3.1 show that determining the non-equivalent update schedules which preserve limit cycles of a given Boolean network is in general not an easy task, and that it depends strongly on the global activation function F of the network.

As shown in Corollary 3.8, the existence of frozen nodes is a necessary condition for Boolean networks whose update schedules differ to share limit cycles, under certain hypotheses on their architecture. However, the existence of frozen nodes is not by itself a sufficient condition as shows the following example.

Example 3.5. Let us consider the Boolean network $N = (F, s)$, where

$$\forall i \in \{1, \ldots, 9\},\ f_i(x) = \bigvee_{j \in N^-_{G^F}(i)} x_j,$$

with $N^-_{G^F}(i)$ as shown in Figure 3.6, and

$$f_{10} = (\neg x_3 \wedge x_4 \wedge \neg x_9) \vee (x_3 \wedge \neg x_4 \wedge x_9),$$

$s = \{1, 2, 3, 7, 8, 9\}\{4, 5, 6, 10\}$ and $\mathcal{C} = [x^0, x^1, x^2, x^0] \in LC(N)$, with $x^0 = (1, 0, 0, 1, 1, 0, 1, 0, 0, 1)$, $x^1 = (0, 1, 0, 0, 1, 1, 0, 1, 0, 1)$, $x^2 = (0, 0, 1, 1, 0, 1, 0, 0, 1, 1)$. Note that node 10 is frozen in $\mathcal{C}$.

It is easy to check that there is not another non-equivalent update schedule s' such that $\mathcal{C}$ is also a limit cycle of $N' = (F, s')$.

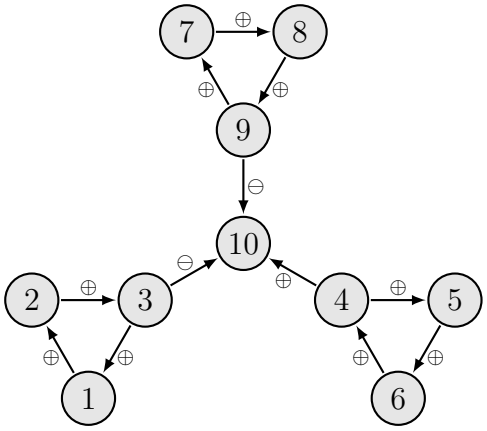


Figure 3.6: Update digraph associated to N defined in Example 3.5.

Nevertheless, the existence of frozen nodes can be, in some cases, a sufficient condition for two Boolean networks $N = (F, s)$ and $N' = (F, s')$, such that s and s' do not belong to the same update schedules equivalence class, to share a given limit cycle as shown in the following result.

Theorem 3.10. *Let be $N = (F, s)$ a Boolean network, $\mathcal{C} = \left[x^k\right]_{k=0}^{p}$, $p > 1$ a limit cycle of N, Z the set of frozen nodes in $\mathcal{C}$ and $v \in Z$, satisfying either of the following conditions:*

1.- All labels incoming to v are of the same type.

2.- $N^-_{G^F}(v) \subseteq Z$.

Then there exists an update schedule $\hat{s}$ with $[\hat{s}]_{G^F} \neq [s]_{G^F}$ and such that $\mathcal{C} \in LC(F, \hat{s})$.

Proof. Let $x^k_v = a$, $a \in \{0, 1\}$, $\forall k \in \{0, \ldots, p-1\}$.

1.- Case i. $\forall u \in N^-_{G^F}(v), \mathrm{lab}_s(u, v) = \ominus$.

Let $u^* \in N^-_{G^F}(v)$: $s(u^*) = \min\limits_{u \in N^-_{G^F}(v)} s(u) = m \geq 1$. Given $k \in \{1, \ldots, m\}$, we define:

$$\begin{array}{ll} & \hat{s}(v) = k, \\ \forall u \neq v, s(u) < k, & \hat{s}(u) = s(u), \\ \forall u \neq v, k \leq s(u) \leq s(v), & \hat{s}(u) = s(v) + 1, \\ \forall u \neq v, s(u) > s(v), & \hat{s}(u) = \begin{cases} s(u) + 1 & \text{if } \exists w \neq v \colon s(w) = s(v) \\ s(u) & \text{if } \nexists w \neq v \colon s(w) = s(v). \end{cases} \end{array}$$

Hence, $\forall u \in N^-_{G^F}(v), \mathrm{lab}_{\hat{s}}(u, v) = \oplus$ and $\forall (w, y) \in A(G^F)$, $w, y \neq v$, $\mathrm{lab}_{\hat{s}}(w, y) = \mathrm{lab}_s(w, y)$. Therefore, $[\hat{s}]_{G^F} \neq [s]_{G^F}$.

Thus,

$$f^{\hat{s}}_v(x^0_u : u \in N^-_{G^F}(v)) = f_v(x^0_u : u \in N^-_{G^F}(v)) = f^s_v(x^1_u : u \in N^-_{G^F}(v)) = x^1_v.$$

By induction, we have $f^{\hat{s}}_v(x^k_u : u \in N^-_{G^F}(v)) = x^k_v = a$, $\forall k \in \{0, \ldots, p-1\}$.

Hence, $\forall u \in N^+_{G^F}(v)$,

$$f^{\hat{s}}_u(x^k_w : w \in N^-_{G^F}(u)) = f^{\hat{s}}_u(x^k_w : w \in N^-_{G^F}(u) \setminus \{u\}, a) = f^s_u(x^k_w : w \in N^-_{G^F}(u))$$

Besides, $f^{\hat{s}}_u(x^k) = f^s_u(x^k)$, $\forall u \in N^+_{G^F}(u) \cup \{u\}$. Therefore, $\mathcal{C} \in LC(F, \hat{s})$.

Case ii. $\forall u \in N^-_{G^F}(v), \mathrm{lab}_s(u, v) = \oplus$ is analogous.

2.- We must consider two cases:

Case i. $\forall\, u \in N^-_{G^F}(v)$: $\mathrm{lab}_s(u, v) = \ominus$ or $\forall\, u \in N^-_{G^F}(v)$: $\mathrm{lab}_s(u, v) = \oplus$. This case is covered by condition 1, where all the inputs have the same label.

CASE II. $\exists\, u_1 \in N^-_{G^F}(v)$, $\mathrm{lab}_s(u,v) = \oplus$ and $\exists\, u_2 \in N^-_{G^F}(v)$, $\mathrm{lab}_s(u,v) = \ominus$.

Let be $N^-_{\oplus}(v) = \{u \in N^-_{G^F}(v)\colon \mathrm{lab}_s(u,v) = \oplus\} \neq \emptyset$ and $u^* \in N^-_{G^F}(v)$ such that $s(u^*) = \max_{u \in N^-_{G^F}(v)} s(u) = M$. Given $k \in \{M+1, \ldots, \left|V\left(G^F\right)\right|\}$, we define:

$$\begin{aligned} & & \hat{s}(v) &= k, \\ \forall u \neq v, s(u) < k, & & \hat{s}(u) &= s(u), \\ \forall u \neq v, s(u) \geq k, & & \hat{s}(u) &= s(u)+1. \end{aligned}$$

Observe that, if $\nexists w \neq v, s(w) = s(v)$, then $\nexists w \neq v, \hat{s}(w) = s(v)$, i.e. there is no vertex updated in time $s(v)$, in this way we need to define s' such that:

$$s'(u) = \begin{cases} \hat{s}(u) & \text{if } \hat{s}(u) < s(v), \\ \hat{s}(u) - 1 & \text{if } \hat{s}(u) > s(v). \end{cases}$$

We have that:

$$\forall\ u \in N^-_{G^F}(v)\colon\ \mathrm{lab}_{\hat{s}}(u,v) = \ominus.$$

Then,

$$\begin{aligned} f^{\hat{s}}_v\left(x^0_u\colon j \in N^-_{G^F}(v)\right) &= f_v\left(x^1_u\colon u \in N^-_{G^F}(v)\right) \\ &= f_v\left(x^0_u\colon j \in N^-_{\oplus}(v);\ x^1_u\colon u \in N^-_{G^F}(v) \setminus N^-_{\oplus}(v)\right) \\ &= f^s_v\left(x^0\right) = x^1_v. \end{aligned}$$

In the same way, we can prove by induction on k that

$$\forall k \in \{0, \ldots, p-1\} : F^{\hat{s}}(x^k) = F^s(x^k) = x^{k+1}.$$

Therefore, $\mathcal{C}$ is also a limit cycle of $\hat{N}$. □

Corollary 3.11. *Let be F a Boolean function, $\mathcal{C} \in LC(F, s^p)$ and $v \in V(G^F)$ a frozen node in $\mathcal{C}$. Then, there exists an update schedule s with $[s]_{G^F} \neq [s^p]_{G^F}$ and such that $\mathcal{C} \in LC(F, s)$.*

Proof. Straightforward from the previous theorem. □

Observe that both conditions in Theorem 3.10 are of different kind. The first one is related to the update digraph and the second one to the limit cycle. Furthermore, note that Theorem 3.10 is also valid for a limit cycle set. In this case, Z corresponds to the intersection of the frozen node sets of every limit cycle. Besides, if we take $W = \{u \in Z\colon N^-_{G^F}(u) \subseteq Z\}$ (the nodes in Z who satisfies the condition 2 of Theorem 3.10), we have the same result for every $U \subseteq W$ of independent nodes, applying simultaneously the update schedules of every node in U. The importance of the frozen nodes of the attractors in the robustness of Boolean networks has been previously studied by Greil et al. (2007); Kauffman (1990).

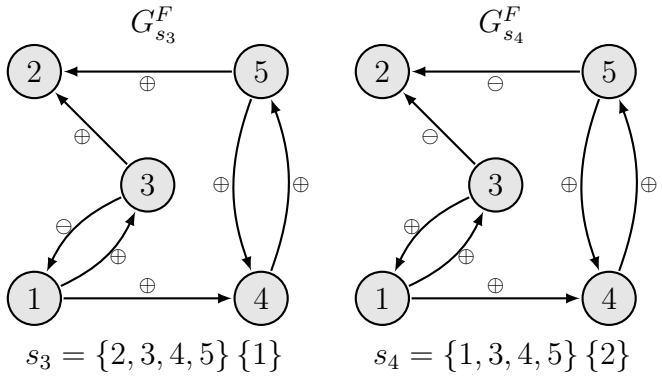


Figure 3.7: Update digraphs corresponding to Boolean networks $N_3 = (F, s_3)$ and $N_4 = (F, s_4)$ mentioned in Example 3.6.

Example 3.6. Let $N_1 = (F, s_1)$ and $\mathcal{C}$ be the Boolean network and the limit cycle defined in Example 3.3. Each frozen node $i \in \{1, 2, 3\}$ satisfies at least one of the conditions established in Theorem 3.10. Node 2 satisfies condition 1, whereas nodes 1 and 3 satisfy condition 2. Therefore, we can define at least three new update schedules s_2, s_3 and s_4, each one in a different equivalence class, such that $\mathcal{C}$ is also a limit cycle of the networks $N_i = (F, s_i)$, $i \in \{2, 3, 4\}$. Update schedule s_2 is described in Example 3.3 and update schedules s_3 and s_4 are shown in Figure 3.7.

Chapter 4

Limit cycle existence problems with deterministic update schedules in Boolean networks

In this chapter, we are interested in studying two problems: given a Boolean function, there exists an update schedule such that the given Boolean network updated under it generates limit cycles? and, there exists an update schedule such that the given Boolean network updated under does not yield any limit cycles? The specific problem of determining the existence of limit cycles of a Boolean network with parallel update is known to be NP-Hard (Just, 2006). Here, we study this problem in the case of other kinds of update schedules (sequential and block-sequential).

Many theoretical and analytic studies have been done about the limit cycles of a Boolean network when different update schedules are used (Aracena et al., 2013b; Demongeot et al., 2008; Goles and Noual, 2012; Elena, 2009; Macauley and Mortveit, 2009). Most of them show that the limit cycles are very sensitive to changes in the update schedule of the network. For instance, in Goles and Salinas (2008) is proved that for networks without negative loops it is not possible that the parallel and the sequential update share limit cycles. In Goles and Matamala (1993) it was proved that a symmetric neuronal network can simulate any arbitrary neuronal network. In Goles and Noual (2012) was studied the dynamics of disjunctive networks updated under block-sequential updated schedules and a classification of the dynamics according to the topology of the interaction digraph was given. In Aracena et al. (2009), equivalence classes of deterministic update schedules were defined according to the labeled digraph associated to a given Boolean network (update digraph). It was also proved that two schedules in the same class yield the same dynamical behavior of a given Boolean network. In Aracena et al. (2013b), as a continuation of the previous work mentioned, it was studied the robustness of the limit cycles (since fixed point are invariant under the update schedule used) of Boolean networks updated under different updates schedules.

In this context, we prove that without restrictions on the interaction digraph, this prob-

lem is NP-Hard even in the case of AND-OR networks for any kind of update schedule (synchronous, sequential and block-sequential). Besides, we give certain families of the networks in which each problem is polynomial, this classes are in AND-OR networks and depend strongly on the topology of the interaction digraph of the network. polynomial, these classes are in AND-OR networks and depend strongly on the topology of the interaction digraph of the network.

About the non existence problem, we prove that it is NP-Complete even for AND-OR functions. We also give a characterization of the existence of solution of the OR case and give some insights to its complexity.

4.1. Limit Cycle Existence Problem

In this section we study the complexity of deciding when there exists an update schedule that generates limit cycles when a given Boolean function is updated under it. That is, we are going to study the following problem

LIMIT CYCLE EXISTENCE PROBLEM **(LCE):** Given a set V of n elements and $F = (f_v)_{v \in V} : \{0,1\}^n \to \{0,1\}^n$. Does there exists an update schedule s such that $LC(F, s) \neq \emptyset$?

MON LCE, **AND-OR LCE** and **OR LCE** problems are the corresponding LCE problems when F is a monotonic, an AND-OR and an OR function, respectively.

A specific and directly related problem is to determine the existence of a limit cycle in a given Boolean network with synchronous schedule. This problem was proved to be NP-Hard even for AND-OR functions (Just, 2006). Here, we are interested in determining for a given Boolean network the existence of a deterministic update schedule that yields a limit cycle.

First, we prove that the general case is NP-Hard.

Theorem 4.1. *LCE is NP-Hard*

Proof. We show that SAT $\leq_p$ LCE.

Given a normal conjunctive formula (ncf) ϕ in variables $w_1, \ldots, w_n$, we consider $F = (f_v)_{v \in V} \colon \{0,1\}^{n+3} \to \{0,1\}^{n+3}$, where $V = \{v_1, \ldots, v_n, v_\phi, z_1, z_2\}$, as follows (see Figure 4.1):

$$\begin{aligned} \forall i \in \{1, \ldots, n\}, \; f_{v_i}(x) &= x_{v_i} \\ f_{v_\phi}(x) &= \phi\left(x_{v_i} : i \in \{1, \ldots, n\}\right) \\ f_{z_1}(x) &= x_{v_\phi} \wedge x_{z_2} \\ f_{z_2}(x) &= x_{v_\phi} \wedge x_{z_1} \end{aligned}$$

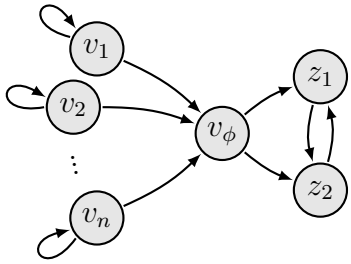


Figure 4.1: Interaction digraph of the transformation defined in Theorem 4.1.

Then, we have:

($\Longrightarrow$) Let w be such that $\phi(w) = 1$. Then, if we consider the update schedule $s = \{v_1, \ldots, v_n, v_\phi\}\,\{z_1, z_2\}$, it is clear that $\mathcal{C} = [(w, 1, 0, 1), (w, 1, 1, 0), (w, 1, 0, 1)] \in LC(F, s)$.

($\Longleftarrow$) Let us suppose that $\forall w\colon \phi(w) = 0$. Then, for every update schedule s, we have that:

- $\forall x \in \{0,1\}^{n+3},\ \forall i \in \{1, \ldots, n\} : f^s_{v_i}(x) = x_{v_i}$.
- $\forall x \in \{0,1\}^{n+3}\colon f^s_{v_\phi}(x) = 0$. Therefore,
- $\forall x \in \{0,1\}^{n+3}\colon f^s_{z_i}(F^s(x)) = 0, i \in \{1, 2\}$.

Thus, $LC(F, s) = \emptyset$, for every update schedule s. □

Now we prove that the LCE problem restricted to AND-OR functions is also NP-Hard.

Theorem 4.2. *AND-OR LCE is NP-Hard.*

Proof. We show that SAT $\leq_p$ AND-OR LCE.

Given a ncf ϕ in variables $w_1, \ldots, w_n$ with clauses $C_1, \ldots, C_m$ and let us define $F = (f_v)_{v \in V}\colon \{0,1\}^{4n+m+5} \to \{0,1\}^{4n+m+5}$ according to the following table:

See G^F in Figure 4.2. There, and trough all this work in AND-OR networks, gray nodes represent AND nodes and white nodes represent OR nodes.

Here, $\forall i \in \{1, \ldots, n\}$, nodes v_i represent literals w_i and nodes $\bar{v}_i$ represent literals $\neg w_i$.

Now, we note that:

1. For any update schedule s, $\forall x^0 \in \{0,1\}^{4n+m+5}$:
   - $\forall k \geq 1,\ \forall i \in \{1, \ldots, n\},\ \forall v \in \{v_i, \bar{v}_i\} : x^{k+1}_v = f^s_v(x^k) = x^0_v$

$v \in V$	Type	$N^-_{G^F}(v)$
$v_i,\ i \in \{1,\dots,n\}$	AND	$\{v_i\}$
$\bar{v}_i,\ i \in \{1,\dots,n\}$	AND	$\{\bar{v}_i\}$
$o_i,\ i \in \{1,\dots,n\}$	OR	$\{v_i, \bar{v}_i\}$
$a_i,\ i \in \{1,\dots,n\}$	AND	$\{v_i, \bar{v}_i\}$
A	AND	$\{o_1, \cdots, o_n\}$
O	OR	$\{a_1, \cdots, a_n\}$
$v_{C_j},\ j \in \{1,\dots,m\}$	OR	$\{v_i \colon w_i \in C_j\} \cup \{\bar{v}_i \colon \neg w_i \in C_j\}$
v_ϕ	AND	$\{v_{C_1}, \cdots, v_{C_m}\}$
z_1	AND	$\{z_2, v_\phi, A\}$
z_2	OR	$\{z_3, O\}$
z_3	OR	$\{z_1\}$

Table 4.1: Definition of F in the transformation defined in Theorem 4.2.

- $\forall k \geq 2,\ \forall i \in \{1,\dots,n\},\ \forall v \in \{o_i, a_i\} : x_v^{k+1} = f_v^s(x^k) = x_v^0$
- $\forall k \geq 2,\ \forall j \in \{1,\dots,m\} : x_{v_{C_j}}^{k+1} = f_{v_{C_j}}^s(x^k) = x_{v_{C_j}}^1$
- $\forall k \geq 2,\ \forall v \in \{A, O, v_\phi, z_1, z_2, z_3\} : x_v^{k+1} = f_v^s(x^k) = x_v^2$

2. $f_A(x) = 1 \wedge f_O(x) = 0 \iff \forall i \in \{1,\dots,n\} : x_{\bar{v}_i} = \neg x_{v_i}$

($\Longrightarrow$) If $\exists \hat{w} \colon \phi(\hat{w}) = 1$, and we consider the update schedule s and the limit cycle $\mathcal{C} = [x^0, x^1, x^0]$ as described in the following table:

$v \in V$	v_i	$\bar{v}_i$	o_i	a_i	C_j	A	O	v_ϕ	z_1	z_2	z_3
$s\,(v)$	1	1	1	1	1	1	1	1	2	3	2
x_v^0	$\hat{w}_{v_i}$	$\neg\hat{w}_{v_i}$	1	0	1	1	0	1	1	0	0
x_v^1	$\hat{w}_{v_i}$	$\neg\hat{w}_{v_i}$	1	0	1	1	0	1	0	1	1

Clearly, $\mathcal{C} \in LC(F, s)$.

($\Longleftarrow$) Let s be an update schedule such that $\mathcal{C} = \left[x^k\right]_{k=0}^p \in LC(F, s)$.

For the first note above, only nodes $z_1,\ z_2$ or z_3 can cycle. For these nodes to cycle, it is necessary that:

$$f_{v_\phi}(x^0) = 1$$
$$f_A(x^0) = 1$$
$$f_O(x^0) = 0$$

From the first equation, we have that $\phi\left(x_{v_i}^0, x_{\bar{v}_i}^0 : i \in \{1,\dots,n\}\right) = 1$. Second and third equations imply that $\forall i \in \{1,\dots,n\} : x_{\bar{v}_i} = \neg x_{v_i}$. Therefore, $\phi(x_i^0 : i \in \{1,\dots,n\}) = 1$. □

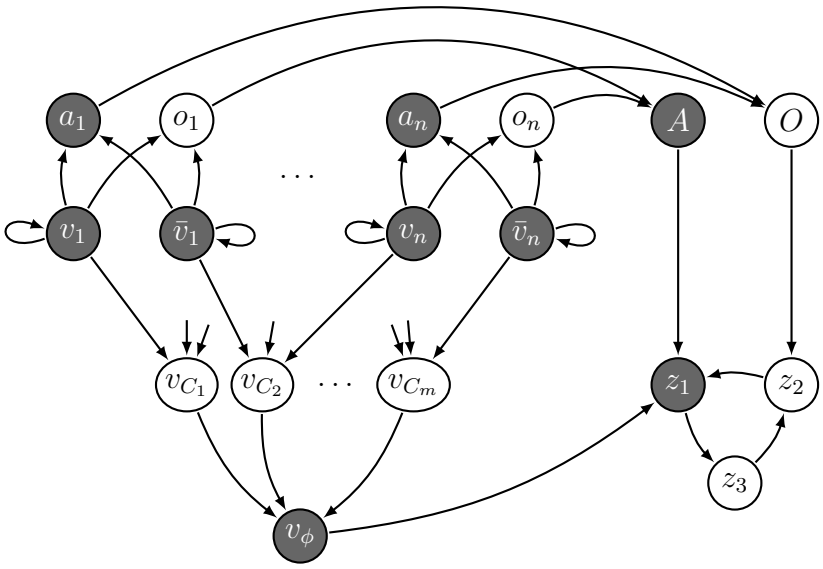


Figure 4.2: Interaction digraph of the transformation defined in Theorem 4.2.

Remark 4.1. It is clear from the proof of the previous theorem that AND-OR LCE is NP-Hard even in the following cases:

I.- Restricted to the parallel update schedule. In this case we remove the vertex z_3 and we add an arc from z_1 to z_2.

II.- Restricted to sequential update schedules.

III.- Restricted to limit cycles of length 2.

IV.- Restricted to maximum in-degree equal to 2. To see this, we just need to add intermediary nodes before every node that has in-degree greater than two as is exemplified in Figure 4.3. We note that the nodes that fulfill this condition are z_1, A, O, v_ϕ and the clause nodes. To simplify the transformation for clause nodes, we could consider 3-SAT instead of SAT. This transformation is enough because the only nodes that cycle are z_1, z_2 and z_3.

4.1.1. OR Limit Cycle Existence Problem

Now, we are going to give some partial results in order to study OR LCE.

The dynamical behavior of an OR network parallel updated is completely characterized by the following theorem, proved in Jarrah et al. (2010), but with our wording:

Theorem 4.3. *Let F be an OR function and let us consider $N = (F, s^p)$, where s^p is the parallel update schedule.*

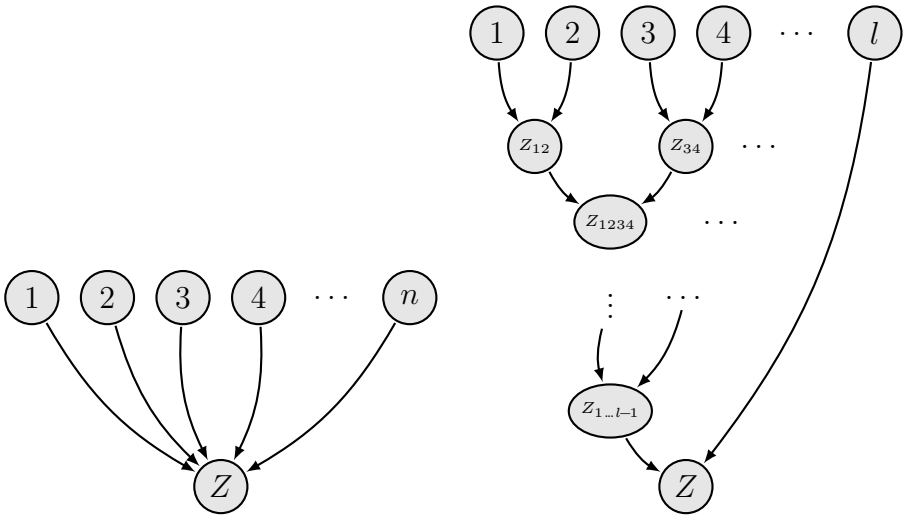


Figure 4.3: Example for odd l of the transformation mentioned in Remark 4.1 to deal with nodes with in-degree in greater than two in Theorem 4.2.

1. *If G^F is strongly connected then:*

$$LC(N) = \emptyset \text{ if and only if } G^F \text{ is primitive.}$$

2. *If G^F is not strongly connected then:*

$$LC(N) = \emptyset \text{ if and only if } G^F \text{ is primitive or it does not have cycles.}$$

OR networks updated under another kind of update schedules were studied in Goles and Noual (2012), where, in our wording, was shown the following result:

Proposition 4.4. *Let F be an OR function with symmetric G^F. Then, for all $s \neq s^p$ $LC(F, s) = \emptyset$. Furthermore, $LC(F, s^p) \neq \emptyset$ if and only if G^F is bipartite. In this case, all limit cycles are of length 2.*

We exhibit an straightforward corollary.

Corollary 4.5. *Let F be an OR function with G^F a complete digraph. Then, for every update schedule s, $LC(F, s) = \emptyset$.*

It is clear that the condition in the above proposition can be tested in polynomial time and therefore SYMMETRIC OR LCE is polynomial. We are now considering the problem in the general case.

Since in Theorem 4.3 the limit cycles dynamics of OR networks synchronously updated is polynomial characterized, our approach will consist in studying G^{F^s}. Constructing this digraph from the Boolean function it is not an easy task, since several compositions of OR functions must be made. However, it is enough to work with the most easily to construct $\mathcal{P}(G_s^F)$, since next lemma shows that they are both the same digraph.

Lemma 4.6. *If F is an OR (AND) function, then F^s is also an OR (AND) function and $G^{F^s} = \mathcal{P}(G_s^F)$.*

Proof. Straightforward from the definition of operator $\mathcal{P}$ and the fact that the composition of OR functions is also an OR function (and therefore F^s is an OR function). □

It also was proven in Goles and Noual (2012), but rewrited in our wording, the following lemma:

Lemma 4.7. *Given an update digraph G_{lab} with $G = (V, A)$ and let us consider $\mathcal{P}(G_{\text{lab}}) = (V, A')$. Then,*

$$(v_0, v_m) \in A' \iff \exists \{v_1, \ldots, v_{m-1}\} \subseteq V : \forall i \in \{0, \ldots, m-1\},\ (v_i, v_{i+1}) \in A \wedge$$
$$(\text{lab}(v_0, v_1) = \oplus \wedge\ \text{lab}(v_i, v_{i+1}) = \ominus,\ \forall i \in \{2, \ldots, m-1\})$$

Now we prove an equivalent relation between the cycles in an update digraph and the cycles in the parallel digraph, that will allow us to characterize the existence of solutions of OR LCE.

Lemma 4.8. *Given an update digraph G_{lab}. Then, every cycle in G_{lab} produce a cycle in $\mathcal{P}(G_{\text{lab}})$ with length the number of the $\oplus$-labeled arcs of it. Conversely, every cycle in $\mathcal{P}(G_{\text{lab}})$ comes from a cycle in G_{lab} with a number of $\oplus$-labeled arcs equal to the length of the cycle.*

Proof. Let us consider $G = (V, A)$ and $\mathcal{P}(G_{\text{lab}}) = (V, A')$

($\Longrightarrow$) Let C be a cycle in G_{lab} with N $\oplus$-labeled arcs, and let us consider:

$$\begin{aligned} A^+ &= \{e \in A(C) :\ \text{lab}(e) = \oplus\} \\ A^- &= \{e \in A(C) :\ \text{lab}(e) = \ominus\} \end{aligned}$$

If $A^- = \emptyset$, then the result es trivially true.
Let us suppose that $A^- \neq \emptyset$ and let us consider a path $P_- = [i, v_0, \ldots, v_m, j] \subseteq C$ such that $(i, v_0), (v_m, j) \in A^+$ and $(v_k, v_{k+1}) \in A^-,\ 0 \leq k < m$. This path exists since A^+ and A^- are non empty. Then, necessarily $(i, v_k) \in A',\ 0 \leq k \leq m$ (Lemma 4.7). Now, let us consider the paths:

$$P_+^i = [v_0^i, \ldots, v_{m_i}^i, k^i] \subseteq C,\ 0 \leq i < m$$

such that for each i, $(v_j^i, v_{j+1}^i) \in A^+,\ 0 \leq j < m_i$ and $(k^i, j_k^i),\ (j_0^i, v_0^i) \in A^-$. And the paths $P_-^l,\ 0 \leq l < m$, which are P_--like paths without i and j, such that:

$$C = P_+^0, P_-^0, P_+^1, P_-^1, \ldots, P_-^{m-1}, v_0^0$$

By the noted above for the path P_-, paths P_-^i generates the arcs $(v_{m_i}^i, v_0^{i+1}) \in A',\ 0 \leq i < m$. Therefore, C induce a cycle $\hat{C}$ in $\mathcal{P}(G_{\text{lab}})$ with the form:

$$\hat{C} = P_+^0 \setminus \{k^0\},\ P_+^1 \setminus \{k^1\}, \ldots, P_+^m \setminus \{k^m\}, v_0^0$$

Since $\sum_{i=1}^{m-1} \left|P_+^i \setminus \{k^i\}\right| = N$, we have that $\hat{C}$ has length N.

($\Longleftarrow$) The proof if analogous going backward and using the converse implication of Lemma 4.7.

□

Next proposition give us a characterization of the existence of solutions of OR LCE.

Proposition 4.9. *Let $N = (F, s)$ be an OR network with strongly connected G^F. Then, $LC(N) \neq \emptyset$ if and only if there exists a number $2 \leq k \leq g(G^F)$, such that each cycle in G_s^F has a multiple of k $\oplus$-labeled arcs.*

Proof.

$$\begin{aligned}
LC(N) \neq \emptyset &\iff G^{F^s} \text{ is not primitive (Theorem 4.3)} \\
&\iff k = \rho(G^{F^s}) > 1 \\
&\iff \text{every cycle in } G^{F^s} \text{ has length a multiple of } k \\
&\iff \text{every cycle in } G_s^F \text{ has a multiple of } k \ \oplus\text{-labeled arcs} \\
&\quad \text{(Lemma 4.6 and Lemma 4.8)}
\end{aligned}$$

□

We note that above characterization was also proved in Goles and Noual (2012), but with other wording and different technique.

In Section 4.2 we give some partial results on the study of the complexity of the problem defined by this characterization.

4.1.2. Polynomial cases

We are now extending the result for symmetric OR functions to symmetric AND-OR functions. The LCE problem here will be referred as SYMMETRIC AND-OR LCE.

First, we are going to show that symmetry is not a sufficient condition for the problem to be polynomial.

Proposition 4.10. *SYMMETRIC LCE is NP-Hard.*

Proof. The proof is similar to that of Theorem 4.1, with the same limit cycle $\mathcal{C}$, but changing the local activation functions by the following:

$$\begin{aligned} \forall i \in \{1,\dots,n\}\,,\ f_{v_i}(x) &= x_{v_i} \wedge x_{v_\phi} \\ f_{v_\phi}(x) &= \phi\left(x_{v_i} : i \in \{1,\dots,n\}\right) \wedge (x_{z_1} \vee x_{z_2}) \\ f_{z_1}(x) &= x_{v_\phi} \wedge x_{z_2} \\ f_{z_2}(x) &= x_{v_\phi} \wedge x_{z_1} \end{aligned}$$

In this case G^F is symmetric as shown in Figure 4.4.

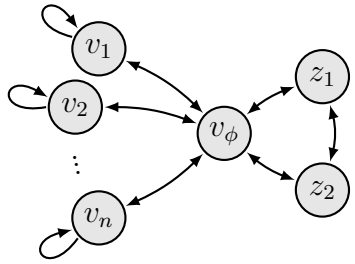


Figure 4.4: Interaction digraph of the transformation defined in Proposition 5.18.

□

Now we give some definitions in order to prove the result.

Definition 4.1. Given F an AND-OR function with symmetric G^F.

- We denote each non trivial connected component of $G\left[V_{\mathrm{OR}}(F)\right]$ by $G_1^{\mathrm{OR}},\dots,G_{k_{\mathrm{OR}}}^{\mathrm{OR}}$. We call them OR components of G^F.
- We denote each non trivial connected component of $G\left[V_{\mathrm{AND}}(F)\right]$ by $G_1^{\mathrm{AND}},\dots,G_{k_{\mathrm{AND}}}^{\mathrm{AND}}$. We call them AND components of G^F.
- We define the alternated nodes as

$$V_{\mathrm{AO}} = V \setminus \left(\bigcup_{i=1}^{k_{\mathrm{OR}}} V\left(G_i^{\mathrm{OR}}\right) \ \cup \ \bigcup_{i=1}^{k_{\mathrm{AND}}} V\left(G_i^{\mathrm{AND}}\right) \right)$$

  and we denote by $G_1^{\mathrm{AO}},\dots,G_{k_{\mathrm{AO}}}^{\mathrm{AO}}$, to the connected component of $G\left[V_{\mathrm{AO}}\right]$. We call them alternated components of G^F.
- We call to the set $\left\{G_1^{\mathrm{OR}},\dots,G_{k_{\mathrm{OR}}}^{\mathrm{OR}},G_1^{\mathrm{AND}},\dots,G_{k_{\mathrm{AND}}}^{\mathrm{AND}},G_1^{\mathrm{AO}},\dots,G_{k_{\mathrm{AO}}}^{\mathrm{AO}}\right\}$, an AOA (AND-OR ALTERNATED) decomposition of G^F.

Remark 4.2.

1. The set $\left\{V\left(G_1^{\mathrm{OR}}\right),\ldots,V\left(G_{k_{\mathrm{OR}}}^{\mathrm{OR}}\right),V\left(G_1^{\mathrm{AND}}\right),\ldots,V\left(G_{k_{\mathrm{AND}}}^{\mathrm{AND}}\right),V\left(G_1^{\mathrm{AO}}\right),\ldots,V\left(G_{k_{\mathrm{AO}}}^{\mathrm{AO}}\right)\right\}$ is a partition of $V(G^F)$.

2. Given $i\in\{1,\ldots,k_{\mathrm{AO}}\}$, we note that $\forall u\in V\left(G_i^{\mathrm{AO}}\right)$:

$$\begin{aligned} u\in V_{\mathrm{OR}} &\implies N_{G^F}^-(u)\subseteq V_{\mathrm{AND}}\\ u\in V_{\mathrm{AND}} &\implies N_{G^F}^-(u)\subseteq V_{\mathrm{OR}}\end{aligned}$$

   Therefore, the non trivial alternate components of G^F are bipartite.

Next lemma shows that every non bipartite AND or OR component of G^F is frozen in any limit cycle when the network is updated in a parallel way.

Lemma 4.11. *Given F an AND-OR function with symmetric G^F, $\mathcal{C}\in LC(F,s^p)$ and D an OR or an AND component of G^F. If D is non bipartite, then every node in $V(D)$ is frozen in $\mathcal{C}$.*

Proof. Let $\mathcal{C}=\left[x^k\right]_{k=0}^p\in LC(F,s^p)$ and D be a non bipartite OR component of G^F (the AND case is analogous). Then, there exists cycle of nodes $C=v_1\ldots v_{2N+1}v_1$ in D (cycle of odd length).

Observe that if there exists a path of length l from a node u to a node v and $x_u^t=1$ then $x_v^{t+l}=1$. Hence, for all node $v_i\in V(C)$, $v_i^k=1\Rightarrow v_i^{k+2N+1}=1$. Besides, since G^F is symmetric, for all $v\in V(D)$, and for all $k\in\{0,\ldots,p-1\}$, $x_v^k=1\Rightarrow x_v^{k+2}=1$. Therefore, if there exists $v\in V(C)$ and $k\in\{0,\ldots,p-1\}$ such that $x_v^k=1$, then for all $v\in V(C)$ and for all $k\in\{0,\ldots,p-1\}$, $x_v^k=1$. Thus, every node in the cycle C is frozen. Finally, since G^F is strongly connected, the result holds. □

Observe that, the neighbors nodes of $V(D)$ are not involved in the property of every node in $V(D)$ is frozen in $\mathcal{C}$, but in the value of the nodes of $V(D)$ in $\mathcal{C}$.

Next Theorem give a polynomial testable characterization of when a symmetric AND-OR function can generate limit cycles when it is synchronously updated.

Proposition 4.12. *Given F an AND-OR function with symmetric G^F. Then, $LC(F,s^p)\neq\emptyset$ if and only if there exists a bipartite element in the AOA decomposition of G^F.*

Proof. Let F be an AND-OR function with symmetric G^F.

($\Longrightarrow$) Let $\mathcal{C}\in LC(F,s^p)$. We have two cases:

1.- There exists an OR or an AND bipartite component of G^F, and therefore the result holds.

2.- All OR and AND components of G^F are non bipartite. By Lemma 4.11, all of them are frozen in $\mathcal{C}$. Thus, there exists an alternated component of G^F.

   If every alternated component of G^F is trivial, then each one of them has only frozen incoming neighbors, and therefore are also frozen. Then, there exists a non trivial alternated component of G^F which, by Remark 4.2, is bipartite.

($\Longleftarrow$) Let D be a bipartite element in the AOA decomposition of G^F.

1.- Let us suppose that D is an alternated connected component of G^F. We note that

$$\forall v \in V_{\text{OR}} \cap V(D) : N^-_{G^F}(v) \subseteq \bigcup_{i=1}^{k_{\text{AND}}} V\left(G_i^{\text{AND}}\right)$$

$$\forall v \in V_{\text{AND}} \cap V(D) : N^-_{G^F}(v) \subseteq \bigcup_{i=1}^{k_{\text{OR}}} V\left(G_i^{\text{OR}}\right)$$

If we consider $\mathcal{C} = [x^0, x^1, x^0]$, defined according to Table 4.2, it is clear that $\mathcal{C} \in LC(F, s^p)$.

$\boldsymbol{v \in V\left(G^F\right)}$	$\boldsymbol{V_{\text{OR}} \cap V(D)}$	$\boldsymbol{V_{\text{AND}} \cap V(D)}$	$\boldsymbol{V\left(G^F\right) \setminus V(D)}$
$\boldsymbol{x_v^0}$	1	0	0
$\boldsymbol{x_v^1}$	0	1	0

Table 4.2: Limit cycle from Proposition 4.12 if an alternated component of G^F is considered.

2.- Let us suppose that D is an OR component of G^F.

Let us denote by D^1 and D^2 the two sets of the bi-partition of D, and by V_T the set that contains the nodes of all the trivial alternated components of G^F. We define the following sets:

$$V_T^1 = \left\{v \in V_T : N^-_{G^F}(v) \subseteq V(D^1)\right\} \subseteq V_{\text{AND}}$$
$$V_T^2 = \left\{v \in V_T : N^-_{G^F}(v) \subseteq V(D^2)\right\}$$

Now, we define $\mathcal{C} = [x^0, x^1, x^0]$ according to Table 4.3, it is clear that $\mathcal{C} \in LC(F, s^p)$.

$\boldsymbol{v \in V\left(G^F\right)}$	$\boldsymbol{V\left(D^1\right) \cup V_T^1}$	$\boldsymbol{V\left(D^2\right) \cup V_T^2}$	$\boldsymbol{V\left(G^F\right) \setminus \left(V(D) \cup V_T^1 \cup V_T^2\right)}$
$\boldsymbol{x_v^0}$	1	0	0
$\boldsymbol{x_v^1}$	0	1	0

Table 4.3: Limit cycle from Proposition 4.12 if a bipartite OR component of G^F is considered.

3.- If D is an AND component of G^F, then the proof is analogous that in the OR case. □

Next proposition shows that if a symmetric AND-OR function does not have limit cycles when parallel updated, then it does not have limit cycles under any update schedule.

Proposition 4.13. *Given F an AND-OR function with symmetric G^F. If $LC(F, s^p) = \emptyset$, then for every update schedule $s \neq s^p$, $LC(F, s) = \emptyset$.*

Proof. Let us suppose that $LC(F, s^p) = \emptyset$.

Let $s \neq s^p$ be an update schedule. We note that, since G^F is symmetric, then $s \neq s^p$ if an only if there exists $(u, v) \in A\left(G^F\right)$ such that $s(u) < s(v)$.

Let $\mathcal{C} = \left[x^k\right]_{k=0}^{p} \in LC(F, s)$. Since $LC(F, s^p) = \emptyset$, then all elements in the AOA decomposition of G^F are not bipartite, by Proposition 4.12. Therefore, by Lemma 4.11, every OR and AND component updated in parallel is frozen in C. Besides, by Proposition 4.12, there are only trivial alternated components of G^F.

Now, let $u, v \in V_{\text{OR}}$ (the AND case is analogous) be such that $(u, v) \in A\left(G^F\right)$ and $s(u) < s(v)$. If there exists $k \in \{0, \ldots, p-1\}$ such that $x_u^k = 1$, we have that:

$$x_u^k = 1 \implies x_v^k = 1 \implies x_u^{k+1} = 1 \implies x_v^{k+1} = 1 \cdots \implies x_u^{k-1} = 1 \implies x_v^{k-1} = 1$$

Thus, u and v are frozen in $\mathcal{C}$ at value 1 as well as every node in the same connected component. Otherwise, u is frozen in $\mathcal{C}$ at value 0 as well as every node in the same connected component. In either case, all the OR component is frozen in $\mathcal{C}$.

Finally, all alternated nodes have only frozen neighbors, so they are also frozen. Therefore, every node is frozen in $\mathcal{C}$, which is a contradiction. □

Above results allow us to characterize the existence of solution of SYMMETRIC AND-OR LCE.

Theorem 4.14. *Given F an AND-OR function with symmetric G^F. Then, there exists an update schedule s such that, $LC(F, s) \neq \emptyset$ if and only if there exists a bipartite element in the AOA decomposition of G^F.*

Proof. Straightforward from Proposition 4.12 and Proposition 4.13. □

The difference between Proposition 4.4 and Proposition 4.12 is that, in the OR case, the only schedule than can cycle is the parallel one and only when the interaction digraph is bipartite, and in the AND-OR case, there are others non parallel update schedules that can cycle when there exist the bipartite element in the AOA decomposition of the interaction digraph (for instance, keeping the nodes in the bipartite element in the same block and all the other nodes sequentially), but if such bipartite element does not exist, then no update schedule can cycle.

Corollary 4.15. *SYMMETRIC AND-OR LCE is polynomial.*

Proof. In Theorem 4.14 we characterized the existence of solution of this problem and such characterization is testable in polynomial time. □

4.1.3. Cyclic equivalence classes of updates schedules

In this section, we define another kind of equivalence class of update schedules related to the existence of limit cycles that are generated when a cyclic permutation of the blocks

of an update schedule is done. In this context, sufficient conditions in Boolean networks are given in Macauley and Mortveit (2009) for when two non-equivalent sequential updates induce topologically conjugated limit cycles. Here, we extend these results to block-sequential update schedules.

Proposition 4.16. *Let* $F = (f_v)_{v\in V} : \{0,1\}^v \to \{0,1\}^n$ *be a Boolean function and* $\{B_j\}_{j=1}^m$ *a partition of* V*. Let us consider* $s_0 = B_1 \cdots B_m$*,* $s_j = B_{j+1}\cdots B_m B_1 \cdots B_j$ *and* $N_j = (F, s_j)$*,* $j \in \{0,\dots,m-1\}$*. Then:*

i.- $\forall i,j \in \{0,\dots,m-1\}\colon LC(N_i) \neq \emptyset \iff LC(N_j) \neq \emptyset$,

ii.- $\forall i,j \in \{0,\dots,m-1\}\colon |LC(N_i)| = |LC(N_j)|$,

iii.- $\forall i,j \in \{0,\dots,m-1\}$, $\forall \mathcal{C} = \left[x^k\right]_{k=0}^p \in LC(N_i)$, $\exists \hat{\mathcal{C}} = \left[\hat{x}^k\right]_{k=0}^{\hat{p}} \in LC(N_j)\colon p = \hat{p}$.

Proof.

I.- We prove first the result for N_0 and N_1.

($\Longrightarrow$) Let be $\mathcal{C} = \left[x^k\right]_{k=0}^p \in LC(N_0)$. Let us denote $y_j^k = \left(x_v^k\right)_{v\in B_j}$.
We note that $\forall v \in B_1\colon x_v^{k+1} = f_v^{s_1}(x^k) = f_v\left(y_1^{k+1},\dots,y_{j-1}^{k+1},y_j^k,\dots,y_m^k\right)$.
Now, let us define $\hat{\mathcal{C}} = \left[\hat{x}^k\right]_{k=0}^p$ as:

$$\forall v \in V\colon \hat{x}_v^k = \begin{cases} x_v^{k+1} & \text{if } v \in B_1 \\ x_v^k & \text{if } v \notin B_1 \end{cases}$$

We will show that $\hat{\mathcal{C}} \in LC(N_1)$, by proving that $\hat{x}^{k+1} = F^{s_2}(\hat{x}^k)$, $\forall k$. Let us define $\left[\hat{y}^k\right]_{j=1}^m$ analogous to $\left[y^k\right]_{j=1}^m$.
Let be $v \in B_1$, then for every k we have that:

$$\begin{aligned} f_v^{s_2}(\hat{x}^k) &= f_v\left(\hat{y}_1^k,\, \hat{y}_2^{k+1},\, \dots,\, \hat{y}_m^{k+1}\right) \\ &= f_v(x^{k+1}) = x_v^{k+2} = \hat{x}_v^{k+1} \end{aligned}$$

Now, considering $v \in B_j \setminus B_1$ we have that:

$$\begin{aligned} f_v^{s_2}(\hat{x}^k) &= f_v\left(\hat{y}_1^k,\, \hat{y}_2^{k+1},\, \dots,\, \hat{y}_{j-1}^{k+1},\, \hat{y}_j^k,\, \dots,\, \hat{y}_m^k\right) \\ &= f_v\left(y_1^{k+1}, y_2^{k+1}, \dots, y_{j-1}^{k+1}, y_j^k, \dots, y_m^k\right) \\ &= x_v^{k+1} = \hat{x}_v^{k+1} \end{aligned}$$

Therefore, $LC(N_1) \neq \emptyset$.

($\Longleftarrow$) The proof is completely analogous to the done above.

Now, applying the above result to schedules s_j and s_{j+1} and by induction in j, it is clear that

$$LC(N_0) \neq \emptyset \iff LC(N_1) \neq \emptyset \iff LC(N_2) \neq \emptyset \cdots \iff LC(N_{m-1}) \neq \emptyset$$

II.- Straightforward from the done above.

III.- Straightforward from the proof of part i. □

Remark 4.3. Even though in above corollary the Boolean networks do not necessarily share limit cycles, the existence of a limit cycle in one of the network give us the existence of a limit cycle in every other. In this way, they form an equivalence class for the LCE problem. Besides, all elements in the same class have the same number of limit cycles, and for each limit cycles in one network, there is one in each networks of the same length.

4.2. Non-Primitive Update Digraph Problem

In this section, we are goint to study the following problem

Non-Primitive Update Digraph problem **(NPUD):** Given G a digraph. Does there exists an integer $2 \leq k \leq g(G)$ and lab: $A(G) \to \{\oplus, \ominus\}$ such that G_{lab} is an update digraph and each cycle in G_{lab} has a multiple of k $\oplus$-labeled arcs?

We recall that NPUD problem arise in Section 4.1 as a characterization of the existence of solution of OR LCE. Therefore, we have the following result.

Lemma 4.17. *OR LCE and NPUD are equivalent problems.*

Proof. Straightforward from Proposition 4.9. □

Remark 4.4.

1. Let us recall that condition each cycle in G_{lab} has a multiple of k $\oplus$-labeled arcs is equivalent to $\rho(\mathcal{P}(G_{lab})) = k$, condition that is polynomially testable (Jarrah et al., 2010). Thus, NPUD is NP, and therefore OR LCE.
2. If F is monotonic, solving NUPD will solve LCE, but the converse is not necessarily true.

The complexity of the NPUD problem lies in both, the $\oplus$-labeled arcs and the update digraph conditions. In fact, the first condition can always be solved for $k = 2$, or equivalently, an even number of $\oplus$-labeled arcs. To prove this, we need the following definitions.

Definition 4.2. We consider the following definitions

1. Given an undirected graph $U = (V, E)$, we call an *UOCES* (Undirected Odd Cycles Edge Set) to a set $F \subseteq E$ such that $G[E \setminus F]$ does not have odd length cycles.
2. We say that an UOCES F is minimal if $\forall e \in F\colon F \setminus \{e\}$ is not an UOCES.
3. Given a digraph $G = (V, A)$, we denote $U(G)$ to the undirected graph that comes from G by removing the orientation of its arcs.
4. We call F and *UOCAS* (Undirected Odd Cycles Arc Set) of a digraph G to an UOCES of the undirected graph $U(G)$.

Lemma 4.18. *Given a digraph $G = (V, A)$, there exists a label function* $\text{lab}\colon A \to \{\oplus, \ominus\}$ *such that each cycle in G_{lab} has an even number of $\oplus$-labeled arcs.*

Proof. Let F be a minimal UOCAS of G and let define lab as

$$\forall a \in A\colon \text{lab}(a) = \begin{cases} \ominus & \text{if } a \in F \\ \oplus & \text{if } a \notin F \end{cases}$$

It is clear that each cycle in G_{lab} has an even number of $\oplus$-labeled arcs. □

In this way, we can always define a label function such that the each cycle of the labeled digraph has an even number of $\oplus$-labeled arcs. However, this labeled digraph it is not always an update digraph as shown in the following example.

Example 4.1. Let us consider an OR function F with complete G^F and $|V(G^F)| > 2$. In Corollary 4.5, we prove that for every update schedule s, $LC(F, s) = \emptyset$. Therefore, NPUD in this case does not have a solution. However, the label function defined in Lemma 4.18 is valid and fulfill the first condition of NPUD.

Remark 4.5. In Goles and Noual (2012), it was shown a polynomial characterization that solves OR LCE problem in symmetric networks, that we call SYMMETRIC OR LCE. Therefore, SYMMETRIC NPUD is polynomial, which can only have a solution for $k = 2$. Thus, the open problem is the non symmetric case.

Next are two straightforward lemmas.

Lemma 4.19. *If G is a non symmetric digraph with cycles of length two, then NPUD has a solution in G if and only if it has a solution with $k = 2$.*

Lemma 4.20. *Let G be a strongly connected digraph. Then, every cycle in G has an even amount of $\oplus$ labels if and only if each path between two nodes in G has the same parity.*

Next proposition shows that there is a solution of NPUD in some digraph, then there is solution in any sub-digraph.

Proposition 4.21. *Let $G = (V, A)$ be a digraph and $e \notin A$. If NPUD has a solution in $G + e$, then it also has a solution in G with the same k.*

Proof. Let suppose that lab: $A+e \to \{\oplus, \ominus\}$ is such that $(G+e)_{\text{lab}}$ is an update digraph and every cycle in $G+e$ has a multiple of k $\oplus$-labeled arcs, with $2 \leq k \leq g(G+e)$. Then, clearly $G_{\text{lab}|_G}$ is an update digraph and every cycle in $G+e$ which does not include e has a multiple of k $\oplus$-labeled arcs. Thus, $G_{\text{lab}|_G}$ is an update digraph and every cycle in G has a multiple of k $\oplus$-labeled arcs. Therefore, $G_{\text{lab}|_G}$ is an update digraph and $\rho(\mathcal{P}(G_{lab})) = k$. □

Corollary 4.22. *If there is no solution of NPUD in a sub-digraph of G, then there is no solution in G.*

The following is an example that NPUD not necessarily has a solution for non complete digraphs.

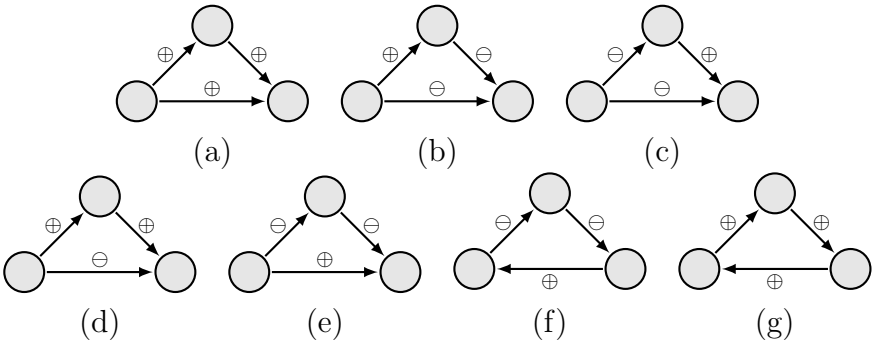


Figure 4.5: Forbidden configurations described in Example 4.2.

Example 4.2. First, we need to be sure that the labeling respect the rules of keeping the parity of label $\oplus$ in alternative paths between two nodes, and the rules for an update digraph. According to this, in Figure 4.5 we show some forbidden configurations. Configuration a,b and c are forbidden because they do not respect the parity of $\oplus$ labeling rule; d and e because they do not respect the update digraph rules; and f because we cannot have two $\ominus$ labels in a cycle of length 3.

In Figure 4.6 we can observe an OR Boolean network where each update schedule does not produce limit cycles. In fact we can observe several cycles in the graph:

- $[1, 2, 3, 1]$ of length 3.
- $[1, 2, 6, 1]$ of length 3.
- $[3, 4, 6, 3]$ of length 3.
- $[3, 5, 6, 5]$ of length 3.
- $[1, 2, 3, 4, 6, 1]$ of length 5.
- $[1, 2, 3, 4, 5, 6, 1]$ of length 6.

The maximum common divisor of the length of the cycles is 1, then there is no limit cycles in the network updated in parallel. To obtain an update schedule that produce a limit cycle, we need that every cycle in the digraph has a multiple of 2 $\oplus$ labels or that every cycle in the digraph has a multiple of $3 = g(G)$ $\oplus$ labels. Since every node in the digraph belongs to a cycle of length 3, is impossible to find an update schedule where every cycle has a length a multiple of three $\oplus$ labels, since this lead to the parallel update schedule. To find an update schedule where the length of every cycle in the associated parallel digraph is a multiple of two, we need to find a labeled digraph which is an update digraph, and also every cycle in the digraph has an even number of $\oplus$ labels and at least two. To have this two conditions we need to avoid the forbidden configurations between three nodes showed in Figure 4.5.

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

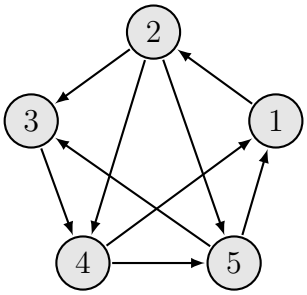


Figure 4.6: Example of an OR function without limit cycles under any update schedule.

Now we will show that this is not possible. Let us suppose that we choose $\text{lab}(5,1) = \oplus$ (See Figure 4.7 a)-d)). To avoid the forbidden configurations a and c we need that $\text{lab}(4,5) = \ominus$ and $\text{lab}(4,1) = \oplus$. Given $\text{lab}(4,5) = \ominus$, the forbidden configuration f forces $\text{lab}(5,3) = \text{lab}(3,4) = \oplus$. Since $\text{lab}(3,4) = \oplus$, to avoid the forbidden configurations a and c we need that $\text{lab}(2,4) = \oplus$ and $\text{lab}(2,3) = \ominus$. In this way, the arc $(2,5)$ it cannot be labeled, since either labeling generates the forbidden configurations b and c between nodes 2, 5 and 3.

Now, let us choose $\text{lab}(5,1) = \ominus$ (See Figure 4.7 e)-h)). We need that $\text{lab}(6,3) = \text{lab}(3,5) = \oplus$ to have at least 2 labels $\oplus$ in the cycle $[1,2,5,1]$. Since $\text{lab}(5,2) = \oplus$, we need that $\text{lab}(2,3) = \oplus$ and $\text{lab}(5,2) = \ominus$ to avoid forbidden configurations a and b. $\text{lab}(2,3) = \oplus$ forces $\text{lab}(3,4) = \ominus$ and $\text{lab}(2,4) = \oplus$ to avoid forbidden configurations a and b. In this manner, the cycle $[3,4,5,3]$ has two arcs labeled $\ominus$, which is forbidden.

It is known that non primitive monotonic networks have limit cycles when they are synchronously updated. Next we will show that in some cases, there are other classes of updates schedules that also generate limit cycles. But first, we need some previous results.

The following theorem was proved in Bang-Jensen and Gutin (2007).

Theorem 4.23. *Let G=(V,A) be a strongly connected digraph with* $\rho = \rho(G) > 1$*, then* $\forall k \in \{1,\ldots,\rho\}$, $\exists V_k \subseteq V : \forall v \in V_k,\ N_G^-(v) \subseteq V_{k+1}$ *with* $V_{\rho+1} = V_1$.

Next lemma shows that in a monotonic Boolean network, every node has at least one incoming neighbor in the parallel digraph.

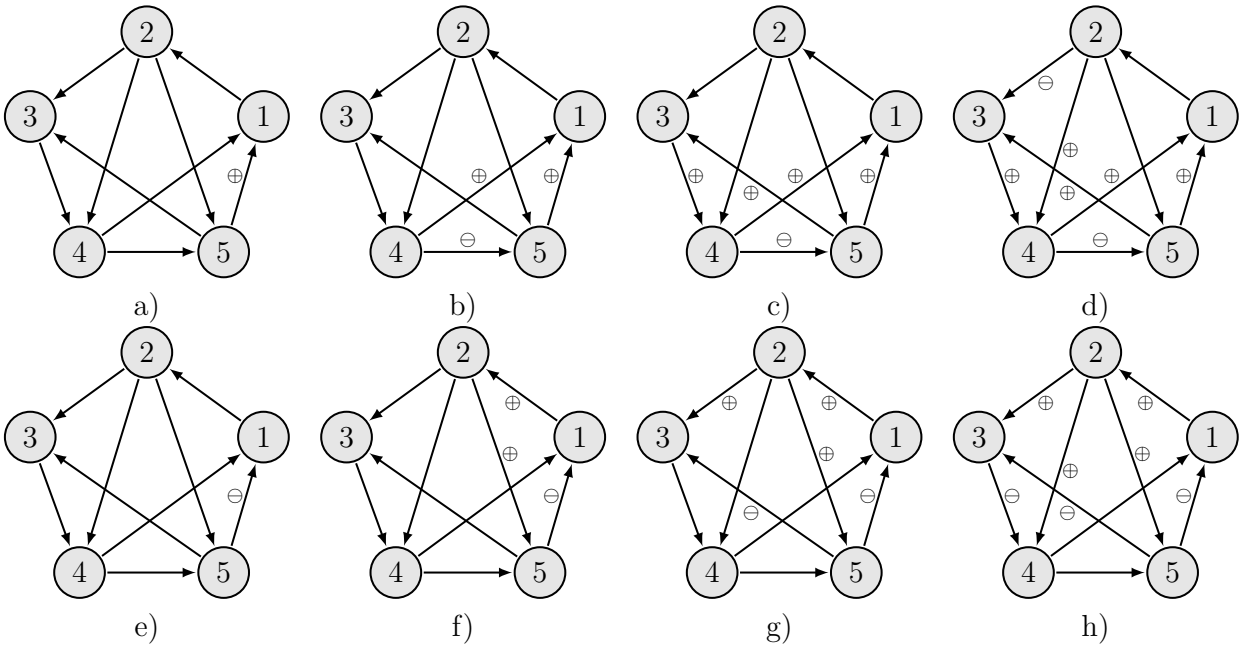


Figure 4.7: Labeling results as detailed in Example 4.2.

Lemma 4.24. *Let $N = (F, s)$ be a monotonic network. Then, $\forall v \in V\left(G^F\right), N^-_{G^{F_s}}(v) \neq \emptyset$.*

Proof. Let suppose that $\exists v \in V\colon N^-_{G^{F_s}}(v) = \emptyset$. Then, it is clear that:

$$\forall x \in \{0,1\}^n,\ \forall k\colon (f^s_v)^k(x) = 0 \quad \veebar \quad (f^s_i)^k(x) = 1$$

Therefore, $\vec{1} \veebar \vec{0}$ is not a fixed point of N, which is a contradiction. □

Theorem 4.25. *Let F be a monotonic Boolean function with $G^F = (V, A)$ strongly connected, non symmetric and $\rho = \rho(G) > 1$. Then there exists an update schedule $s \notin [s^p]$ such that $LC(F, s) \neq \emptyset$.*

Proof. By Theorem 4.23 we have a partition of V, $\{V_k\}_{k=1}^{\rho}$ such that $\forall k \in \{1, \ldots, \rho\}$, $\forall v \in V_k$, $N^-_{G^F}(v) \subseteq V_{k+1}$ with $V_{\rho+1} = V_1$. The idea is to label appropriately all arcs going from V_k, to V_{k+1} for some chosen k's.

For each k, it is easy to see that every cycle C in G^F goes trough $V_k \to V_{k+1}$, $\frac{\mathcal{L}(C)}{\rho}$ times. We can choose $2 \leq K < \rho$ sets $\left\{V_{k_j}\right\}_{j=1}^{K}$ and $\oplus$-label all arcs from V_{k_j} to V_{k_j+1}, and we will have that each cycle C in G^F has $\frac{\mathcal{L}(C)}{\rho}K$ $\oplus$-labeled arcs. Note that if $K = p$, then we have that all arcs are labeled $\oplus$, which is class of the parallel update schedule. However, even if $K < p$, this construction not always give us an update schedule not equivalent to the parallel one. To refine the result, we have to consider two cases.

1. If $\rho < g(G^F)$ or $2 < \rho = g(G^F)$, then above construction it is valid.
2. If $2 = \rho = g(G^F)$, then above construction only give the class of the parallel update schedule as a solution. However, since G^F is non symmetric then we can construct two more non equivalent update schedules in the following way:
   - We $\oplus$-label all cycles of length two.
   - We $\oplus$-label all remaining arcs from V_1 to V_2 and $\ominus$-label all remaining arcs from V_2 to V_1. This give us the first update schedule.
   - We $\ominus$-label all remaining arcs from V_1 to V_2 and $\oplus$-label all remaining arcs from V_2 to V_1. This give us the second update schedule.

□

Remark 4.6. The remaining case is when G^F is symmetric. In this case, when F is an OR function, the only solution is s^p (Proposition 4.4).

Corollary 4.22 allow us to search for forbidden configuration in order to study the NPUD problem, as done in Example 4.2. Next, is one of such configurations.

Definition 4.3. We call the following digraph $G = (V, A)$ a Bow of length m if:

- $V \equiv \{v_1, \ldots, v_{m+2}\}$
- $\forall i \in \{1, \ldots, m\} \colon (v_i, v_{i+1}), (v_{i+1}, v_i) \in A$.
- $((v_1, v_{m+2}), (v_{m+1}, v_{m+2}) \in A) \veebar ((v_{m+2}, v_1), (v_{m+2}, v_{m+1}) \in A)$

See an example of a Bow in Figure 4.8.

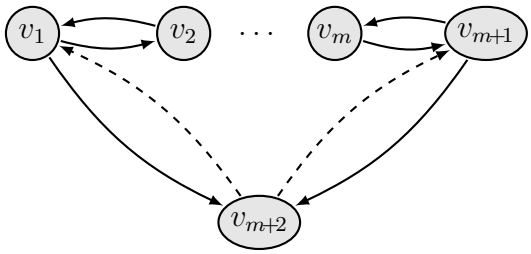


Figure 4.8: Example of a Bow of length m.

Proposition 4.26. *There is no solution of NPUD in any Bow of odd length.*

Proof. Let be $G = (V, A)$ a Bow of odd length m. We can suppose that $(v_1, v_{m+2}), (v_{m+1}, v_{m+2}) \in A$, since the other case is analogous. By Lemma 4.19, we must only consider solving NPUD for $k = 2$. Therefore, if there exists a solution of NUPD in G, every arc in A between nodes in $\{v_1, \ldots, v_{m+1}\}$ must be labeled $\oplus$. Now, we have four options:

1. To set $\text{lab}(v_1, v_{m+2}) = \oplus = \text{lab}(v_{m+1}, v_{m+2})$. In this case we have that the path $[v_1, v_{m+2}]$ has a odd number (equal to one) of $\oplus$-arcs, and the path $[v_1, v_2, \ldots, v_{m+1}, v_{m+2}]$ has an even number (equal to $m + 1$) of $\oplus$-arcs. Therefore, by Lemma 4.20, this is not a solution of NPUD.

2. To set $\text{lab}(v_1, v_{m+2}) = \ominus = \text{lab}(v_{m+1}, v_{m+2})$. In this case we have that the path $[v_1, v_{m+2}]$ has a even number of $\oplus$-arcs (equal to zero), and the path $[v_1, v_2, \ldots, v_{m+1}, v_{m+2}]$ has an odd number (equal to m) of $\oplus$-arcs. Therefore, by Lemma 4.20, this is not a solution of NPUD.

3. To set $\text{lab}(v_1, v_{m+2}) = \oplus$ and $\text{lab}(v_{m+1}, v_{m+2}) = \ominus$. In this case, the resulting labeled digraph is not an update digraph because of the forbidden cycle $[v_1, v_{m+2}, v_{m+1}, v_m, \ldots, v_2, v_1]$.

4. To set $\text{lab}(v_1, v_{m+2}) = \ominus$ and $\text{lab}(v_{m+1}, v_{m+2}) = \oplus$. In this case, the resulting labeled digraph is not an update digraph because of the forbidden cycle $[v_1, v_2, \ldots, v_{m+1}, v_{m+2}, v_1]$.

In either case, there is no solution of NPUD. □

Corollary 4.27. *Let G a digraph. If G contain as a sub-digraph a Bow of odd length, then there is no solution of NPUD for G.*

Proof. Straightforward from Proposition 4.26 and Corollary 4.22. □

Remark 4.7. It is easy to check that converse of above corollary holds if $|V(G)| \leq 4$. In this case, there could only be Bows of length 1.

Example 4.3. Converse of Corollary 4.27 does not hold if $|V(G)| \geq 5$, as we can see in the following example. We will consider the same digraph as in Example 4.2 (G), but adding the arc $(2, 1)$ (G'). Since there is no solution of NPUD in G, neither there is a solution in G'. However, there is no Bow of any length in G'.

4.3. Limit Cycle Non Existence Problem

In this section, we study the complexity of deciding when there exists an update schedule that has only fixed points as attractors. That is, we are going to consider the following problem

LIMIT CYCLE NON EXISTENCE PROBLEM **(LCNE):** Given F a Boolean function. Does there exists an update schedule s such that: $LC(F, s) = \emptyset$?

This problem is NP-Hard in the general case, as shown in the following theorem.

Theorem 4.28. *LCNE is NP-Hard.*

Proof. We are going to show that SAT $\leq_p$ LCNE.

First, we will recall the dynamical behavior of the network $N^N = (F^N, s^p)$, where:

$$\begin{aligned} \forall i \in \{1, \ldots, n-1\}, \ f_i^N(x) &= x_i \veebar \left(\bigwedge_{j=i+1}^{n} x_j \right) \\ f_n^N(x) &= \neg x_n \end{aligned}$$

The dynamic of this network consist in a single limit cycle containing all global states.

Now, let be ϕ a ncf in variables $w_1 \ldots, w_n$ and let us define the set V and the function $F = (f_v)_{v \in V} : \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ as follows:

$$\begin{aligned} \forall i \in \{1, \ldots, n-1\}, \ f_{v_i}(x) &= x_{v_i} \veebar \left(\bigwedge_{j=i+1}^{n} x_{v_j} \wedge \neg x_{v_\phi} \right), \\ f_{v_n}(x) &= \neg x_{v_n} \veebar x_{v_\phi} \\ f_{v_\phi}(x) &= \phi\left(x_{v_i} : i \in \{1, \ldots, n\}\right) \end{aligned}$$

See the interaction digraph in Figure 4.9.

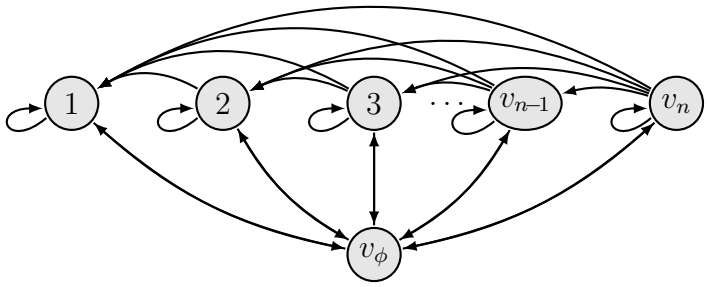


Figure 4.9: Interaction digraph of the transformation defined in Theorem 4.28.

Now we prove the equivalence,

($\Longleftarrow$) If $\forall w, \ \phi(w) = 0$, then given any initial global state $x^0 \in \{0,1\}^{n+1}$, we have that $\forall k \geq 1 \colon x_{n+1}^k = 0$. Thus, $\forall p \geq 1$ we have that:

$$x_n^{k+p} = \begin{cases} \neg x_n^k & \text{if } p \text{ odd} \\ x_n^k & \text{if } p \text{ even} \end{cases}$$

Therefore, $LC(F, s) \neq \emptyset$, for any update schedule s.

($\Longrightarrow$) Let us suppose that $\exists \hat{w}\colon \phi(\hat{w}) = 1$ and let be $s = \{v_1, \ldots, v_n\}\{v_\phi\}$.

First we must note that for any given initial global state $x^0 \in \{0,1\}^{n+1}$, if $x^k_{v_\phi} = 1$ for some $k > 0$, then

$$\begin{aligned} x^{k+1}_{v_i} &= f^s_{v_i}(x^k) = x^k_{v_i} \\ x^{k+1}_{v_n} &= f^s_{v_n}(x^k) = x^k_{v_n} \\ x^{k+1}_{v_\phi} &= f^s_{v_\phi}(x^k) = 1 = x^k_{v_\phi}. \end{aligned}$$

Therefore, $F^s(x^{k+p}) = x^k,\ \forall p \geq 1$. Now we must consider the following cases:

1.- If $x^0_{v_\phi} = 0$, then as long as $x^k_{v_\phi} = 0,\ N = (F, s)$ has the same dynamical behavior as N^N, and therefore, for some $k, x^k_{v_i} = \hat{x}_{v_i},\ \forall i \in \{1, \ldots, n\}$. Thus, $x^k_{v_\phi} = 1$ and then $F^s(x^{k+p}) = x^{k+1},\ \forall p \geq 1$.

2.- If $x^0_{v_\phi} = 1$ and $f^s_{v_\phi}(x^0) = 0$, we apply the same argument as before, for some $k > 1$.

3.- If $x^0_{v_\phi} = 1$ and $f^s_{v_\phi}(x^0) = 1$, we apply the same argument as before, for $k = 1$.

Therefore, $LC(F, s) = \emptyset$.

□

4.3.1. Polynomial cases

In this section we are going to show some polynomial cases of the LCNE problem and moreover, we show that the answer to the problem is always positive.

First, we show that the OR case of the LCNE problem referred as OR LCNE, is polynomial. In order to do that, we need the following definitions.

Definition 4.4. Given a digraph $G = (V, A)$ and $\mathcal{V} \subseteq V$ a minimum FVS of the cycles of G. With this set, we can obtain the next FAS of the cycles of G:

$$\bigcup_{v \in \mathcal{V}} N^+_G(v)$$

From this FAS, we can extract a minimal FAS, $\mathcal{A}_\mathcal{V}$, and define a labeled digraph $G_\mathcal{V} = (G, \mathrm{lab}_\mathcal{V})$ as follows:

$$\mathrm{lab}_\mathcal{V}(u, v) = \begin{cases} \oplus & \text{if } (u, v) \in \mathcal{A}_\mathcal{V} \\ \ominus & \text{i.a.c.} \end{cases}$$

In Aracena et al. (2011), was proved the following result:

Lemma 4.29. *$G_\mathcal{V}$ is an update digraph.*

It is straightforward from above definitions the following lemma:

Lemma 4.30. *There exists a sequential update schedule $s_{\mathcal{V}}$ such that $G_{\mathcal{V}} = G_{s_{\mathcal{V}}}$.*

Now the OR LCNE case.

Theorem 4.31. *Let F be an OR (AND) function. Then, there exists a sequential update schedule s such that $N = (F, s)$ has only have fixed points as attractors.*

Proof. Let $\mathcal{V}$ be a minimum FVS of the cycles of $G^F = (V, A)$ and let consider $N = (F, s)$, with $s = s_{\mathcal{V}}$ defined in Lemma 4.30. We will prove that N has only fixed point as attractors. For doing so, we will analyze the structure of $G^{F^s} = (V, A^s)$. First we note that:

$$\forall (u, v) \in A,\ \mathrm{lab}_s(u, v) = \ominus,\ \exists\ w \in \mathcal{V} \colon (w, v) \in A^s$$

Besides, all arcs $(v, u),\ v \in \mathcal{V}$ such that $\mathrm{lab}_s(v, u) = \oplus$ remains in G^{F^s}. Therefore, the strongly connected components of G^{F^s} are formed entirely by nodes of $\mathcal{V}$ with no arc incoming from the nodes of $\mathcal{V}^C$.

On the other hand, we know that every node $v \in \mathcal{V}$, has a cycle $\mathcal{C}$ in G^F such that $\mathcal{V} \cap V(\mathcal{C}) = \{v\}$. These cycles induce a loop on $v, \forall v \in \mathcal{V}$, in G^{F^s} (Corollary 4.6 and Lemma 4.8), which is monotonic, and thus G^{F^s} is primitive. In addition, F^s it is necessarily an OR (AND) function due to Corollary 4.6. Therefore, by Theorem 4.3 we have that $LC(N) = LC(F^s, s^p) = \emptyset$. □

Remark 4.8. Clearly OR LCNE is polynomial with positive answer, however, the proof above does not allow us to find the update schedule in polynomial time. Nevertheless, in Goles and Noual (2012) was shown a polynomial way to find one such update schedule, although it is not necessarily a sequential one.

Remark 4.9. In Fogelman et al. (1985) was proved that monotonic symmetric threshold networks do not have limit cycles updated under any sequential update schedule, therefore the LCNE problem in this kind of networks is polynomial with positive answer. In particular, SYMMETRIC AND-OR LCNE is polynomial.

Above results, might led us to think that the LCNE problem will always have a solution for monotonic or AND-OR functions. However, the following to examples shows that this is not the case.

Example 4.4. Let us consider the monotonic function $F = (f_i)_{i=1}^5 \colon \{0,1\}^5 \to \{0,1\}^5$ as defined in Figure 4.10. It is easy to see that no matter the update schedule applied, there are always at least two limit cycles: one having node 4 frozen at value 0 and node 5 frozen at value 1, and the other one having node 4 frozen at value 1 and node 5 frozen at value 0. However, they are not necessarily the same limit cycles in each network. We note that the loops at nodes 4 and 5 can be replaced by two cycles of length 2, adding two additional nodes, one for node 4 and one for node 5.

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

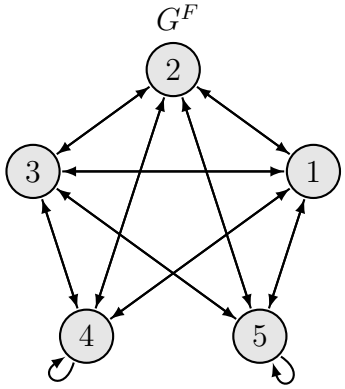


Figure 4.10: Boolean function described in Example 4.4.

Next example shows that if we remove the symmetry to AND-OR functions, the the LCNE problem does not necessarily have a solution.

Example 4.5. Let us consider the monotonic function $F = (f_i)_{i=1}^{12} \colon \{0,1\}^{12} \to \{0,1\}^{12}$ as defined in Figure 4.11. We note that nodes 4 and 5 will remain constant trough the whole dynamics. Therefore, we can decompose its dynamics into four sub-dynamics:

N_{00}: That is the dynamical behavior when node 4 and node 5 take value 0.

N_{11}: That is the dynamical behavior when node 4 and node 5 take value 1.

N_{01}: That is the dynamical behavior when node 4 takes value 0 and node 5 takes value 1.

N_{10}: That is the dynamical behavior when node 4 takes value 1 and node 5 takes value 0.

In Table 4.4 are shown the four sub-dynamics. In a) is shown the evaluation of nodes 4 and 5 and in b) is shown its simplification. The sub-digraphs associated to sub-dynamics N_{01} and N_{10} are shown in Figure 4.12a) and Figure 4.12b), respectively.

Now, we are looking for an update schedule that does not generates limit cycles for N_{00}, N_{11}, N_{01} and N_{10} simultaneously. Since N_{00} and N_{11} do not have limit cycles under any update schedule, we just need to focus in N_{01} and N_{10}.

If we analyze N_{10} (case N_{01} is analogous), we note that the only update schedules that do not generate limit cycles are those in which its update digraph have only one arc labeled $\oplus$. Without lost of generality, we can suppose that such arc is $(1,7)$. Then, as necessary conditions such that the resulting labeled digraph be an update digraph, we have that:

$$
\begin{aligned}
\mathrm{lab}(2,8) = \ominus \wedge \mathrm{lab}(8,3) = \ominus &\implies \mathrm{lab}(3,12) = \oplus \vee \mathrm{lab}(12,2) = \oplus, \\
\mathrm{lab}(3,9) = \ominus \wedge \mathrm{lab}(9,1) = \ominus &\implies \mathrm{lab}(1,10) = \oplus \vee \mathrm{lab}(10,3) = \oplus
\end{aligned}
$$

Thus, N_{01} will have necessarily two arcs labeled $\oplus$. Therefore, there is no update schedule such that $LC(F,s) = \emptyset$.

$$
\begin{aligned}
f_1(x) &= x_6 \vee x_9 \vee x_{11}\\
f_2(x) &= x_6 \vee x_7 \vee x_{12}\\
f_3(x) &= x_6 \vee x_8 \vee x_{10}\\
f_4(x) &= x_4\\
f_5(x) &= x_5\\
f_6(x) &= x_4 \wedge x_5\\
f_7(x) &= x_1 \wedge x_4\\
f_8(x) &= x_2 \wedge x_4\\
f_9(x) &= x_3 \wedge x_4\\
f_{10}(x) &= x_1 \wedge x_5\\
f_{11}(x) &= x_2 \wedge x_5\\
f_{12}(x) &= x_3 \wedge x_5
\end{aligned}
$$

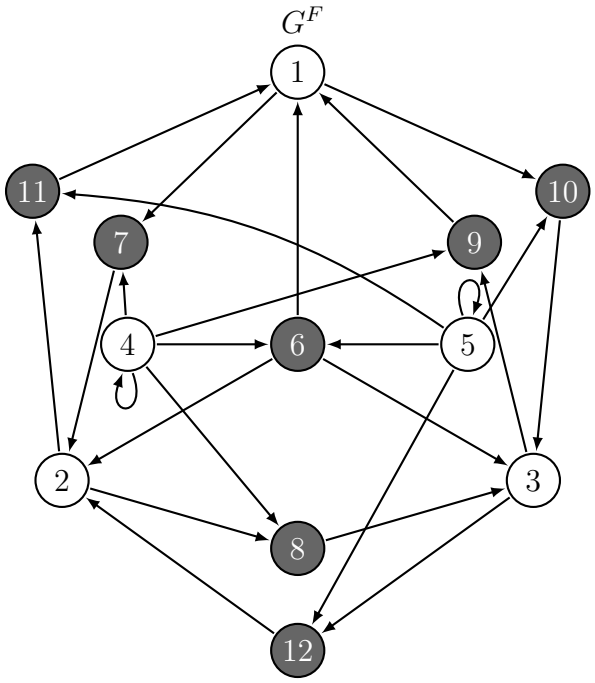


Figure 4.11: Boolean function described in Example 4.5.

	N_{00}	N_{11}	N_{01}	N_{10}
a)	$f_1(x) = x_6 \vee x_9 \vee x_{11}$ $f_2(x) = x_6 \vee x_7 \vee x_{12}$ $f_3(x) = x_6 \vee x_8 \vee x_{10}$ $f_4(x) = 0$ $f_5(x) = 0$ $f_6(x) = 0$ $f_7(x) = 0$ $f_8(x) = 0$ $f_9(x) = 0$ $f_{10}(x) = 0$ $f_{11}(x) = 0$ $f_{12}(x) = 0$	$f_1(x) = x_6 \vee x_9 \vee x_{11}$ $f_2(x) = x_6 \vee x_7 \vee x_{12}$ $f_3(x) = x_6 \vee x_8 \vee x_{10}$ $f_4(x) = 1$ $f_5(x) = 1$ $f_6(x) = 1$ $f_7(x) = x_1$ $f_8(x) = x_2$ $f_9(x) = x_3$ $f_{10}(x) = x_1$ $f_{11}(x) = x_2$ $f_{12}(x) = x_3$	$f_1(x) = x_6 \vee x_9 \vee x_{11}$ $f_2(x) = x_6 \vee x_7 \vee x_{12}$ $f_3(x) = x_6 \vee x_8 \vee x_{10}$ $f_4(x) = 0$ $f_5(x) = 1$ $f_6(x) = 0$ $f_7(x) = 0$ $f_8(x) = 0$ $f_9(x) = 0$ $f_{10}(x) = x_1$ $f_{11}(x) = x_2$ $f_{12}(x) = x_3$	$f_1(x) = x_6 \vee x_9 \vee x_{11}$ $f_2(x) = x_6 \vee x_7 \vee x_{12}$ $f_3(x) = x_6 \vee x_8 \vee x_{10}$ $f_4(x) = 1$ $f_5(x) = 0$ $f_6(x) = 0$ $f_7(x) = x_1$ $f_8(x) = x_2$ $f_9(x) = x_3$ $f_{10}(x) = 0$ $f_{11}(x) = 0$ $f_{12}(x) = 0$
b)	$f_1(x) = 0$ $f_2(x) = 0$ $f_3(x) = 0$ $f_4(x) = 0$ $f_5(x) = 0$ $f_6(x) = 0$ $f_7(x) = 0$ $f_8(x) = 0$ $f_9(x) = 0$ $f_{10}(x) = 0$ $f_{11}(x) = 0$ $f_{12}(x) = 0$	$f_1(x) = 1$ $f_2(x) = 1$ $f_3(x) = 1$ $f_4(x) = 1$ $f_5(x) = 1$ $f_6(x) = 1$ $f_7(x) = 1$ $f_8(x) = 1$ $f_9(x) = 1$ $f_{10}(x) = 1$ $f_{11}(x) = 1$ $f_{12}(x) = 1$	$f_1(x) = x_{11}$ $f_2(x) = x_{12}$ $f_3(x) = x_{10}$ $f_4(x) = 0$ $f_5(x) = 1$ $f_6(x) = 0$ $f_7(x) = 0$ $f_8(x) = 0$ $f_9(x) = 0$ $f_{10}(x) = x_1$ $f_{11}(x) = x_2$ $f_{12}(x) = x_3$	$f_1(x) = x_9$ $f_2(x) = x_7$ $f_3(x) = x_8$ $f_4(x) = 1$ $f_5(x) = 0$ $f_6(x) = 0$ $f_7(x) = x_1$ $f_8(x) = x_2$ $f_9(x) = x_3$ $f_{10}(x) = 0$ $f_{11}(x) = 0$ $f_{12}(x) = 0$

Table 4.4: Sub-dynamics of the AND-OR network described in Example 4.5

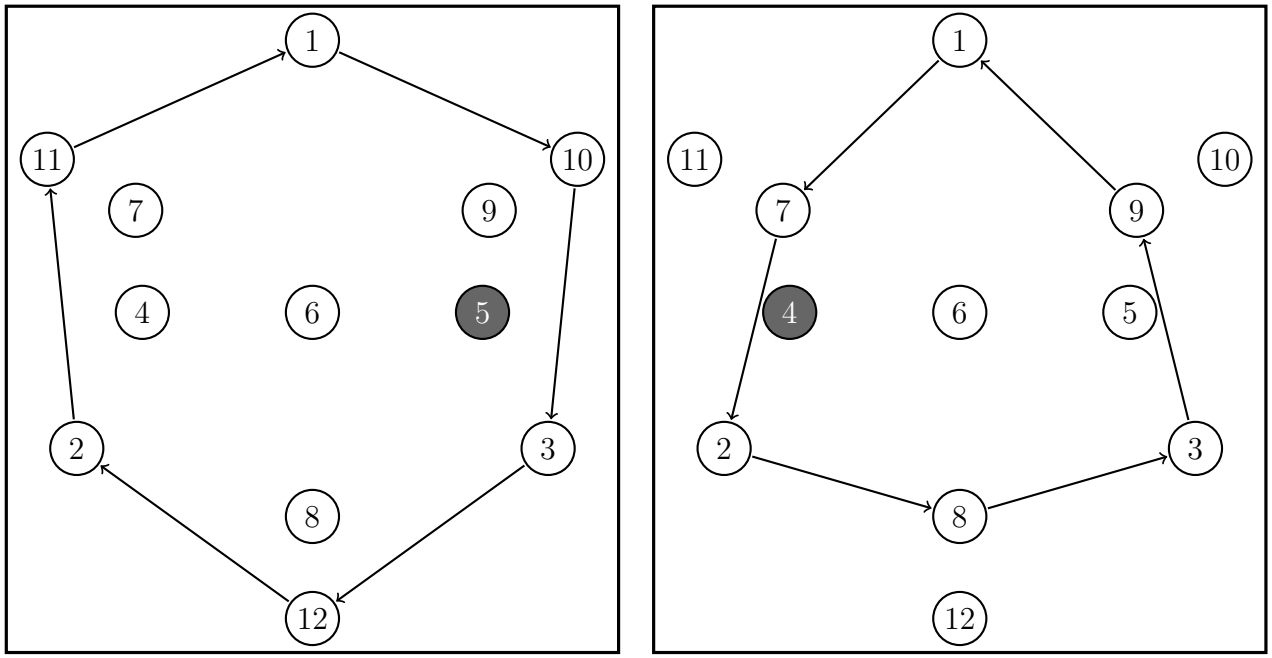


Figure 4.12: Sub-networks associated to the the sub-dynamics of the AND-OR network described in Example 4.5

Chapter 5

Feasible dynamics problems with deterministic update schedules in Boolean networks

In this chapter we are interested in studying the decision problem of when there exists an update schedule such that a given Boolean function has as a limit cycle a given sequence of global state vectors when it is updated under such update schedule. This problem arise for instance, from the reconstruction of a genetic regulatory network from observed data, as a mean to understand the function of the system (Shmulevich et al., 2002; Akutsu et al., 1999).

We start by studying the case of a simple transition, and we prove the the problem is NP-Complete even in the case of OR networks. Nevertheless, we give some polynomial cases for both the AND-OR and the OR cases. We also give an algorithm (non polynomial) that is as polynomial as we can get, to decide the OR case and that give us a solution if the answer is positive. This algorithm is also applicable to the AND-OR case, since the AND-OR case reduces to the OR case. in the case of the sequence problem, we prove that for any interaction digraph, this problem is NP-Complete even in the case of Disjunctive networks. Besides, we give certain families of the networks in which each problem is polynomial, this classes are in AND-OR networks and depend strongly on the topology of the interaction digraph of the network. polynomial, this classes are in AND-OR networks and depend strongly on the topology of the interaction digraph of the network.

We also study another related problem, and we prove that all of them are NP-Complete even in the case of OR networks.

5.1. Feasible Transition Problem

In this section we study the complexity of deciding when there exists an update schedule that generates a given transition when a given Boolean function is updated under it. That is, we are going to study the following problem

FEASIBLE TRANSITION PROBLEM **(FT):** Given a set V of n elements, $F = (f_v)_{v \in V} : \{0,1\}^n \to \{0,1\}^n$ and $x, y \in \{0,1\}^n$. Does there exists an update schedule s such that $F^s(x) = y$?

AND-OR FT and **OR FT** problems are the corresponding FT problems when F is an AND-OR and an OR function, respectively.

First we show that the problem is NP-Complete in the general case:

Theorem 5.1. *FT is NP-Complete.*

Proof. It is clear that FT is NP. To prove NP-Hardness, we show that SAT $\leq_p$ FT. Given a ncf ϕ in variables $w_1, \ldots, w_n$, we consider $x = \vec{0} = (0, \ldots, 0)$, $y = \vec{1} = (1, \ldots, 1) \in \{0,1\}^{n+1}$ and $F \colon \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ as follows (see Figure 5.1):

$$\forall i \in \{1, \ldots, n\} \colon f_{v_i}(x) = \neg x_{v_i}$$
$$f_{v_\phi}(x) = \phi(x_{v_i} \colon i \in \{1, \ldots, n\})$$

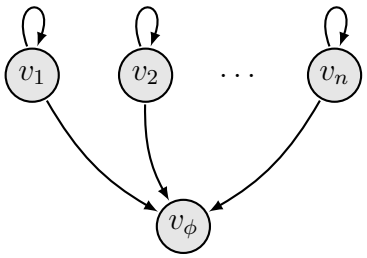


Figure 5.1: Interaction digraph of the transformation defined in Theorem 5.1.

($\Longrightarrow$) If there exists w such that $\phi(w) = 1$, then defining $s = \{v_i \colon w_i = 1\}\{v_\phi\}\{v_i \colon w_i = 0\}$, it is clear that $F^s(x) = y$.

($\Longleftarrow$) Let us suppose that there exists an update schedule s such that $F^s(x) = y$. Then we have that:

$$1 = f^s_{v_\phi}(x) = f_{v_\phi}(h^s_1(x), \ldots, h^s_n(x))$$

where

$$\forall i \in \{1, \ldots, n\} : h^s_i(x) = \begin{cases} x_{v_i} & \text{if } s(v_\phi) \leq s(v_i) \\ f_{v_i}(x) = y_{v_i} & \text{if } s(v_\phi) > s(v_i) \end{cases}$$

If we define $x^s = (h_1^s(x), \ldots, h_n^s(x))$, the global state just before node ϕ gets updated, then we have that

$$\phi(x^s) = f_{v_\phi}(x^s) = 1$$

□

For further study, we need the following definition:

Definition 5.1. Given $F\colon \{0,1\}^n \to \{0,1\}^n$, $x, y \in \{0,1\}^n$, we define for each $q, r \in \{0,1\}$:

$$V_{qr}(x,y) := \{v \in V\left(G^F\right) : x_v = q \ \wedge \ y_v = r\}$$

And the set of constant nodes:

$$V_c(x,y) = V_{00}(x,y) \cup V_{11}(x,y)$$

When there is no confusion, we will ignore the argument (x, y) in the previous definitions.

This allows to prove the following result, that establishes necessary conditions for AND-OR FT to have a solution.

Lemma 5.2. *Let $F : \{0,1\}^n \to \{0,1\}^n$ be an AND-OR function, $x, y \in \{0,1\}^n$ and s an update schedule. If $F^s(x) = y$ then, for each $v \in V\left(G^F\right)$ we have that:*

1.- If $v \in V_{OR}(F)$ and

i.- *$v \in V_{10} \cup V_{00}$, then $\forall u \in N^-_{G^F}(v)$:*

$$(u \in V_{01} \wedge s(u) \geq s(v)) \vee (u \in V_{10} \wedge s(u) < s(v)) \vee u \in V_{00}$$

ii.- *$v \in V_{01} \cup V_{11}$, then $\exists u \in N^-_{G^F}(v)$:*

$$(u \in V_{01} \wedge s(u) < s(v)) \vee (u \in V_{10} \wedge s(u) \geq s(v)) \vee u \in V_{11}$$

2.- If $v \in V_{AND}(F)$ and

i.- *$v \in V_{01} \cup V_{11}$, then $\forall u \in N^-_{G^F}(v)$:*

$$(u \in V_{10} \wedge s(u) \geq s(v)) \vee (u \in V_{01} \wedge s(u) < s(v)) \vee u \in V_{11}$$

ii.- *$v \in V_{10} \cup V_{00}$, then $\exists u \in N^-_{G^F}(v)$:*

$$(u \in V_{10} \wedge s(u) < s(v)) \vee (u \in V_{01} \wedge s(u) \geq s(v)) \vee u \in V_{00}$$

Proof. Let $v \in V\left(G^F\right)$.

1.- If $v \in V_{\mathrm{OR}}(F)$,

   I.- Let us suppose $v \in V_{10} \cup V_{00}$ and let $u \in N^-_{G^F}(v)$. Since $f^s_v(x) = 0$, necessarily $x_u = 0 \vee y_u = 0$. Now:

   - If $u \in V_{01}$, then $x_u = 0 \wedge y_u = 1$. Since $f^s_v(x) = 0$, it is necessary that $s(u) \geq s(v)$.
   - If $u \in V_{10}$, then $x_u = 1 \wedge y_u = 0$. Since $f^s_v(x) = 0$, it is necessary that $s(u) < s(v)$.
   - Otherwise, j must necessarily be in V_{00}.

   II.- Straightforward from the definition of OR functions and analogous argument as before.

2.- If $v \in V_{\mathrm{AND}}(F)$, the proof is straightforward from the definition of AND functions and analogous argument as before. □

Remark 5.1. We know Theorem 3.1 that Boolean networks updated under different updates schedules that generate the same update digraph have the same dynamical behavior. Therefore, we focus on finding an update digraph wich satisfies certain restrictions. In this way, according to the definition of an update digraph and to the established in the previous lemma, we have that for OR nodes (AND nodes), all incoming arcs of the nodes in $V_{00} \cup V_{10}$ ($V_{11} \cup V_{01}$) have their labels uniquely defined. To satisfy the necessary conditions such that $F^s(x) = y$, at least one incoming arc to the nodes in $V_{11} \cup V_{01}$ ($V_{00} \cup V_{10}$) must be chosen and labeled accordingly. It is in this choice where the complexity of the problem arises.

Next lemma shows that considering the FT problem with or without constant nodes are equivalent problems in AND-OR networks.

Lemma 5.3. *Let be* $F\colon \{0,1\}^n \to \{0,1\}^n$, *and* $x, y \in \{0,1\}^n$. *There exists* $\hat{F}\colon \{0,1\}^n \to \{0,1\}^n, \hat{x}, \hat{y} \in \{0,1\}^n$, *where* $V_c(\hat{x}, \hat{y}) = \emptyset$, *such that FT AND-OR has a solution with instance* $\langle F, x, y\rangle$ *if and only if it has a solution with instance* $\langle \hat{F}, \hat{x}, \hat{y}\rangle$.

Proof. Let be $G^F = (V, A)$. First, we implicitly define $\hat{x}$ and $\hat{y}$ by defining:

- $V_{01}(\hat{x}, \hat{y}) = V_{01}(x, y) \cup V_{11}(x, y)$.
- $V_{10}(\hat{x}, \hat{y}) = V_{10}(x, y) \cup V_{00}(x, y)$.

Clearly, $V_c(\hat{x}, \hat{y}) = \emptyset$.

Now, $\hat{F}$ is an AND-OR function defined by its interaction digraph $G^{\hat{F}} = (V, \hat{A})$ and a node set partition $\left\{\hat{V}_{\mathrm{AND}}, \hat{V}_{\mathrm{OR}}\right\}$ according to the following transformation (see an example in Figure 5.2).

First, we remove from A all arcs outgoing from the constant nodes since they are not necessary for establishing the value of their outgoing neighbors. That is:

$$\hat{A} = A \setminus \{(u,v) \in A \colon u \in V_c\,(x,y)\}$$

Since we removed some arcs, now there might be nodes that get all its incoming arcs removed. If for instance, one of these nodes is an OR node that changes from zero to one, i.e., it belongs to $V_{01}\,(x,y)$, FT AND-OR with instance $\langle \hat{F}, \hat{x}, \hat{y} \rangle$ will not have a solution since the OR function is defined as the constant function zero when the node have empty incoming neighborhood, nevertheless if it has or not solution with instance $\langle F, x, y \rangle$. Thus, with the objective of do not change the existence of solutions, we redefine the local functions of the nodes in the sets

$$U_{\text{OR}} = \left\{ v \in V_{\text{OR}} \cap V_{01}\,(\hat{x},\hat{y}) : N^-_{G^{\hat{F}}}(v) = \emptyset \wedge N^-_{G^F}(v) \neq \emptyset \wedge N^-_{G^F}(v) \nsubseteq V_{00}\,(x,y) \right\}$$
$$U_{\text{AND}} = \left\{ v \in V_{\text{AND}} \cap V_{10}\,(\hat{x},\hat{y}) : N^-_{G^{\hat{F}}}(v) = \emptyset \wedge N^-_{G^F}(v) \neq \emptyset \wedge N^-_{G^F}(v) \nsubseteq V_{11}\,(x,y) \right\}$$

according to:

$$\hat{V}_{\text{OR}} = U_{\text{AND}} \cup (V_{\text{OR}} \setminus U_{\text{OR}})$$
$$\hat{V}_{\text{AND}} = U_{\text{OR}} \cup (V_{\text{AND}} \setminus U_{\text{AND}})$$

Now we prove the equivalence,

($\Longrightarrow$) Let s be solution of AND-OR FT with instance $\langle F, x, y \rangle$. Then if we consider $\hat{s} \in [s']_{G^{\hat{F}}}$, where $\text{lab}_{s'}$ is defined as

$$\forall a \in \hat{A} \colon \text{lab}_{s'}(a) = \text{lab}_s(a)$$

it is clear from the definitions of an AND-OR function, $\hat{F}, \hat{x}$ and $\hat{y}$ that $\hat{F}^{\hat{s}}(\hat{x}) = \hat{y}$.

($\Longleftarrow$) Let s be solution of AND-OR FT with instance $\langle \hat{F}, \hat{x}, \hat{y} \rangle$. The update schedule s generates certain labels in all arcs removed from G^F.

1. Let $v \in V_{00}\,(x,y) \cap V_{\text{OR}}$, $u \in N^+_{G^F}(v) \cap V_{10}\,(\hat{x},\hat{y}) \cap V_{\text{OR}}$ and $w \in N^+_{G^F}(v) \cap V_{01}\,(\hat{x},\hat{y}) \cap V_{\text{OR}}$:
   - Whichever the label of arc (v,u) might be, the node u will receive from the node v the value 0 that it needs.
   - Since, for the node w the value 0 of the node v is useless, it does not matter at all the label that the arc (v,w) might take.
2. The analysis is analogous in the other cases.

□

Remark 5.2. It is clear from the proof of the previous lemma that a solution in one problem is also a solution in the other one.

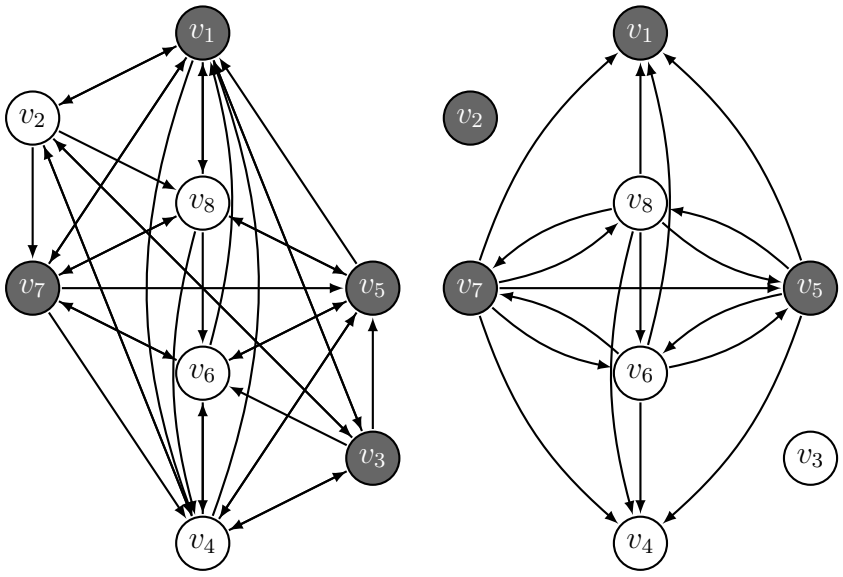


Figure 5.2: Example of the transformation defined in Lemma 5.3 as detailed in Example 5.1.

Example 5.1. Let us consider an AND-OR function F with interaction digraph as shown in Figure 5.2 a).

If we consider $V_{11} = \{1, 2\}$, $V_{00} = \{3, 4\}$, $V_{01} = \{5, 6\}$ and $V_{10} = \{7, 8\}$, then the interaction digraph of $\hat{F}$ is shown in Figure 5.2 b).

Now, we show that the AND-OR case reduces to the OR case. Next, we prove that AND-OR FT is NP-Complete, and therefore OR FT it is too.

Proposition 5.4.

$$\textit{AND-OR FT} \leq_p \textit{OR FT}$$

Proof. Let $F : \{0,1\}^n \to \{0,1\}^n$ be an AND-OR function and $x, y \in \{0,1\}^n$. By Lemma 5.3, we can suppose $V_c(x, y) = \emptyset$.

We construct an OR function $\hat{F}\colon \{0,1\}^{n+m} \to \{0,1\}^{n+m}$, $\hat{x}, \hat{y} \in \{0,1\}^{n+m}$, where m is defined below, such that:

$$\exists s\colon F^s(x) = y \iff \exists \hat{s}\colon \hat{F}^{\hat{s}}(\hat{x}) = \hat{y}$$

We define $\hat{F}$ by its interaction digraph $G^{\hat{F}} = (\hat{V}, \hat{A})$ constructed from $G^F = (V, A)$.

The idea of the transformation is the following: OR nodes remain the same, and for each AND node we add an structure that allow us to simulate the AND behavior in one transition with only OR nodes. This structure includes the original AND node: v, a copy of it: $\hat{v}$, the incoming neighbors denoted $N^-_{G^F}(v) = \left\{u^v_1, \ldots, u^v_{m(v)}\right\}$, and a copy of each one of

them: $\hat{u}_1^v, \ldots, \hat{u}_{m(v)}^v$, where $m(v) = \left|N_{G^F}^-(v)\right|$. Therefore, $m = |V_{\text{AND}}| + \sum_{v \in V_{\text{AND}}} m(v)$. The connections are defined according to the incoming neighborhood as detailed in Table 5.1 (see and example in Figure 5.3).

$w \in \hat{V}$	$N_{G^{\hat{F}}}^-(w)$
$v \in V_{\text{OR}}$	$N_{G^F}^-(v)$
$v \in V_{\text{AND}}$	$\{\hat{v}\}$
$\hat{v} \colon v \in V_{\text{AND}}$	$\left\{\hat{u}_1^v, \ldots, \hat{u}_{m(v)}^v\right\}$
$\hat{u}_k^v \colon v \in V_{\text{AND}}, k \in \{1, \ldots, m(v)\}$	$\{u_k^v\}$

Table 5.1: Definition of $G^{\hat{F}}$ defined in Proposition 5.4.

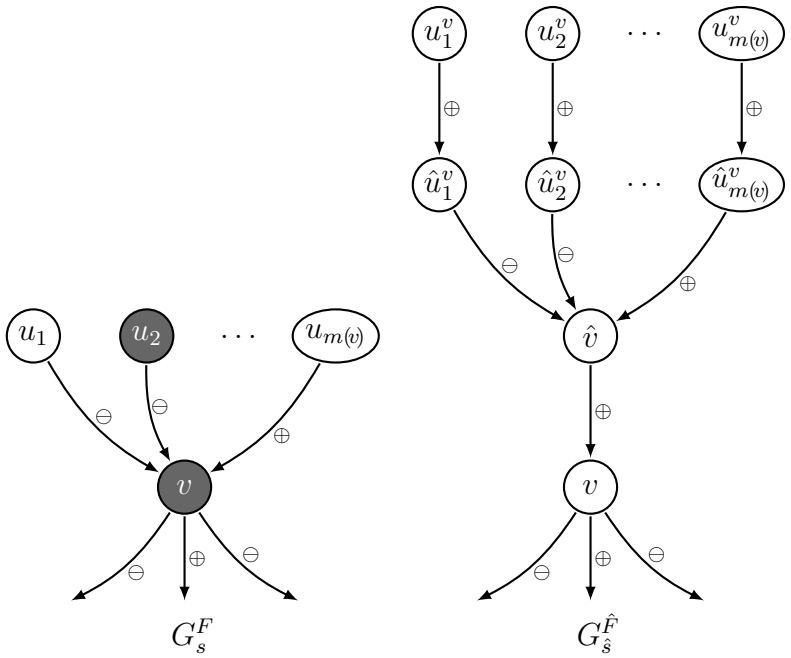


Figure 5.3: Example of the structure associated to an AND node v in the transformation defined in Proposition 5.4.

Finally, we define $\hat{x}, \hat{y}$ by

$$\forall v \in V \colon \hat{x}_v = x_v \wedge \ \hat{y}_v = y_v$$

and

$$\forall \hat{u} \in \hat{V} \setminus V \colon (\hat{x}_{\hat{u}} = \neg x_u) \wedge \ (\hat{y}_{\hat{u}} = \neg y_u)$$

That is,

$$\forall \hat{u} \in \hat{V} \setminus V \colon u \in V_{qr}(x, y) \Longrightarrow \hat{u} \in V_{rq}(\hat{x}, \hat{y})$$

since $q = \neg r$.

By Lemma 5.2, we have that for every update schedule $\hat{s}$ such that $\hat{F}^{\hat{s}}(\hat{x}) = \hat{y}$ and for each $v \in V_{\text{AND}}$:

$$\forall k \in \{1, \ldots, m(v)\} : \hat{s}(u_k^v) \geq \hat{s}(\hat{u}_k^v) \tag{5.1}$$
$$\hat{s}(\hat{v}) \geq \hat{s}(v) \tag{5.2}$$

($\Longrightarrow$) Let us suppose that $\exists s \colon F^s(x) = y$. We define $\hat{s}$ according to:

$$\forall v \in V \colon \hat{s}(v) = s(v) \tag{5.3}$$
$$\forall \hat{v} \in \hat{V} \setminus V \colon \hat{s}(\hat{v}) = s(v) \tag{5.4}$$

It is clear that $\hat{s}$ as defined above fulfill conditions (5.1) and (5.2).

Now, since each AND node v do not changes its outgoing neighbors, and the nodes in $N^-_{G^F}(v)$ do not change their incoming neighbors, we have that every node in $N^-_{G^F}(v)$ do not change their value when applying $\hat{F}$. Then,

- $\forall v \in V_{\text{AND}}$:

$$\forall k \in \{1, ..., m(v)\}, \ (5.1) \Longrightarrow \quad (u_k^v \in V_{qr}(x, y) \Longrightarrow \hat{u}_k^v \in V_{rq}(\hat{x}, \hat{y})) \tag{5.5}$$
$$(5.3), \ (5.4), (5.5) \Longrightarrow \quad (v \in V_{qr}(x, y) \Longrightarrow \hat{v} \in V_{rq}(\hat{x}, \hat{y})) \tag{5.6}$$
$$(5.2), (5.6) \Longrightarrow \quad (v \in V_{qr}(x, y) \Longrightarrow v \in V_{qr}(\hat{x}, \hat{y})) \tag{5.7}$$

- $\forall v \in V_{\text{OR}}$: by

$$v \in V_{qr}(x, y) \Longrightarrow v \in V_{qr}(\hat{x}, \hat{y}) \tag{5.8}$$

Therefore, $\hat{F}^{\hat{s}}(\hat{x}) = \hat{y}$. We note that, $\forall s' \in [\hat{s}]_{G^{\hat{F}}} : \hat{F}^{s'}(\hat{x}) = \hat{y}$.

($\Longleftarrow$) If OR FT has a solution with instance $\langle \hat{F}, \hat{x}, \hat{y} \rangle$, we take the update schedule $\hat{s}$ with the least amount of blocks. Then, by (5.1) and (5.2), we can define s according to:

$$\forall v \in V \colon s(v) = \hat{s}(v)$$
$$\forall \hat{v} \in \hat{V} \setminus V \colon s(v) = \hat{s}(\hat{v})$$

The analysis is analogous to the done before, since the converse of the right side of (5.5), (5.6), (5.7) and (5.8) hold. □

Now we show that the AND-OR case is NP-Complete.

Theorem 5.5. *AND-OR FT is NP-Complete.*

	$w \in V$	type	x_w	y_w	$N^-_{G^F}(w)$
1	$v_i,\ i \in \{1,\dots,n\}$	OR	0	1	$\{z_1, v_\phi\}$
2	$\bar{v}_i,\ i \in \{1,\dots,n\}$	OR	0	1	$\{z_1, v_\phi\}$
3	$o_i,\ i \in \{1,\dots,n\}$	OR	0	1	$\{v_i, \bar{v}_i\}$
4	$a_i,\ i \in \{1,\dots,n\}$	AND	1	0	$\{v_i, \bar{v}_i\}$
5	A	AND	0	1	$\{O, o_1, \cdots, o_n\}$
6	O	OR	1	0	$\{a_1, \cdots, a_n\}$
7	$a'_i,\ i \in \{1,\dots,n\}$	AND	0	1	$\{a_i\}$
8	z_1	OR	1	0	$\{z_2\}$
9	z_2	AND	0	1	$\{z_3\}$
10	z_3	OR	1	0	$\{z_1, o_1, \cdots, o_n, a'_1, \cdots, a'_n\}$
11	z_4	OR	1	0	$\{O, v_{C_1}, \cdots, v_{C_m}\}$
12	$v_{C_j},\ j \in \{1,\dots,m\}$	OR	0	1	$\{v_i \colon w_i \in C_j\} \cup \{\bar{v}_i \colon \neg w_i \in C_j\}$
13	v_ϕ	AND	0	1	$\{v_{C_1}, \cdots, v_{C_m}\}$

Table 5.2: Definition of F in the transformation defined in Theorem 5.5.

Proof. First, as in the general case, this problem is NP. To prove NP-hardness, we show that SAT $\leq_p$ AND-OR FT.

Given a ncf ϕ in variables $w_1, \dots, w_n$, with clauses $C_1, \dots, C_m$, we define V and $F = (f_v)_{v \in V} \colon \{0,1\}^{5n+m+7} \to \{0,1\}^{5n+m+7}$, $x, y \in \{0,1\}^{5n+m+7}$ as described in Table 5.2.

Here, $\forall\, i \in \{1,\dots,n\}$, nodes v_i and $\bar{v}_i$, represent literals w_i and $\neg w_i$, respectively. Nodes $v_{C_1}, \dots, v_{C_m}$ represent the clauses. Nodes z_1, z_2, z_3, z_4 and v_ϕ, are technical nodes to make sure that each literal node is updated either before or after all clause nodes (see Figures 5.4 and 5.5). Finally, $\forall\ i \in \{1,\dots,n\}$, nodes o_i, a_i, a'_i together with nodes A and O and its connection structure with nodes z_1, z_2 and z_3 make sure that nodes v_i and $\bar{v}_i$ have opposite values when necessary (see Figures 5.4 and 5.5), using that:

$$f_A(x) = 1 \ \wedge\ f_O(x) = 0 \quad \Longleftrightarrow \quad \forall i \in \{1,\dots,n\} : x_{\bar{v}_i} = \neg x_{v_i} \quad (*)$$

($\Longrightarrow$) Let w be such that $\phi(w) = 1$, then we consider the update schedule s as defined in Table 5.3. It is clear from Table 5.3 that $F^s(x) = y$.

($\Longleftarrow$) Let s be an update schedule such that $F^s(x) = y$.

Let x^s be the global state just before node v_ϕ gets updated. Then $f_{v_\phi}\left(x^s_{v_{C_j}} : j \in \{1,\dots,m\}\right) = 1$. Besides, given the definitions of F, x and y, we have the following conditions for s:

1) $\forall i \in \{1,\dots,n\},\ s(v_i) \leq s(z_1)\ \vee s(v_\phi) < s(v_i)$.
2) $\forall i \in \{1,\dots,n\},\ s(\bar{v}_i) \leq s(z_1)\ \vee s(v_\phi) < s(\bar{v}_i)$.
3) $\forall i \in \{1,\dots,n\},\ s(v_i) < s(o_i)\ \vee s(\bar{v}_i) < s(o_i)$.

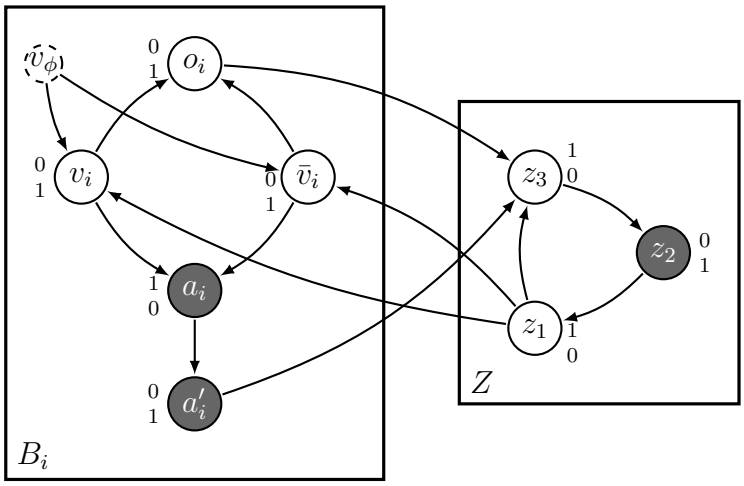


Figure 5.4: Structure that together with its connection structure with nodes A and O allows that literal nodes have opposite values when necessary (Theorem 5.5).

4) $\forall i \in \{1,\ldots,n\}\,,\ s(a_i) \leq s(v_i)\ \vee s(a_i) \leq s(\bar{v}_i)$.
5) $\forall i \in \{1,\ldots,n\}\,,\ s(o_i) < s(A)\ \wedge\ s(A) \leq s(O)$.
6) $\forall i \in \{1,\ldots,n\}\,,\ s(a_i) < s(O)$.
7) $\forall i \in \{1,\ldots,n\}\,,\ s(a'_i) \leq s(a_i)$.
8) $s(z_1) \leq s(z_2)$.
9) $s(z_2) \leq s(z_3)$.
10) $\forall i \in \{1,\ldots,n\}\,,\ s(z_3) \leq s(a'_i) \wedge s(z_3) \leq s(o_i)\ \wedge\ s(z_1) < s(z_3)$.
11) $\forall j \in \{1,\ldots,m\}\,,\ s(z_4) \leq s(v_{C_j})\ \wedge\ s(O) < s(z_4)$.
12) $\forall j \in \{1,\ldots,m\}\,, \exists i \in \{1,\ldots,n\} : s(v_i) < s(v_{C_j}) \vee s(\bar{v}_i) < s(v_{C_j})$.
13) $\forall j \in \{1,\ldots,m\}\,,\ s(v_{C_j}) < s(v_\phi)$.

If we observe the labels generated by s, we can see that only labels incoming to nodes v_{C_j}, o_i, a_i, v_i and $\bar{v}_i$ remain undefined. Now, from the previous conditions we have that:

a) Let be $i \in \{1,\ldots,n\}$ such that $s(v_i) \leq s(z_1)$, then we have:

$$s(v_i) \leq s(z_1) \overset{(10,7)}{<} s(a_i) \overset{(6)}{<} s(O) \overset{(11)}{<} s(z_4) \overset{(11)}{\leq} s(v_{C_j}) \overset{(13)}{<} s(v_\phi)$$

b) Analogously, if $s(v_\phi) < s(v_i)$, then $s(z_1) < s(v_i)$.

c) 1), a) and b) imply:

$$\forall i \in \{1,\ldots,n\} : s(v_i) \leq s(z_1)\ \veebar\, s(v_\phi) < s(v_i)$$

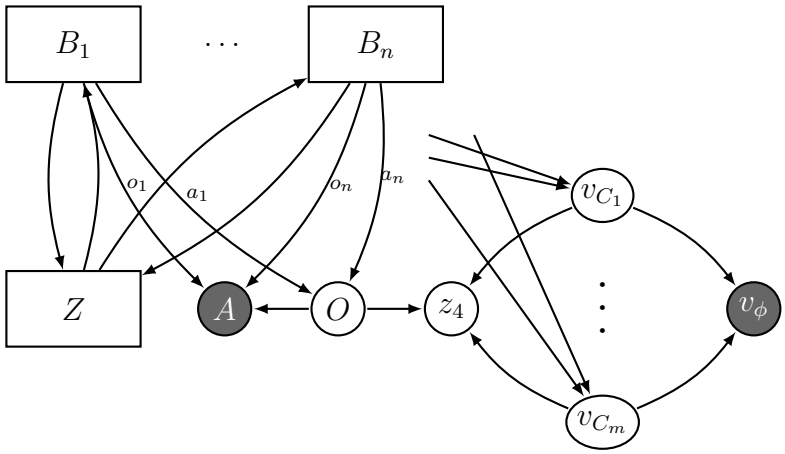


Figure 5.5: G^F. The structure connection of z_4 and v_ϕ allows that literal nodes update either before or after all clause nodes, in the transformation defined in Theorem 5.5.

d) Analogously:
$$\forall i \in \{1,\dots,n\} : s(\bar{v}_i) \leq s(z_1) \ \veebar\ s(v_\phi) < s(\bar{v}_i)$$

e) Besides, if $s(v_i) \leq s(z_1)$, then:
$$s(v_i) \leq s(z_1) \overset{(10,7)}{<} s(a_i) \overset{(4)}{\Longrightarrow} s(a_i) \leq s(\bar{v}_i) \overset{(2)}{\Longrightarrow} s(v_\phi) < s(\bar{v}_i)$$

f) Analogously, if $s(v_\phi) < s(v_i)$, then $s(\bar{v}_i) \leq s(z_1)$.

Properties c) and d) imply that when clause nodes get updated, they all have the same input, namely $x^s = \left(x^s_{v_1},\dots,x^s_{v_n},x^s_{\bar{v}_1},\dots,x^s_{\bar{v}_1}\right)$. Therefore, $f_{v_\phi}(x^s) = 1$.
Properties $(*)$, e) and f) imply that $x^s = \left(x^s_{v_1},\dots,x^s_{v_n},\neg x^s_{v_1},\dots,\neg x^s_{v_n}\right)$ and therefore $\phi(\hat{x}) = 1$, where $\hat{x} = \left(x^s_{v_1},\dots,x^s_{v_n}\right)$. □

5.1.1. OR Feasible Transition Problem

In this section we consider $F\colon \{0,1\}^n \to \{0,1\}^n$ an OR function with interaction digraph $G = (V,A)$ and $x, y \in \{0,1\}^n$. We are trying to find out an update schedule s such that: $F^s(x) = y$. As a consequence of Lemma 5.3, we can consider $V_c = \emptyset$.

Remark 5.3. We note that if $V_c \neq \emptyset$, as an extension of Lemma 5.3, we can also remove all arcs incoming to nodes in $V_{01} \cup V_{11}$ that have at least one incoming arc from a node in V_{11}, since they will get the value one that they need no matter the labels of their incoming arcs. That is, $\hat{A} = A \setminus R$, where:

$$R = \{(u,v) \in A\colon u \in V_c\} \cup \left\{(u,v) \in A\colon v \in N^+_{11} \cap (V_{01} \cup V_{11})\right\}$$
$$N^+_{11} = \bigcup_{v \in V_{11}} N^+_G(v)$$

k	$s(k)$	x_k	$F^s(x)_k$
$v_i\colon w_i = 1$	1	$0 \cdots 0$	$1 \cdots 1$
$v_i\colon w_i = 0$	11	$0 \cdots 0$	$1 \cdots 1$
$\bar{v}_i\colon w_i = 1$	12	$0 \cdots 0$	$1 \cdots 1$
$\bar{v}_i\colon w_i = 0$	2	$0 \cdots 0$	$1 \cdots 1$
z_1	3	1	0
z_2	4	0	1
z_3	5	1	0
a_i	6	$1 \cdots 1$	$0 \cdots 0$
a'_i	6	$0 \cdots 0$	$1 \cdots 1$
o_i	6	$0 \cdots 0$	$1 \cdots 1$
A	7	0	1
O	7	1	0
z_4	8	1	0
v_{C_j}	9	$0 \cdots 0$	$1 \cdots 1$
v_ϕ	10	0	1

Table 5.3: Transition table of the states defined in the transformation defined in Theorem 5.5.

You can see an example of this transformation in Figure 5.6. We note that after removing these arcs, we may get AND nodes. However, these AND nodes will have empty incoming neighborhood and therefore they are not part of the problem.

Example 5.2. Let us consider an OR function F with interaction digraph as shown in Figure 5.6 a). If we consider $V_{11} = \{1, 2\}$, $V_{00} = \{3, 4\}$, $V_{01} = \{5, 6\}$ and $V_{10} = \{7, 8\}$, then the result of the transformation is shown in Figure 5.6 b).

Now, for the existence of such s, Lemma 5.2 give us observe two immediate necessary conditions that depend only on x and y, and not on the structure of G:

- $\forall v \in V_{10},\ \forall u \in N_G^-(v)\colon x_u = 0 \ \vee\ y_u = 0$
- $\forall v \in V_{01},\ \exists u \in N_G^-(v)\colon x_u = 1 \ \vee\ y_u = 1$

We call above conditions as *compatible neighbors property*.

Since constructing an update schedule that correspond to a certain update digraph can be done in polynomial time (Aracena et al., 2011), we focus on solving the problem by labeling only a sufficient amount of arcs.

According to Lemma 5.2, all arcs incoming to a node in V_{10}, are uniquely determined, i.e., $\forall v \in V_{10},\ \forall u \in N_G^-(v)$:

- $u \in V_{01} \implies \mathrm{lab}(u, v) = \oplus$.

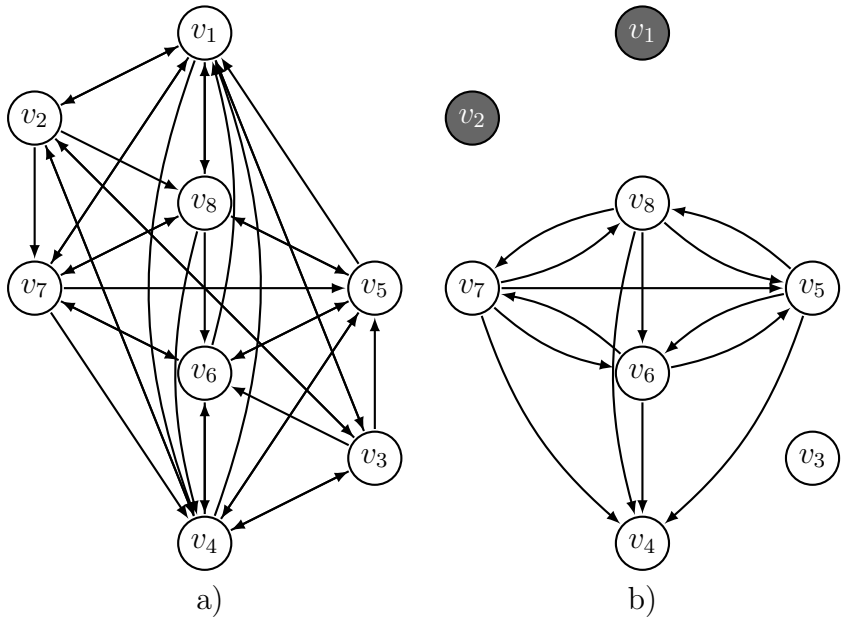


Figure 5.6: Example of the transformation defined in Lemma 5.3 and Remark 5.3.

- $u \in V_{10} \implies \text{lab}(u, v) = \ominus$.

Lemma 5.6. *If $G[V_{10}]$ has cycles, then OR FT has no solution.*

Proof. If $G[V_{10}]$ has a cycle C, then by Lemma 5.2 we have that:

$$\forall a \in A(C)\colon \text{lab}(a) = \ominus$$

This give us a forbidden cycle in G^R_{lab} which tell us that G_{lab} is not an update digraph. Therefore OR FT has no solution. □

Now, we have to analyze the incoming arcs of the nodes in V_{01} and we would like to have something similar to the nodes in V_{10}. Since the OR function tell us that for this nodes, only one incoming one is necessary to determine its value, we need to define the label of only one incoming arc. However, which one we chose? It is in this choice were it lies the complexity of this problem.
Nevertheless, Lemma 5.2 tell us that for $v \in V_{01}$ and $u \in N^-_G(v)$:

- If we label $u \in V_{01}$ then $\text{lab}(u, v)$ must be $\ominus$.
- If we label $u \in V_{10}$ then $\text{lab}(u, v)$ must be $\oplus$.

This conditions are summarized at Figure 5.7.

To study the labeling condition of the nodes in V_{01}, we need the following definitions.

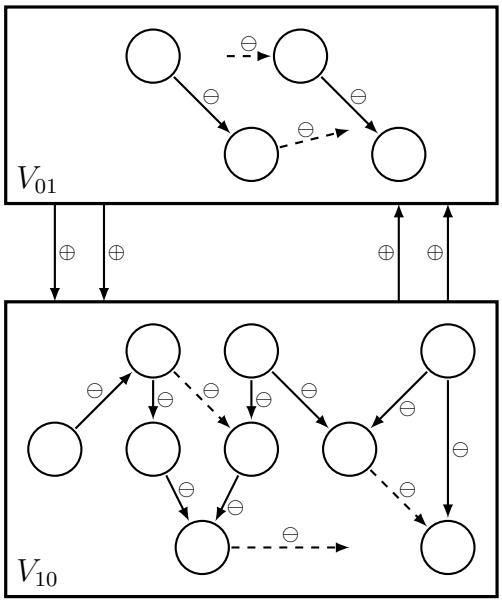


Figure 5.7: Necessary labels for the existence of solution for OR FT.

Definition 5.2. We define the following sets:

$$L^+ = \{v \in V_{01} \colon N_G^+(v) \cap V_{10} \neq \emptyset\}$$
$$L^- = \{v \in V_{01} \colon N_G^-(v) \cap V_{10} \neq \emptyset\}$$
$$O = \{v \in V_{01} \colon N_G^-(v) \cap V_{10} = \emptyset\}$$

Remark 5.4. We note that for OR FT to have a solution it is necessary that:

- $\forall v \in L^+,\ \forall u \in N_G^+(v) \cap V_{10} \colon\ \mathrm{lab}(u,v) = \oplus$
- For each node in $\{v \in L^+ \colon N_G^-(v) \subseteq V_{10}\}$, at least one incoming arc must be labeled $\oplus$.

Definition 5.3. Let $G^P = (V^P, E^P)$ the poset graph of $G[V_{01}]$ and $\{S_t^0\}_{t=1}^q \subseteq V^P$ its source nodes.

Remark 5.5. If $V_{11} \neq \emptyset$, then by Lemma 5.3, every node in $N_{11}^+ \cap V_{01}$ is a source node of $\hat{G}^P$, that we will call *constant induced source nodes*, and denote $\{C_t^0\}_{t=1}^{q_1}$.
The *non constant induced source nodes* will be denoted $\{N_t^0\}_{t=1}^{q_2}$.

Now we show another necessary condition for OR FT to have a solution.

Lemma 5.7. *If there exists a non trivial source node of G^P contained in O, then OR FT does not have a solution.*

Proof. Let be $t \in \{1, \ldots, q\}$ such that $S_t^0 \subseteq O$. By Lemma 5.2, and definition of O, we must $\ominus$-label at least one incoming arc of every node of S_t^0. Clearly in this case we have a forbidden cycle in G_{lab}^R and therefore OR FT does not have a solution. □

Therefore for OR FT to have a solution, every source node of G^P must have at least one node in L^-.

Remark 5.6. We note that:

- If $V_{11} \neq \emptyset$, then the previous lemma must be applied to the non trivial and non constant induced source nodes of G^P.
- If $\left\{v \in L^- \colon N_G^-(v) \subseteq V_{10}\right\} \neq \emptyset$ then every node in this set will be a trivial non constant induced source node of G^P.
- Every node in the above set does not fulfill condition of Lemma 5.7.

Corollary 5.8. *If $L^- = \emptyset$, then OR FT does not have a solution.*

Above results can be summarized in the following theorem

Theorem 5.9. *Let be V a set of n nodes and $F = (f_v)_{v \in V}$ be an OR function with interaction digraph $G = (V, A)$, and $x, y \in \{0,1\}^n$ without constant nodes. If there exists an update schedule s such that $F^s(x) = y$, then the following conditions hold:*

i. V_{10} and V_{01} has the compatible neighbors property.

ii. $G[V_{10}]$ is acyclic.

iii. $L^- \neq \emptyset$.

iv. Every non trivial source node of G^P have at least one node in L^-.

Now we show sufficient conditions for OR FT to have a solution.

Theorem 5.10. *If necessary conditions of Theorem 5.9 hold and*

i. There exists $A_0 = \{(u_t, v_t)\}_{t=1}^q \subseteq A$ such that

$$\forall t \in \{1, \ldots, q\} : u_t \in V_{10} \wedge v_t \in S_t^0$$

ii. The partially labeled digraph G_{lab} that comes from Lemma 5.2 and Remark 5.4, including the correct labeling of A_0, is an update digraph.

Then, there exists an update schedule s such that $F^s(x) = y$.

Proof. We consider G_{lab} the update digraph that comes from Lemma 5.2 and Remark 5.4, including the correct labeling of A_0. First we note that:

$$\forall t \in \{1, \ldots, q\} : \text{lab}\,(u_t, v_t) = \oplus$$

Now, we do the following:

- Starting from v_1, we build a spanning tree of the nodes in $R_G^+(v_1) \cap V_{01}$ and we add the nodes to a new defied set S_T.
- For each $t \in \{2, \ldots, q\}$, starting from v_t, we build a spanning tree of the nodes in $\left(R_G^+(v_t) \cap V_{01}\right) \setminus S_T$ and we add the nodes to S_T.
- We label every arc between nodes in S_T as $\ominus$.

Since,

1. $\forall t \in \{1, \ldots, q\} : \text{lab}\,(u_t, v_t) = \oplus \wedge \text{lab}\,(v_t, w_t) = \ominus$, where w_t is the corresponding node in S_T,
2. $\forall u \in L^+ \cap S_T,\ \forall v \in N_G^+(u) \cap V_{10} \colon \text{lab}\,(u, v) = \oplus \quad \wedge \quad \text{lab}\,(w, u) = \ominus$, where w is the corresponding node in S_T,

we have that this new labeled digraph, G_{lab}, is an update digraph. Since all necessary dynamical conditions are met, we have that every update schedule s such that

$$\forall a \in Sup\,(G_{\text{lab}}) : \text{lab}_s\,(a) = \text{lab}\,(a)$$

is such that $F^s(x) = y$, and can be found in polynomial time. □

Remark 5.7. If $V_{11} \neq \emptyset$, then $A_0 = \{(u_t, v_t)\}_{t=1}^{q_2}$ where:

$$\forall t \in \{1, \ldots, q_2\} : u_t \in N_t^0$$

Next example shows that condition i from Theorem 5.10 cannot be checked sequentially.

Example 5.3. Let us consider an OR function with part of its interaction digraph as shown in Figure 5.8. There, gray nodes represent nodes in V_{10} and red nodes represent nodes in $\{N_t^0\}_{t=1}^{q_2}$ (see Definition 5.3 and Remark 5.5).

1. If we first label arc $(4, i)$ as $\oplus$, then necessarily arcs $(7, f_1)$ and $(9, f_2)$ must be labeled $\ominus$, because otherwise G_{lab} would not be an update digraph. Additionally, arcs $(6, f_1)$ and $(13, f_2)$ must be labeled $\oplus$ to fulfill dynamical conditions.
2. If we now label arc $(1, j)$ as $\oplus$, then necessarily incoming arcs to node k_1 must be labeled $\ominus$ so G_{lab} remains an update digraph. Therefore, OR FT does not get a solution in this way since there is no local solution for node k_2. We obtain the same conclusion if we first label arc $(2, j)$ as $\oplus$.

Thus, there is no solution if we proceed in this way. However, we might find a solution if we start labeling arc $(3, i)$.

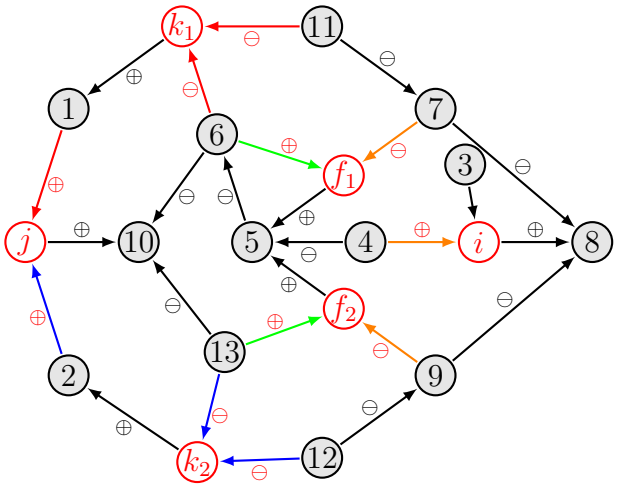


Figure 5.8: Example of an OR network such that condition i from Theorem 5.10 cannot be checked sequentially.

Algorithm

Summarizing the previous results, we obtain the following algorithm, for a given digraph $G = (V, A)$, $x, y \in \{0,1\}^{|V|}$.

1. We apply the transformation to deal with constant nodes according to the established in Lemma 5.3 and Remark 5.3.
2. If there is a cycle between nodes in V_{10}, then there is no solution. If there is not, we go to the next step.
3. We label the incoming arcs to nodes in V_{10} according to the established in Lemma 5.2.
4. If the resulting partial labeled digraph it is an update digraph, then we go to the next step, if it is not, then there is no solution.
5. We check for the existence of the set A_0 from Theorem 5.10.
6. OR FT has a solution if and only if such set exists.

Only step 4 is non polynomial. We note that this algorithm can be extended to obtain an update schedule if it exists. The complete algorithm will be presented in Appendix A.

5.1.2. Polynomial cases

In this section we are going to give two cases, depending of the structure of the interaction digraph, in which FT is polynomial.

Proposition 5.11. *AND-OR FT is polynomial if* $\Delta^- \left(G^F\right) \leq 1$.

Proof. Let $\langle F, x, y\rangle$ be an instance of AND-OR FT. By Lemma 5.3, we can suppose $V_c = \emptyset$. Since $\Delta^-\left(G^F\right) \leq 1$, the necessary conditions established in Lemma 5.2 become sufficient. Now, it is easy to see that these conditions can be checked in polynomial time. □

Remark 5.8. We note that in this class of networks, given s_1 and s_2 two different updates schedules, we have that: $F^{s_1} = F^{s_2} \iff G^F_{s_1} = G^F_{s_2} \iff LC(F, s_1) = LC(F, s_2)$ (Aracena et al., 2009, 2013b).

Proposition 5.12. *SYMMETRIC OR FT is polynomial.*

Proof. Let $\langle F, x, y\rangle$ be an instance of OR FT, with symmetric $G^F = (V, A)$. By Lemma 5.3, we can suppose $V_c = \emptyset$. By the results of the previous section (summarized in the given algorithm), we only need to prove that the existence of the set A_0 from Theorem 5.10 can be decided in polynomial time.

We note that for the existence of solution, by Lemma 5.6, it is necessary that V_{01} be disconnected, i.e., $\forall v \in V_{01}\colon N^-_G(v) \cap V_{01} = \emptyset$. Besides, by Lemma 5.7, it is necessary that every source node of the poset graph of V_{01} has an element that has an incoming arc from a node in V_{10}. Since V_{10} is disconnected, choosing any set A_0 fulfilling its definition will give us an update digraph and therefore SYMMETRIC OR FT has a solution.

In this way, the existence or non existence of the update schedule can be decided in polynomial time. □

5.2. Feasible Limit Cycle Problem

In this section we study the complexity of determining the existence of an update schedule such that a given sequence of state vectors is a limit cycle for a given global activation function. That is, we are going to consider the problem

Feasible Limit Cycle Problem **(FLC):** Given a set V of n elements and $F = (f_v)_{v\in V} : \{0,1\}^n \to \{0,1\}^n$ and a sequence $\mathcal{C} = \left[x^k\right]_{k=0}^{p}$ such that $x^k \in \{0,1\}^n$, x^k are pairwise distinct and $x^p \equiv x^0$. Does there exist an update schedule s such that $\mathcal{C} \in LC\left(F, s\right)$?

MON FLC and **OR FLC** problems are the corresponding FLC problems when F is a monotonic and an OR function, respectively. We note that this problem gain importance when we consider several kind of update schedules because if it is restricted to the parallel update schedule is trivially polynomial.

Now, to prove the next result, we are going to use an special case of the general satisfiability problem (SAT), which is defined as follows:

NOT-ALL-EQUAL SATISFIABILITY **(NAESAT):** Given ϕ a ncf in variables $w_1, \ldots, w_n$. Does there exist w such that $\phi(w) = 1$ and there is no clause in ϕ all literals of which are set to 1?

This problem is known to be NP-Complete (Schaefer, 1978). Observe that NAESAT is equivalent to: given a ncf ϕ, does there exists w such that $\phi(w) = \phi(\bar{w}) = 1$?

First, we prove that the general case is NP-Complete.

Theorem 5.13. *FLC is NP-Complete.*

Proof. It is clear that FLC is NP. To prove NP-Hardness we show that NAESAT $\leq_p$ FLC.

Given a 3-ncf ϕ in variables $w_1, \ldots, w_n$, we consider $V = \{v_1, \ldots, v_n, v_\phi\}$, $x^0 = (\vec{0}, 1)$, $x^1 = (\vec{1}, 1)$ and $\mathcal{C} = [x^0, x^1, x^0]$, where $\vec{0} = (0, \ldots, 0)$, $\vec{1} = (1, \ldots, 1) \in \{0,1\}^n$ and $F = (f_v)_{v \in V} \colon \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ as follows:

$$\begin{aligned} \forall i \in \{1, \ldots, n\} \colon f_{v_i}(x) &= \neg x_{v_i} \\ f_{v_\phi}(x) &= \phi(x_{v_i} : i \in \{1, \ldots, n\}) \end{aligned}$$

See G^F in Figure 5.9.

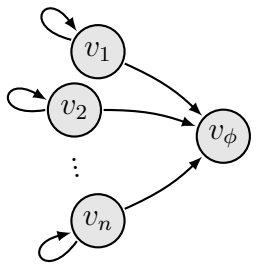


Figure 5.9: Interaction digraph of the transformation defined in Theorem 5.13.

($\Longrightarrow$) If there exists w such that $\phi(w) = \phi(\bar{w}) = 1$, then by defining $s = \{v_i \colon w_i = 1\}\{v_\phi\}\{v_i \colon w_i = 0\}$, it is clear that $\mathcal{C} \in LC(F, s)$.

($\Longleftarrow$) Let us suppose that there exists an update schedule s such that $\mathcal{C} \in LC(F, s)$. Then we define $w \in \{0,1\}^n$ such that $w_i = 1 \iff s(v_i) < s(v_\phi)$. It is easy to check that $\phi(w) = \phi(\bar{w}) = 1$. □

Next, we show that the ideas of the previous proof can be extended to the monotonic case.

Theorem 5.14. *MON FLC is NP-Complete.*

Proof. As in the general case, this problem is NP. To prove NP-Hardness, we show that SAT $\leq_p$ MON FLC. Given a ncf ϕ in variables $w_1, \ldots, w_n$, we build $F = (f_v)_{v \in V} \colon \{0,1\}^{2n+3} \to \{0,1\}^{2n+3}$, $\mathcal{C} = [x^0, x^1, x^2 \equiv x^0]$ where $x^0, x^1 \in \{0,1\}^{2n+3}$ and $V = \{v_1 \ldots, v_n, \bar{v}_1, \ldots, \bar{v}_n, z_1, z_2, z_3\}$, as follows:

$$
\begin{array}{ll}
\forall i \in \{1, \ldots, n\} & f_{v_i}(x) = x_{\bar{v}_i} \wedge (x_{z_1} \vee x_{z_3}) \\
\forall i \in \{1, \ldots, n\} & f_{\bar{v}_i}(x) = x_{v_i} \\
& f_{z_1}(x) = \bigwedge_{i=1}^{n} x_{v_i} \\
& f_{z_2}(x) = x_{z_1} \wedge \hat{\phi}\,(x_{v_i}, x_{\bar{v}_i} \colon i \in \{1 \ldots, n\}) \\
& f_{z_3}(x) = x_{z_2}
\end{array}
$$

where nodes v_i represent literals w_i; nodes $\bar{v}_i$ represent literals $\neg w_i$ and $\hat{\phi}$ is the monotonic version of ϕ, in variables $x_{v_1}, \ldots, x_{v_n}, x_{\bar{v}_1}, \ldots, x_{\bar{v}_n}$, that comes from ϕ replacing literals w_i by x_{v_i} and $\neg w_i$ by $x_{\bar{v}_i}$ (see Figure 5.10). Finally, we define $x^1 = \overline{x^0}$ and:

$$
x^0_u = \begin{cases} 1 & \text{if } u \in \{v_1, \ldots, v_n\} \\ 0 & \text{if } u \in \{\bar{v}_1, \ldots, \bar{v}_n, z_1, z_2, z_3\} \end{cases}
$$

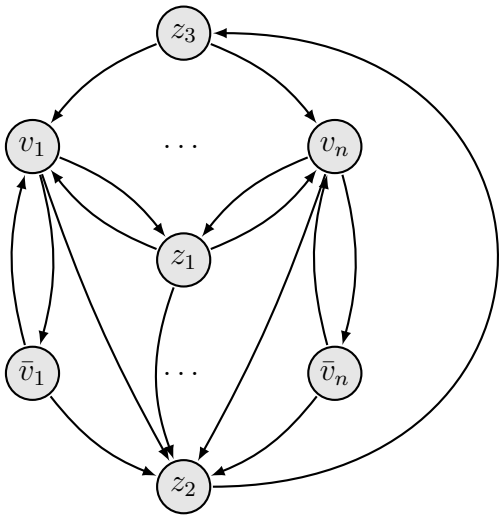


Figure 5.10: Interaction digraph of the transformation defined in Theorem 5.14.

The definition of F is similar than in Theorem 5.13, but monotonically. In order to achieve the monotony property, we add the $\bar{v}_i$ nodes that allow us to use $\hat{\phi}$ instead of ϕ, and the role of v_ϕ back there, that is to allow cycling, is monotonically done here by nodes z_1, z_2 and z_3.

($\Longrightarrow$) Let w be such that $\phi(w) = 1$, then if we consider the update schedule:

$$
s = \{z_1\}\{v_i, \bar{v}_i \colon w_i = 0\}\{z_2\}\{v_i, \bar{v}_i \colon w_i = 1\}\{z_3\}
$$

$v \in V$	$v_i\colon w_i = 1$	$v_i\colon w_i = 0$	$\bar{v}_i\colon w_i = 1$	$\bar{v}_i\colon w_i = 0$	z_1	z_2	z_3
$s(v)$	4	2	4	2	1	3	5
x^0_v	1	1	0	0	0	0	0
$x^1_v = F^s(x^0)_v$	0	0	1	1	1	1	1
$x^0_v = F^s(x^1)_v$	1	1	0	0	0	0	0

Table 5.4: Transition table of the states defined in Theorem 5.14.

From Table 5.4, it is clear that $\forall k \in \{0,1\} : F^s(x^k) = x^{k+1}$ and $x^2 \equiv x^0$. Therefore, $\mathcal{C} \in LC(F,s)$.

($\Longleftarrow$) Let s be an update schedule such that $\forall k \in \{0,1\} : F^s(x^k) = x^{k+1}$ and let x^s be the global state just before node z_1 get updated. Since $1 = x^1_{z_2} = f^s_{z_1}(x^0) = f_{z_1}(x^s)$, we have that $\hat{\phi}(x^s_{v_i}, x^s_{\bar{v}_i} : i \in \{1,\dots,n\}) = 1$. On another hand, we note that $\forall i \in \{1,\dots,n\}$:

1) $x^0_{v_i} = 1, x^1_{v_i} = 0, x^0_{\bar{v}_i} = 0, x^1_{\bar{v}_i} = 1 \implies s(\bar{v}_i) \leq s(v_i)$.

2) $x^1_{v_i} = 0, x^0_{v_i} = 1, x^1_{\bar{v}_i} = 1, x^0_{\bar{v}_i} = 0 \implies s(v_i) \leq s(\bar{v}_i)$.

3) Since v_i and $\bar{v}_i$ are connected by a cycle of length 2, necessarily

$$(s(v_i) \geq s(z_1) \wedge s(\bar{v}_i) \geq s(z_1)) \veebar (s(v_i) < s(z_1) \wedge s(\bar{v}_i) < s(z_1))$$

Thus, 1) and 2) imply that $\forall i \in \{1,\dots,n\} : s(v_i) = s(\bar{v}_i)$ and then $\forall i \in \{1,\dots,n\}, \forall k \in \{0,1\} : x^k_{\bar{v}_i} = \neg x^k_{v_i}$. From this and 3), we have that $\forall i \in \{1,\dots,n\}, \forall k \in \{0,1\} : x^s_{\bar{v}_i} = \neg x^s_{v_i}$.

Therefore, $\phi(\hat{x}) = 1$, with $\hat{x} = \left(x^s_{v_i}\right)_{i=1}^n$. □

To prove the OR case, we need a completely different approach, since above ideas are not sufficient when we are restricted to OR functions. First we prove the SAT variation we are going to use. We define:

$\mathbf{SAT}_{01}$: Given ϕ a ncf such that $\phi(\vec{0}) = \phi(\vec{1}) = 1$. Does there exists $x \notin \left\{\vec{0}, \vec{1}\right\}$ such that $\phi(x) = 1$?

Lemma 5.15. *SAT_{01} is NP-Complete.*

Proof. It is clear that SAT_{01} is NP. To prove NP-Hardness, we show that $\mathrm{SAT} \leq_p \mathrm{SAT}_{01}$.

Let ϕ be a ncf in variables $x_1 \dots, x_n$ and clauses $C_1, \dots, C_n$. We define $\hat{\phi}$ a ncf as follows:

$$\hat{\phi}(x) = \begin{cases} \bigwedge_{j=1}^{m} \bigwedge_{\substack{i,k=1 \\ i \neq k}}^{n} (C_j \vee \neg x_i \vee x_k) & \text{if } \phi(\vec{0}) = \phi(\vec{1}) = 0 \\ x_1 \vee \neg x_2 & \text{if } \phi(\vec{0}) = 1 \vee \phi(\vec{1}) = 1 \end{cases}$$

Clearly, $\hat{\phi}(\vec{0}) = \hat{\phi}(\vec{1}) = 1$.

($\Longrightarrow$) Let $x \in \{0,1\}^n$ be such that $\phi(x) = 1$. Hence,

- If $x \in \left\{\vec{0}, \vec{1}\right\}$, then $\hat{\phi}(x) = x_1 \vee \neg x_2$ and considering $\hat{x} = (1,0)$ we have that $\hat{\phi}(\hat{x}) = 1$.
- If $x \notin \left\{\vec{0}, \vec{1}\right\}$, then considering $\hat{x} = x$ we have that $\hat{\phi}(\hat{x}) = 1$.

($\Longleftarrow$) Let $\hat{x}$ be such that $\hat{\phi}(\hat{x}) = 1$ and let us suppose that $\forall x\colon \phi(x) = 0$, then there exist $j \in \{1, \ldots, m\}$ such that $C_j(\hat{x}) = 0$.

Thus, $\forall i \neq k \in \{1, \ldots, n\} : C_j(\hat{x}) \vee \neg\hat{x}_i \vee \hat{x}_k = \neg\hat{x}_i \vee \hat{x}_k = 1$.

Now, if there exists $k \in \{1, \ldots, n\}$ such that $\hat{x}_k = 0$, then for each $i \neq k \in \{1, \ldots, n\}$ we have that $\hat{x}_i = 0$ and therefore, $\hat{x} = \vec{0}$. Otherwise, $\hat{x} = \vec{1}$.

Analogously, if there exists $i \in \{1, \ldots, n\}$ such that $\hat{x}_i = 1$, then for each $k \neq i \in \{1, \ldots, n\}$ we have that $\hat{x}_k = 1$ and therefore, $\hat{x} = \vec{1}$. Otherwise, $\hat{x} = \vec{0}$.

Thus, $\hat{\phi}$ is only satisfiable by $\vec{0}$ and $\vec{1}$. □

Now we prove that OR FLC is NP-Complete.

Theorem 5.16. *OR FLC is NP-Complete.*

Proof. We prove that $\mathrm{SAT}_{01} \leq_p$ OR FLC.

Let ϕ be ncf in variables $w_1, \ldots, w_n$ with clauses $C_0, \ldots, C_{m-1}$ such that $\phi(\vec{0}) = \phi(\vec{1}) = 1$.

We define an OR function F and a limit cycle $\mathcal{C}$ such that each variable w_i is represented by a node $v_i \in V(G^F)$ and whose value is defined according to the relative order of schedule between node v_i and a given node v_ϕ. Besides, each clause of ϕ is associated to a transitions in the limit cycle $\mathcal{C}$.

More precisely, we define $F = (f_v)_{v \in V}\colon \{0,1\}^{3m+n+4} \to \{0,1\}^{3m+n+4}$ an OR function by its interaction digraph defined in Table 5.5 (see an example in Figure 5.11).

$\boldsymbol{v \in V}$	$\boldsymbol{N^-_{G^F}(v)}$
$v_i,\ i \in \{1, \ldots, n\}$	$\{z_0\} \cup \{C^1_j\colon w_i \in C_j\} \cup \{C^2_j\colon \neg w_i \in C_j\}$
$C^k_j,\ j \in \{0, \ldots, m-1\},\ k \in \{1,2\}$	$\{C^{k-1}_j\}$
$C^0_j,\ j \in \{0, \ldots, m-1\}$	$\{C^2_{j-1 \bmod m}\}$
$z_k,\ k \in \{0,1,2\}$	$\{z_{k-1 \bmod 3}\}$
v_ϕ	$\{v_1, \ldots, v_n\}$

Table 5.5: Definition of G^F defined in Theorem 5.16.

And, we define $\mathcal{C} = [x^{0,0}, x^{1,0}, x^{2,0}, x^{0,1}, \dots] = \left[x^{k,j}\right]_{k\in\mathbb{Z}_3, j\in\mathbb{Z}_m}$ of length $3m$ as:

$$
\begin{aligned}
x^{k,j}_{v_i} &= \begin{cases} 1 & \text{if } k = 0 \vee (k = 1 \wedge w_i \in C_j) \vee (k = 2 \wedge \neg w_i \in C_j) \\ 0 & \text{otherwise} \end{cases} \\
x^{k,j}_{C^{k'}_{j'}} &= \begin{cases} 1 & \text{if } j = j' \wedge k = k' \\ 0 & \text{otherwise} \end{cases} \\
x^{k,j}_{z_{k'}} &= \begin{cases} 1 & \text{if } k = k' \\ 0 & \text{otherwise} \end{cases} \\
x^{k,j}_{v_\phi} &= 1.
\end{aligned}
$$

See an example in Table 5.6.

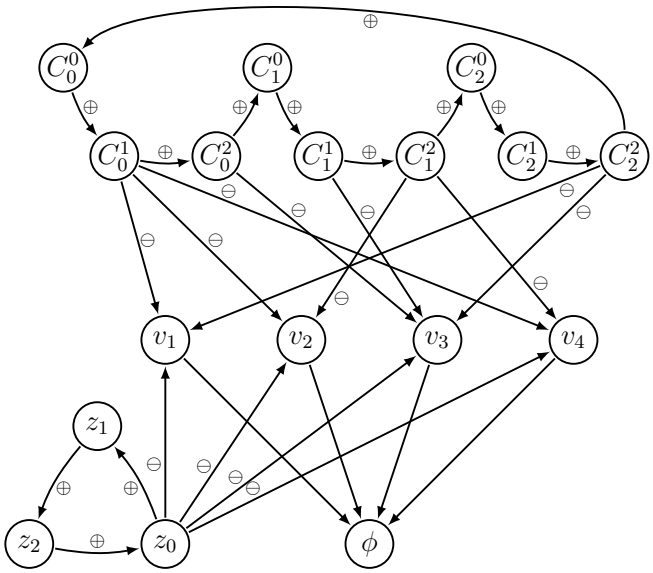


Figure 5.11: Example of G^F for $\phi\,(w) = (w_1 \vee w_2 \vee \neg w_3 \vee w_4) \wedge (\neg w_2 \vee w_3 \vee \neg w_4) \wedge (\neg w_1 \vee \neg w_3)$ according to the transformation defined in Theorem 5.16.

In this way, each clause C_j in ϕ is represented by the vectors $x^{1,j}$ and $x^{2,j}$ such that: $x^{1,j}_{v_i} = 1$ and $x^{2,j}_{v_i} = 0$ if the literal w_i is in C_j; $x^{1,j}_{v_i} = 0$ and $x^{2,j}_{v_i} = 1$ if $\neg w_i$ is in C_j, and $x^{1,j}_{v_i} = 0$ and $x^{2,j}_{v_i} = 0$ otherwise. Hence, for all $j \in \{0, \dots, m-1\}$, $x^{2,j}_{v_\phi} = 1$ if and only if there exists $i \in \{1, \dots, n\}$ such that either $x^{1,j}_{v_i} = 1$, $x^{2,j}_{v_i} = 0$ and $s(v_i) \geq s(v_\phi)$ or $x^{1,j}_{v_i} = 0$, $x^{2,j}_{v_i} = 1$ and $s(v_i) < s(v_\phi)$. Therefore, we obtain an equivalence between the relative order of nodes v_i and v_ϕ, and the value of the variable w_i as follows:

$$
s\,(v_i) \geq s\,(v_\phi) \quad \Longleftrightarrow \quad w_i = 1 \tag{5.9}
$$

In this way, variable v_ϕ remains frozen with value equal to one if and only if all clauses are satisfiable.

	C_0^0	C_0^1	C_0^2	C_1^0	C_1^1	C_1^2	C_2^0	C_2^1	C_2^2	z_0	z_1	z_2	v_1	v_2	v_3	v_4	v_ϕ
$x^{0,0}$	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1
$x^{1,0}$	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1
$x^{2,0}$	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1
$x^{0,1}$	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	1	1
$x^{1,1}$	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1
$x^{2,1}$	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	1
$x^{0,2}$	0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1
$x^{1,2}$	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1
$x^{2,2}$	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	1
$x^{0,3}$	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1

Table 5.6: Example of $\mathcal{C}$ for $\phi(w) = (w_1 \vee w_2 \vee \neg w_3 \vee w_4) \wedge (\neg w_2 \vee w_3 \vee \neg w_4) \wedge (\neg w_1 \vee \neg w_3)$ according to the transformation defined in Theorem 5.16.

Besides, we note that, by Lemma 5.2 applied to each transition in C, we have that for every update schedule s such that $\mathcal{C} \in LC(F, s)$:

$$\forall k \in \{1,2\},\ \forall j \in \{0,\dots,m-1\}: \quad s\left(C_j^{k-1}\right) \geq s\left(C_j^k\right) \tag{5.10}$$

$$\forall j \in \{0,\dots,m-1\}: \quad s\left(C_{j-1 \bmod m}^2\right) \geq s\left(C_j^0\right) \tag{5.11}$$

$$\forall k \in \{0,1,2\}: \quad s\left(z_{k-1 \bmod 3}\right) \geq s\left(z_k\right) \tag{5.12}$$

$$\forall i \in \{1,\dots,n\}: \quad s\left(z_0\right) < s\left(v_i\right) \tag{5.13}$$

$$\forall i \in \{1,\dots,n\},\ \forall k \in \{1,2\},\ \forall j \in \left\{t\colon C_t^k \in N_{G^F}^-(i)\right\}: \quad s\left(C_j^k\right) < s\left(v_i\right) \tag{5.14}$$

We note that conditions (5.10)–(5.14) define uniquely the labels of the arcs involved as shown in Figure 5.11.

We show now the details of the proof:

($\Longrightarrow$) Let $w \notin \left\{\vec{0}, \vec{1}\right\}$ be such that $\phi(w) = 1$, then we define the update schedule s as

$$s = B_1\, B_2\, B_3\, B_4\, B_5,$$

where

$$\begin{aligned}
B_1 &= \left\{C_j^k : j \in \{0,\dots,m-1\},\ k \in \{0,1,2\}\right\},\\
B_2 &= \left\{z_k : k \in \{0,1,2\}\right\},\\
B_3 &= \left\{v_i : w_i = 0\right\},\\
B_4 &= \left\{v_\phi\right\},\\
B_5 &= \left\{v_i : w_i = 1\right\}.
\end{aligned}$$

It is easy to see that s satisfies conditions (5.9)–(5.14) and for all $v \in V \setminus \{v_\phi\}$,

$$x_v^{k,j} = \begin{cases} f_v^s(x^{2,j-1 \bmod m}) & \text{if } k = 0\\ f_v^s(x^{k-1,j}) & \text{otherwise} \end{cases}$$

Also by condition (5.9) and by definition of F and $\mathcal{C}$ we have that:

$$f^s_{v_\phi}\left(x^{k,j}\right) = \begin{cases} \bigvee_{\{i:s(v_i)<(v_\phi)\}} x^{0,j}_{v_i} \vee \bigvee_{\{i:s(v_i)\geq(v_\phi)\}} x^{2,j-1 \bmod m}_{v_i} & \text{if } k=0 \\ \bigvee_{\{i:s(v_i)<(v_\phi)\}} x^{1,j}_{v_i} \vee \bigvee_{\{i:s(v_i)\geq(v_\phi)\}} x^{0,j}_{v_i} & \text{if } k=1 \\ \bigvee_{\{i:s(v_i)<(v_\phi)\}} x^{2,j}_{v_i} \vee \bigvee_{\{i:s(v_i)\geq(v_\phi)\}} x^{1,j}_{v_i} & \text{if } k=2 \end{cases}$$

Moreover,

$$\begin{aligned} w \neq \vec{1} &\Longrightarrow \bigvee_{\{i:s(v_i)<(v_\phi)\}} x^{0,j}_{v_i} = 1, \text{ and} \\ w \neq \vec{0} &\Longrightarrow \bigvee_{\{i:s(v_i)\geq(v_\phi)\}} x^{0,j}_{v_i} = 1, \text{ and} \\ \phi(w)=1 &\Longrightarrow \bigvee_{\{i:s(v_i)<(v_\phi)\}} x^{2,j}_{v_i} \vee \bigvee_{\{i:s(v_i)\geq(v_\phi)\}} x^{1,j}_{v_i} = 1. \end{aligned}$$

Hence $\forall j \in \{0,\ldots,m-1\}\,,\ \forall k \in \{0,1,2\} : f^s_{v_\phi}\left(x^{k,j}\right) = 1$. Therefore, $\mathcal{C} \in LC\,(F,s)$.

($\Longleftarrow$) Let s be an update schedule such that $\mathcal{C} \in LC(F,s)$. Then, by definition of $\mathcal{C}$, conditions (5.10)–(5.14) are satisfied and we define w by condition (5.9). Clearly, $\phi(w)=1$.

Since $\phi(\vec{0}) = \phi(\vec{1}) = 1$, then following configurations do not appear in $\mathcal{C}$:

$x^{k,j}$	v_1	$\cdots$	v_n	v_ϕ
$x^{0,j}$	1	$\vec{1}$	1	1
$x^{1,j}$	1	$\vec{1}$	1	1
$x^{2,j}$	0	$\vec{0}$	0	1
$x^{0,j+1}$	1	$\vec{1}$	1	1

$x^{k,j'}$	v_1	$\cdots$	v_n	v_ϕ
$x^{0,j'}$	1	$\vec{1}$	1	1
$x^{1,j'}$	0	$\vec{0}$	0	1
$x^{2,j'}$	1	$\vec{1}$	1	1
$x^{0,j'+1}$	1	$\vec{1}$	1	1

Thus, $\exists i,j \in \{1,\ldots,n\} : s\,(v_i) \geq s\,(v_\phi) \wedge s\,(v_j) < s\,(v_\phi)$ and therefore $w \notin \left\{\vec{0},\vec{1}\right\}$. □

Remark 5.9. Note that the proof of the previous theorem can be modified to prove that OR FLC is NP-Complete restricted to sequential update schedules. We just need to add two extra nodes in the digraph: a node a to the cycle with nodes C^k_j and a node b to the cycle with nodes z_k. In this way, the limit cycle $\mathcal{C}$ will be $x^{k,j}_b = x^{k,j}_{z_0}$ and $x^{k,j}_a = x^{k,j}_{C^0_0}$ in the new nodes. From Lemma 5.2 we deduce conditions about the update schedule compatible with a sequential update.

5.2.1. Polynomial cases

In this section, we are going to show some cases in which FLC is polynomial.

Given the results of Section 4.1, we can prove that SYMMETRIC OR FLC is also polynomial.

Proposition 5.17. *SYMMETRIC OR FLC is polynomial.*

Proof. Let F be an OR function with symmetric G^F and a sequence of different state vectors $\mathcal{C} = \left[x^k\right]_{k=0}^{p}, x^k \in \{0,1\}^n, x^p = x^0$. According to Proposition 4.4, to determine the solution to the problem is necessary and sufficient to check $p = 2$, $F(x^0) = x^1$ and $F(x^1) = x^0$, which can be done in polynomial time. □

However, symmetrization by itself is not a sufficient condition as shown in the following result.

Proposition 5.18. *SYMMETRIC FLC is NP-Hard.*

Proof. To prove this result, we keep the limit cycle of Theorem 5.13 but we change the local activation functions by the following:

$$\begin{aligned} \forall i \in \{1,\ldots,n\}\colon f_{v_i}(x) &= \neg x_{v_i} \wedge x_{v_\phi} \\ f_{v_\phi}(x) &= \phi(x_{v_i} : i \in \{1,\ldots,n\}) \end{aligned}$$

In this way G^F is symmetric, and the proof is similar to Theorem 5.13. □

Now we consider FLC problem in AND-OR networks but only for limit cycles of length 2, that we call AND-OR 2-FLC problem. For simplicity, we are going to prove the OR case, since the extension to the AND-OR case is straightforward.

Proposition 5.19. *OR 2-FLC is polynomial.*

Proof. Let be F an AND-OR function and $\mathcal{C} = [x^0, x^1, x^2 \equiv x^0]$. The idea is to label G^F according to the dynamical conditions imposed by the limit cycle using Lemma 5.2, then if this digraph partially labeled is an update digraph, the Extension Theorem shown in (Aracena et al., 2011) ensure that there exists an update schedule that generates it. We note that we construct an update digraph instead of the update schedule, since in Aracena et al. (2009) was shown that two different update schedules that yields the same update digraph have the same dynamical behavior. Now we proceed to the details of the algorithm.

1. First, we note that considering that we want that $F^s(x^0) = x^1$ for some update schedule s, we can define, according to Definition 5.1, the sets $V_{10}, V_{01}, V_{00}, V_{11}, V_c$. It is clear that nodes in V_c are frozen in $\mathcal{C}$.

2. We decompose G^F into its strongly connected components $G^1, G^2, \dots G^m$.

3. For every node in V_{11}, we check that every reachable node from it belongs to V_{11}. By the dynamical behavior of OR functions, if there is a reachable node that does not belongs to V_{11}, then $\mathcal{C}$ can not be a limit cycle under any update schedule.

4. For every node in V_{00}, we check that every node that reach it node belongs to V_{00}. By the behavior of OR functions, if there is such a node that does not belongs to V_{00}, then $\mathcal{C}$ can not be a limit cycle under any update schedule.

   At this point, we have only frozen strongly connected components, namely $G^1_{00}, G^2_{00}, \dots G^{m_0}_{00}$ for those that are frozen at value 0 and $G^1_{11}, G^2_{11}, \dots G^{m_1}_{11}$ for those that are frozen at value 1, or components in which every node cycle in $\mathcal{C}$, namely $G^1_c, G^2_c, \dots G^{m_c}_c$. Besides, each component G^i_{11} send arcs to components of the same kind, as well as the components G^i_{00} only receive arcs from components of the same kind.

   Due to the behavior of OR functions, it is clear that the labels in the arcs between nodes in the frozen connected components can take any value to fulfill the dynamical conditions imposed by $\mathcal{C}$, as well as the labels in the arcs between components. Therefore, we just need to label the arcs in the components G^i_c.

5. For each node in the components G^i_c, we label their incoming arcs according to the conditions in the update schedule established in Lemma 5.2.

6. We check if the digraph partially labeled is an update digraph. Such test is polynomial as shown in Aracena et al. (2011). If the digraph partially labeled is an update digraph, the Extension Theorem ensure us that there is a label function lab defined in all arcs of G^F such that (G^F, lab) is an update digraph and such that it contains the partial labeled digraph as a sub-digraph. Therefore, there exists an update schedule s such that $\mathcal{C} \in LC(F, s)$. Besides, such an update schedule can be found in polynomial time (Aracena et al., 2011). □

Corollary 5.20. *AND-OR 2-FLC is polynomial.*

Proof. We just need to apply an analogous procedure as in the previous theorem, but considering the AOA decomposition of the interaction digraph. □

5.3. Other related problems

In this section, we study another related decision problems, defined as follows:

1. Predecessor Problem **(PP)**: Given $F : \{0,1\}^n \to \{0,1\}^n$ and $y \in \{0,1\}^n$. Does there exists $x \in \{0,1\}^n$ such that $F(y) = x$?

   It is known that this problem is NP-Complete (ref?).

2. Boolean Network Predecessor Problem **(BNP)**: Given $N = (F, s)$ a Boolean network and $y \in \{0,1\}^n$. Does there exists $x \in \{0,1\}^n$ such that $F^s(x) = y$?

3. Update Schedule Predecessor Problem **(USP)**: Given $F : \{0,1\}^n \to \{0,1\}^n$ and $y \in \{0,1\}^n$. Does there exist an update schedule s and $x \in \{0,1\}^n$ such that $F^s(x) = y$?

4. Feasible Multiple Transitions Problem **(FMT)**: Given $F : \{0,1\}^n \to \{0,1\}^n$, and two sets $\left\{x^k \colon k \in \{0,\ldots,p\}\right\} \subseteq \{0,1\}^n$ and $\left\{y^k \colon k \in \{0,\ldots,p\}\right\} \subseteq \{0,1\}^n$. Does there exist an update schedule s such that $\forall k \in \{0,\ldots,p\} : F^s(x^k) = y^k$?

5. Feasible Trajectory Problem **(FTy)**: Given $F : \{0,1\}^n \to \{0,1\}^n$ and a set $\left\{x^k \colon k \in \{0,\ldots,p\}\right\} \subseteq \{0,1\}^n$. Does there exist an update schedule s such that $\forall k \in \{0,\ldots,p-1\} : F^s(x^k) = x^{k+1}$?

Now we show all the complexity results.

Theorem 5.21. *BNP is NP-Complete.*

Proof. Clearly, this problem is NP. To show NP-Hardness, we show that PP $\leq_p$ BNP.

Let $F : \{0,1\}^n \to \{0,1\}^n$ be a Boolean function and $y \in \{0,1\}^n$, and let us consider $N = (F, s^p)$ where s^p is the parallel update schedule. Then, it is clearly true that:

$$\exists x \in \{0,1\}^n \colon F(x) = y \quad \Longleftrightarrow \quad \exists x \in \{0,1\}^n \colon F^{s^p}(x) = y$$

□

Theorem 5.22. *USP is NP-Complete.*

Proof. Clearly, this problem is NP. To show NP-Hardness, we show that SAT $\leq_p$ USP.

Given a ncf ϕ in variables $w_1, \ldots, w_n$, we consider $x = \vec{1} \in \{0,1\}^{n+1}$, V and $F = (f_v)_{v \in V} \colon \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ as follows:

$$\begin{aligned} \forall i \in \{1,\ldots,n\} \colon f_{v_i}(x) &= \neg x_{v_i} \\ f_{v_\phi}(x) &= \phi(x_{v_i} \colon i \in \{1,\ldots,n\}) \end{aligned}$$

($\Longrightarrow$) If there exists w such that $\phi(w) = 1$, then we define $y = \vec{0} \in \{0,1\}^{n+1}$ and $s = \{v_i \colon w_i = 1\}\,\{v_\phi\}\,\{v_i \colon w_i = 0\}$. Clearly $F^s(x) = y$.

($\Longleftarrow$) Let us suppose that there exists an update schedule s and $x \in \{0,1\}^n$ such that $F^s(x) = y$. If we consider $w = \left(x^s_{v_i}\right)_{i=1}^n$, where x^s is the global state just before node v_ϕ gets updated, then $\phi(w) = 1$.

□

Theorem 5.23. *OR USP is NP-Complete.*

Proof. It is clear that as in the general case, this problem is NP. To prove NP-Hardness, we show that OR FT $\leq_p$ OR USP.

Let be $F\colon \{0,1\}^n \to \{0,1\}^n$ an OR function with interaction digraph $G^F = (V, A)$ and $x, y \in \{0,1\}^n$ and let us define for each $q \in \{0,1\}$:

$$V_q = \{v \in V_{qq} \cup V_{\neg qq} \colon N^-_{G^F}(v) \subseteq V_c\}$$

We built $\hat{F}\colon \{0,1\}^{2n} \to \{0,1\}^{2n}$ by constructing $G^{\hat{F}} = \left(\hat{V}, \hat{A}\right)$, where $\hat{V}$ contains each node $v \in V$ and a copy of it, denoted $\hat{v}$, and $\hat{A}$ is defined as follows:

1.- $\hat{A} = A \setminus \{(v,u) \in A \colon v \in V_c,\ u \notin V_0 \cup V_1\}$.

   I.- $\forall q \in \{0,1\},\ \forall v \in V_q$, we choose a node $w_{v,q} \in N^-_{G^F}(v) \cap V_{qq}$, if exists. We note that w_q always exists if there is a solution.

   II.- $\hat{A} = \hat{A} \setminus \left\{(u,v) \in A \colon v \in V_q,\ u \in N^-_{G^F}(v) \setminus \{w_{v,q}\},\ q \in \{0,1\}\right\}$.

2.- $\forall v \in V \colon N^-_{G^{\hat{F}}}(\hat{v}) = \{v\}$.

Finally, we define $\hat{y} := (y, x)$. See an example of the transformation in Figure 5.12.

($\Longrightarrow$) If there exists update schedule $s = B_1\ B_2 \cdots B_m$ such that $F^s(x) = y$, then considering $\hat{s} = \left\{v \in \hat{V} \setminus V\right\} \{v \in B_1\} \{v \in B_2\} \cdots \{v \in B_m\}$ and $\hat{x} = (x, x) \in \{0,1\}^{2n}$, we have:

$$\begin{aligned}
&\forall v \in V\colon && \hat{f}^{\hat{s}}_{\hat{v}}(\hat{x}) = \hat{x}_v = x_v \\
&\forall v \in V \setminus (V_0 \cup V_1)\colon && \hat{f}^{\hat{s}}_{v}(\hat{x}) = f^s_v(x) = y_v \\
&\forall q \in \{0,1\},\ \forall v \in V_q\colon && \hat{f}^{\hat{s}}_{v}(\hat{x}) = \begin{cases} \hat{x}_{w_{v,q}} & \text{if } s(w_{v,q}) \geq s(v) \\ \hat{y}_{w_{v,q}} & \text{if } s(w_{v,q}) < s(v) \end{cases} \\
& && = q = y_v
\end{aligned}$$

Therefore, $\hat{F}^{\hat{s}}(\hat{x}) = \hat{y}$.

($\Longleftarrow$) Let us suppose that there exists an update schedule $\hat{s} = B_1\ B_2 \cdots B_m$ and $\hat{x} \in \{0,1\}^{2n}$ such that $\hat{F}^{\hat{s}}(\hat{x}) = \hat{y}$. Then, if we consider the update schedule $s = \{v \in B_1 \cap V\} \{v \in B_2 \cap V\} \cdots \{v \in B_m \cap V\}$, removing the empty blocks, and $z = (\hat{x}_{v_i})_{i=1}^n \in \{0,1\}^n$ we have that $F^s(z) = y$.

Now, we are going to show that z can be chosen as x. We note that $\forall v \in V$:

$$\hat{y}_{\hat{v}} = x_v = \begin{cases} y_v & \text{if } \hat{s}(v) < \hat{s}(\hat{v}) \\ \hat{x}_v & \text{if } \hat{s}(v) \geq \hat{s}(\hat{v}) \end{cases} \tag{5.15}$$

and therefore $\forall v \in V\colon \hat{s}(v) \geq \hat{s}(\hat{v}) \implies \hat{x}_v = x_v$.

On another hand, if $v \notin V_c$, then $x_v \neq y_v$ and by (5.15) necessarily $\hat{s}(v) \geq \hat{s}(\hat{v})$ and therefore $\hat{x}_v = x_v$.

Now, let be $v \in V_{qq}$ such that $\hat{s}(v) < \hat{s}(\hat{v})$ and $\hat{x}_v \neq x_v$, with $q \in \{0,1\}$, then

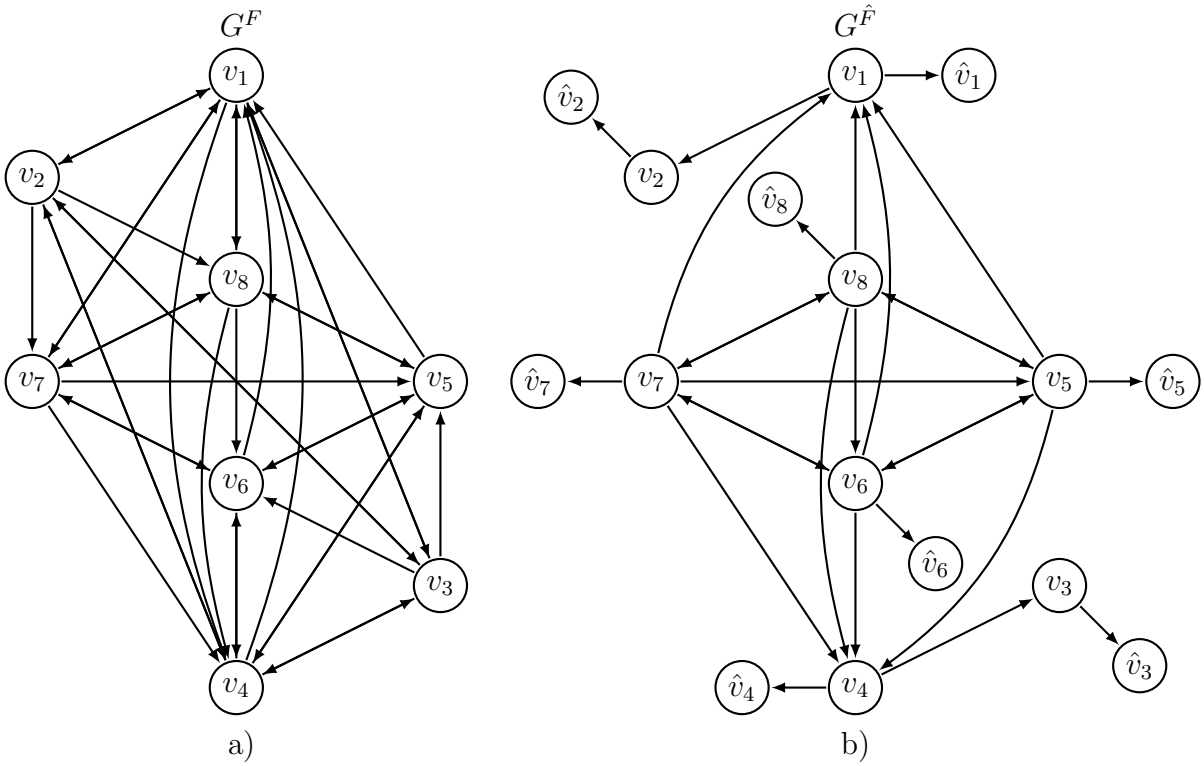


Figure 5.12: Example of the transformation defined in Theorem 5.23, as detailed in Example 5.4.

- If $N^+_{G^{\hat{F}}}(v) = \{\hat{v}\}$, then changing the value of $\hat{x}_v$ to $q = x_v$ does not produce any effect in the value of node v nor the rest of the network.
- If $\exists u \in N^+_{G^{\hat{F}}}(v) \setminus \{\hat{v}\}$, then $u \in V_q$, $\hat{s}(v) < \hat{s}(u)$ and $\hat{y}_u = \hat{y}_v = q$. Therefore, changing the value of $\hat{x}_v$ to $q = x_v$ do not produce any effect in the value of nodes v and u, nor the rest of the network.

Thus, $F^s(x) = y$. □

Example 5.4. Let us consider an OR function F with interaction digraph as shown in Figure 5.12 a). If we consider, $V_{11} = \{1, 2\}$, $V_{00} = \{3, 4\}$, $V_{01} = \{5, 6\}$ and $V_{10} = \{7, 8\}$, then the result of the transformation is shown in Figure 5.12 b), with $V_1 = \{v_2\}$, $V_0 = \{v_3\}$, $w_{v_2,1} = v_1$ and $w_{v_3,0} = v_4$.

Theorem 5.24. *The following problems are NP-Complete:*

i.- *FTy.*

ii.- *FMT.*

iii.- *OR FTy.*

iv.- *OR FMT.*

Proof.

I.- Clearly, this problem is NP. To show NP-Hardness, it is clear that FLC $\leq_p$ FTy through identity reduction. By Theorem 5.13 we have the result.

II.- Clearly, this problem is NP. To show NP-Hardness, it is clear that FTy $\leq_p$ FMT through identity reduction. By part i, we have the result.

III.- Straightforward from Theorem 5.16 and part i.

IV.- Straightforward from parts ii and iii. □

Chapter 6

Applications

In this section we are going to apply some of the resulta of the previous chapters to some Boolean networks used in the literature.

First, we study the mammalian cell cycle network (Fauré et al., 2006; Ruz et al., 2014) and next we study the fission yeast cell-cycle network (Davidich and Bornholdt, 2008; Goles et al., 2013). Here, using mainly the results of Chapter 3, we study the robustness of limit cycles when the update schedule is changed and we exhibit some update schedule classes, different of the parallel one, that generates the same attractors that the respective networks updated in parallel.

6.1. Analysis of the robustness of limit cycles of the mammalian cell cycle network

In this section we study the robustness of the mammalian cell cycle network first introduced in Fauré et al. (2006) and extensively studied in Ruz et al. (2014), where some theoretical results were given, but their approach is essentially by simulations, using all possible update schedules. Our aim here is to give some insights to the robustness of the networks when the update schedule is changed, without using any simulations. This network was also studied in Gómez (2009), but the objective was to find non sequential update schedules that do not shared limit cycles with the parallel one. Now, we are interested in to find non equivalent updated schedules that yield the same limit cycle that the parallel schedule.

We proceed now to describe the network. The network has 10 nodes, that for simplifying notation we denote it as shown in Table 6.1.

The local activation functions are given in Table 6.2 according to described in Ruz et al. (2014). The interaction digraph is given in Figure 6.1.

This network synchronously updated, denoted $N^p = (F, s^p)$ has only one fixed point and one limit cycle of length 7 as attractors, that are described in Table 6.3. The complete

CycD $\equiv v_1$	p27 $\equiv v_6$
Rb $\equiv v_2$	Cdc20 $\equiv v_7$
E2F $\equiv v_3$	Cdh1 $\equiv v_8$
CycE $\equiv v_4$	UbcH10 $\equiv v_9$
CycA $\equiv v_5$	CycB $\equiv v_{10}$

Table 6.1: Notation for the nodes of the mammalian cell cycle network.

$$
\begin{aligned}
f_{v_1}(x) &= x_{v_1} \\
f_{v_2}(x) &= (\neg x_{v_1} \wedge \neg x_{v_{10}}) \wedge ([\neg x_{v_4} \wedge \neg x_{v_5}] \vee x_{v_6}) \\
f_{v_3}(x) &= (\neg x_{v_2} \wedge \neg x_{v_5} \wedge \neg x_{v_{10}}) \vee (x_{v_6} \wedge \neg x_{v_2} \wedge \neg x_{v_{10}}) \\
f_{v_4}(x) &= x_{v_3} \wedge \neg x_{v_2} \\
f_{v_5}(x) &= (\neg x_{v_2} \wedge \neg x_{v_7} \wedge \neg (x_{v_8} \wedge x_{v_9})) \wedge (x_{v_3} \vee x_{v_5}) \\
f_{v_6}(x) &= (\neg x_{v_1} \wedge \neg x_{v_{10}}) \wedge ([\neg x_{v_4} \wedge \neg x_{v_5}] \vee [x_{v_6} \wedge \neg (x_{v_4} \wedge x_{v_5})]) \\
f_{v_7}(x) &= x_{v_{10}} \\
f_{v_8}(x) &= (\neg x_{v_5} \wedge \neg x_{v_{10}}) \vee x_{v_7} \vee (x_{v_6} \wedge \neg x_{v_{10}}) \\
f_{v_9}(x) &= \neg x_{v_8} \vee (x_{v_8} \wedge x_{v_9} \wedge [x_{v_7} \vee x_{v_5} \vee x_{v_{10}}]) \\
f_{v_{10}}(x) &= \neg x_{v_7} \wedge \neg x_{v_8}
\end{aligned}
$$

Table 6.2: Local activation functions of the mammalian cell cycle network.

dynamical behavior is shown in Figure 6.2

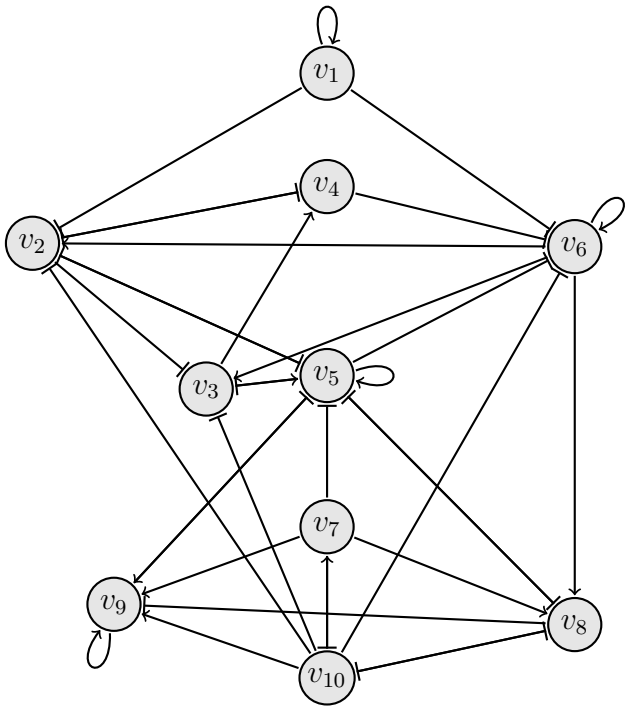


Figure 6.1: Interaction digraph of the mammalian cell cycle network.

$v \in V\left(G^F\right)$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
Fixed Point										
x_v^0	0	1	0	0	0	1	0	1	0	0

Limit Cycle										
x_v^0	1	0	1	0	0	0	0	1	1	0
x_v^1	1	0	1	1	0	0	0	1	0	0
x_v^2	1	0	1	1	1	0	0	1	0	0
x_v^3	1	0	0	1	1	0	0	0	0	0
x_v^4	1	0	0	0	1	0	0	0	1	1
x_v^5	1	0	0	0	1	0	1	0	1	1
x_v^6	1	0	0	0	0	0	1	1	1	0
x_v^7	1	0	1	0	0	0	0	1	1	0

Table 6.3: Attractors of the mammalian cell cycle network synchronously updated.

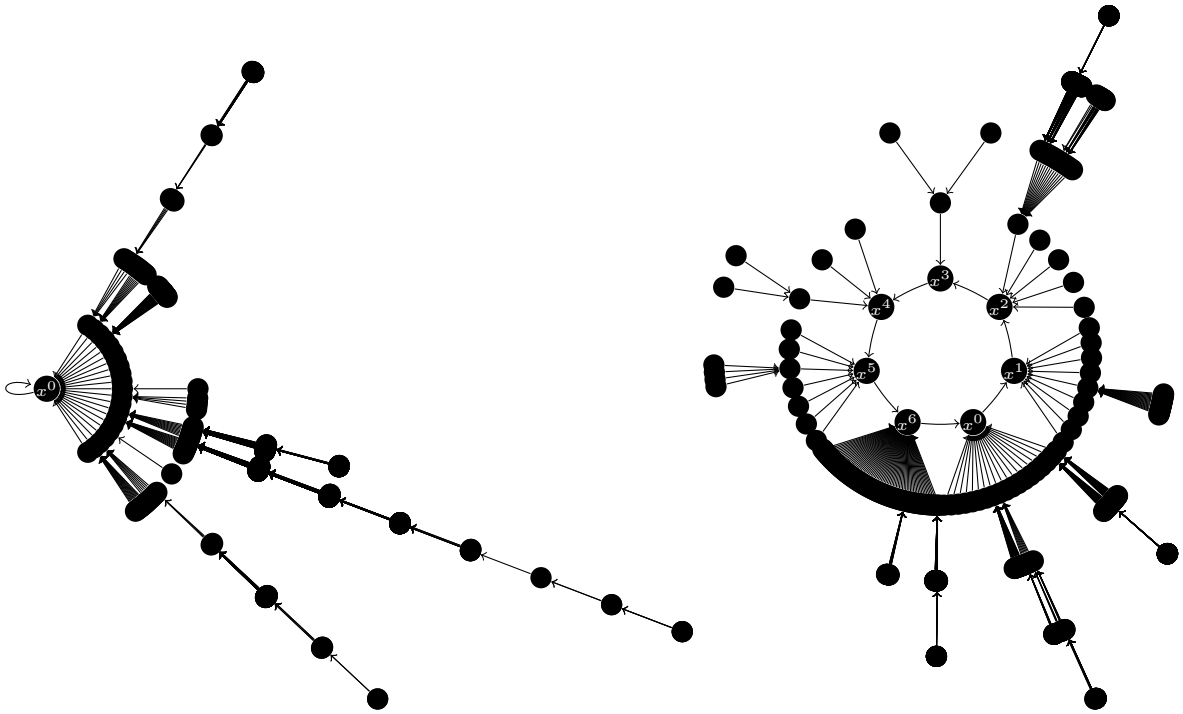


Figure 6.2: Dynamical behavior of the mammalian cell cycle network synchronously updated.

We note that the frozen nodes of $\mathcal{C} = \left[x^k\right]_{k=0}^{7}$ are the nodes in the set $Z = \{v_1, v_2, v_6\}$. It is clear that any sequential update schedule does not have $\mathcal{C}$ as a limit cycle (Goles and Salinas, 2008). Besides, according to Proposition 3.5, any update schedule s satisfying the following property does not has $\mathcal{C}$ as a limit cycle:

$$\forall v \notin Z,\ \forall u \in N_{G^F}^{-}(v)\colon\ \mathrm{lab}_s(u,v) = \ominus$$

For instance,

$$\forall v \notin Z\colon s_v = \{u \in V\colon u \neq v\}\,\{v\}$$

On another hand, by Corollary 3.11, there exists an updated schedule $s \notin [s^p]_{G^F}$, such that $\mathcal{C} \in LC(F, s)$. Moreover, Theorem 3.10 allow us to construct it. In fact, there are several such update schedules. We detail some of them:

$$\begin{aligned}
\forall v \in Z\colon s_v &= \{u \in V\colon u \neq v\}\,\{v\}\\
s_{v_1,v_2} &= \{u \in V\colon u \notin \{v_1, v_2\}\}\,\{v_1, v_2\}\\
s_{v_1,v_6} &= \{u \in V\colon u \notin \{v_1, v_6\}\}\,\{v_1, v_6\}\\
s_{v_2,v_6} &= \{u \in V\colon u \notin \{v_2, v_6\}\}\,\{v_2, v_6\}\\
s_{v_1,v_2,v_6} &= \{u \in V\colon u \notin Z\}\,\{v_1, v_2, v_6\}
\end{aligned}$$

Besides, also have $\mathcal{C}$ as a limit cycle all update schedules that are a division (considering all permutations) of the second block of the last four update schedules above.

On another hand, in Section 4.1.3 another kind of equivalence classes was defined, such that update schedules in the same class have a one-to-one correspondence between their limit cycles, maintaining even their length, but they do not necessarily have limit cycles in common. However, since in this case, the involved nodes are frozen in $\mathcal{C}$, they do share this limit cycle. In this way, according to Proposition 4.16, every cyclic permutation of the update schedules above will have $\mathcal{C}$ as a limit cycle.

Next, we explicit the dynamical behavior of some of these schedules. In Figure 6.3 is shown the update digraph of $N_{v_2} = (F, s_{v_2})$ and in Figure 6.4 is shown its full dynamical behavior.

In Figure 6.5 is shown the update digraph of $N_{6,2} = (F, s_{6,2})$, where $s_{6,2} = \{u \in V\colon u \notin \{v_2, v_6\}\}\,\{v_6\}\,\{v_2\}$. In Figure 6.6 is shown its full dynamical behavior.

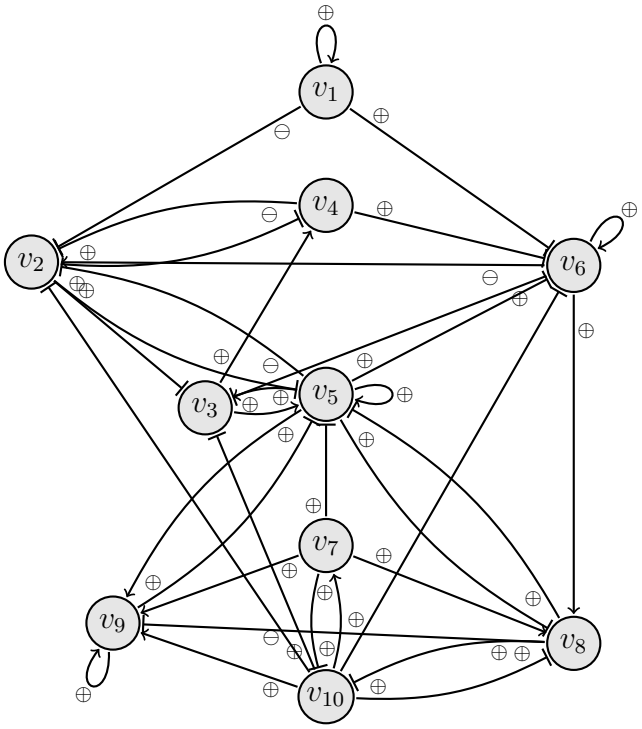


Figure 6.3: $G^F_{s_{v_2}}$.

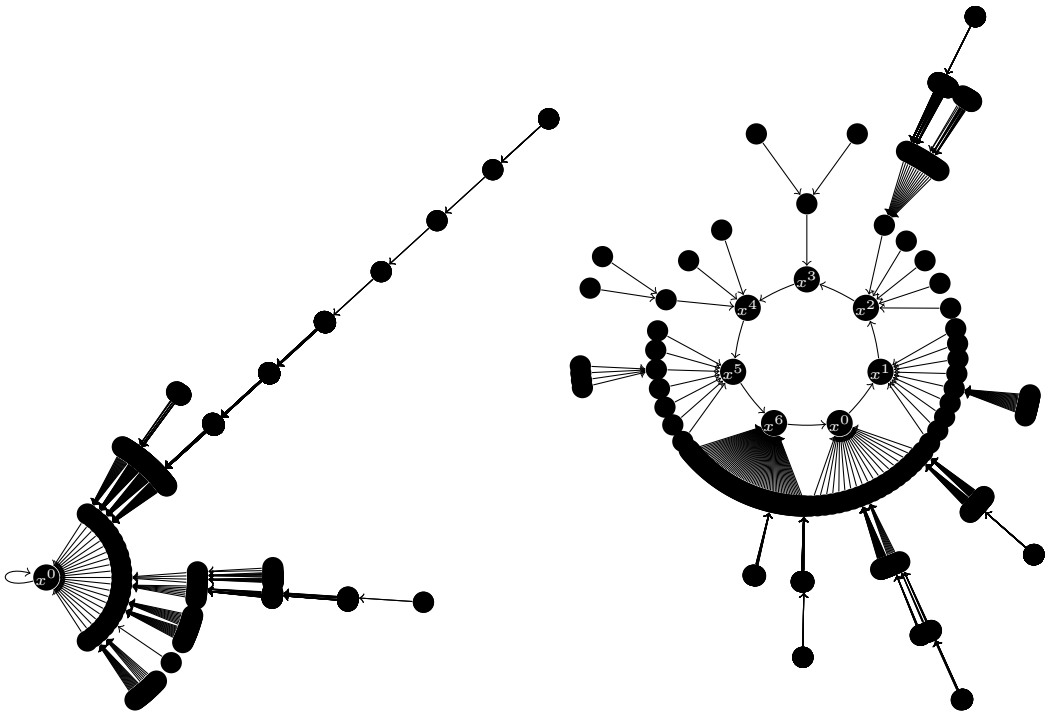


Figure 6.4: Dynamical behavior of N_{v_2}.

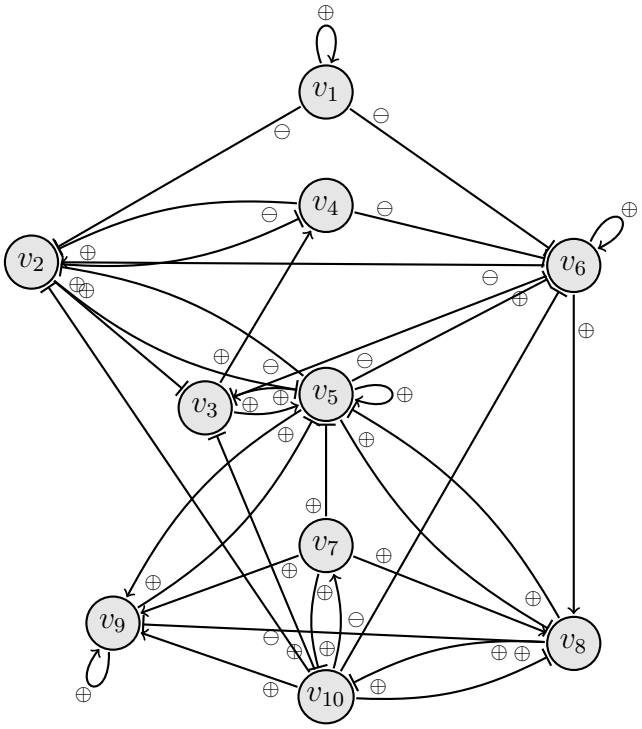


Figure 6.5: $G^F_{s_{6,2}}$.

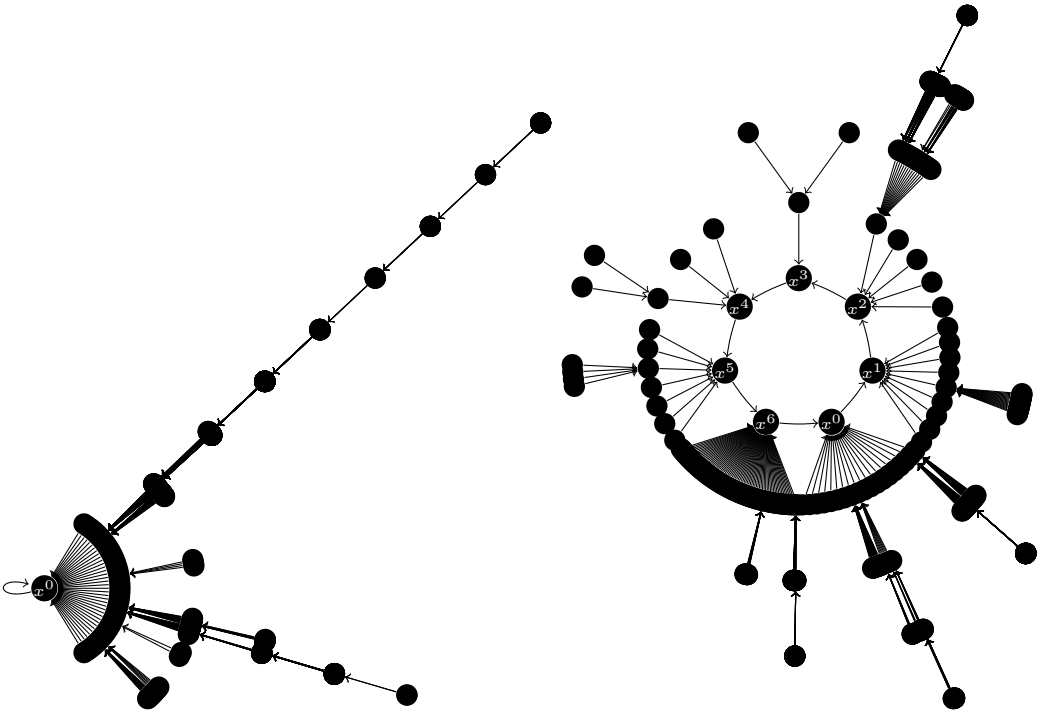


Figure 6.6: Dynamical behavior of $N_{6,2}$.

6.2. Analysis of the robustness of limit cycles of the fission yeast cell-cycle network

In this section we are going to study the robustness of the fission yeast cell-cycle network first introduced in Davidich and Bornholdt (2008) and extensively studied in Goles et al. (2013), where some theoretical and simulation results were given. Just like in the previous section, our aim here is to give some insights to the robustness of the networks when the update schedule is changed, without using any simulations. In particular, we are interested in to find non equivalent updated schedules that yields the same limit cycle that the parallel schedule.

We proceed now to describe the network. The network has 10 nodes, that for simplifying notation we are going to denote as shown in Table 6.4.

Start $\equiv v_1$	Slp1 $\equiv v_6$
Sk $\equiv v_2$	Cdc2/Cdc13* $\equiv v_7$
Cdc2/Cdc13 $\equiv v_3$	Weel/Mik1 $\equiv v_8$
Ste9 $\equiv v_4$	Cdc25 $\equiv v_9$
Rum1 $\equiv v_5$	PP $\equiv v_{10}$

Table 6.4: Notation for the nodes of the fission yeast cell-cycle network.

The local activation functions are threshold functions with an special Heaviside function as described next:

$$\forall \in \{1,\dots,10\} : f_{v_i}(x) = H\left(\sum_{j=1}^{n} w_{ij}x_{v_j} - \theta_i\right) = \begin{cases} 0 & \text{if } \sum_{j=1}^{n} w_{ij}x_{v_j} - \theta_i < 0 \\ 1 & \text{if } \sum_{j=1}^{n} w_{ij}x_{v_j} - \theta_i > 0 \\ x_{v_i} & \text{if } \sum_{j=1}^{n} w_{ij}x_{v_j} - \theta_i = 0 \end{cases}$$

The weight matrix and threshold vector are given in Table 6.5 and the weigh matrix digraph is given in Figure 6.7.

$$W = \left(\begin{array}{c|cccccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} \\ \hline v_1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ v_4 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ v_5 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ v_6 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ v_7 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & 1 & 0 \\ v_8 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ v_9 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ v_{10} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{array}\right), \theta = \begin{pmatrix} 0 \\ 0 \\ -\frac{1}{2} \\ 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Table 6.5: Weight matrix and threshold vector of the fission yeast cell-cycle network.

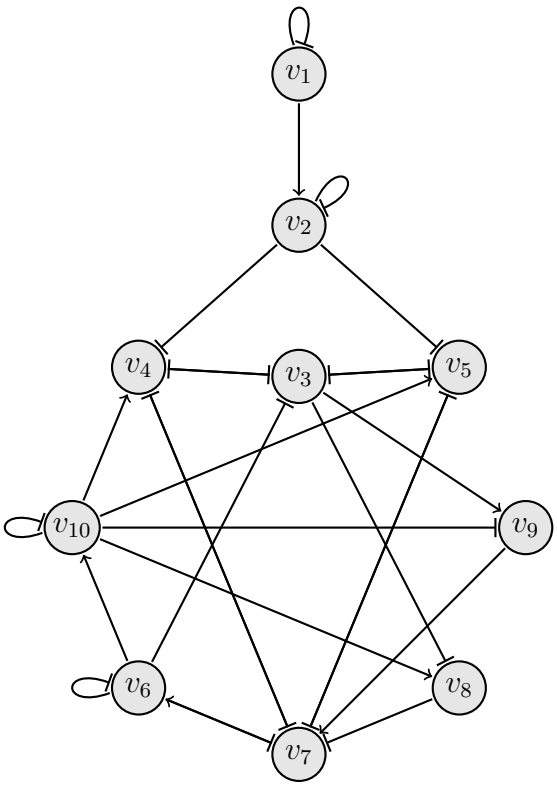


Figure 6.7: Weight matrix digraph of the fission yeast cell-cycle network.

$$
\begin{aligned}
f_{v_1}(x) &= 0\\
f_{v_2}(x) &= x_{v_1}\\
f_{v_3}(x) &= \neg x_{v_4} \wedge \neg x_{v_5} \wedge \neg x_{v_6}\\
f_{v_4}(x) &= (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_4} \wedge \neg x_{v_7}) \vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_4} \wedge x_{v_{10}})\\
&\quad \vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge \neg x_{v_7} \wedge x_{v_{10}})\\
&\quad \vee (\neg x_{v_2} \wedge x_{v_4} \wedge \neg x_{v_7} \wedge x_{v_{10}})\\
&\quad \vee (\neg x_{v_3} \wedge x_{v_4} \wedge \neg x_{v_7} \wedge x_{v_{10}})\\
f_{v_5}(x) &= (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_5} \wedge \neg x_{v_7}) \vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge x_{v_5} \wedge x_{v_{10}})\\
&\quad \vee (\neg x_{v_2} \wedge \neg x_{v_3} \wedge \neg x_{v_7} \wedge x_{v_{10}})\\
&\quad \vee (\neg x_{v_2} \wedge x_{v_5} \wedge \neg x_{v_7} \wedge x_{v_{10}})\\
&\quad \vee (\neg x_{v_3} \wedge x_{v_5} \wedge \neg x_{v_7} \wedge x_{v_{10}})\\
f_{v_6}(x) &= x_{v_7}\\
f_{v_7}(x) &= \neg x_{v_4} \wedge \neg x_{v_5} \wedge \neg x_{v_6} \wedge \neg x_{v_8} \wedge x_{v_9}\\
f_{v_8}(x) &= (\neg x_{v_3} \wedge x_{v_8}) \vee (\neg x_{v_3} \wedge x_{v_{10}}) \vee (x_{v_8} \wedge x_{v_{10}})\\
f_{v_9}(x) &= (x_{v_3} \wedge x_{v_9}) \vee (x_{v_3} \wedge \neg x_{v_{10}}) \vee (x_{v_9} \wedge \neg x_{v_{10}})\\
f_{v_{10}}(x) &= x_{v_6}
\end{aligned}
$$

Table 6.6: Logical functions of the fission yeast cell-cycle network.

The logical equivalent functions are given Table 6.6 and the interaction digraph in Figure 6.8

This network synchronously updated, denoted $N^p = (F, s^p)$ has twelve fixed points and one limit cycle of length 3 as attractors. The limit cycle is described in Table 6.7. The complete dynamical behavior is shown in Figure 6.9

$v \in V\left(G^F\right)$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
x_v^0	0	0	0	0	0	0	0	0	1	1
x_v^1	0	0	1	1	1	0	1	1	0	0
x_v^2	0	0	0	0	0	1	0	0	1	0
x_v^3	0	0	0	0	0	0	0	0	1	1

Table 6.7: Limit cycle of the fission yeast cell-cycle network synchronously updated.

We note that the frozen nodes of $\mathcal{C} = \left[x^k\right]_{k=0}^3$ are the nodes in the set $Z = \{v_1, v_2\}$. It is clear that any sequential update schedule does not have $\mathcal{C}$ as a limit cycle (Goles and Salinas, 2008). Besides, according to Proposition 3.5, any update schedule s satisfying the following property does not have $\mathcal{C}$ as a limit cycle:

$$\forall v \notin Z,\ \forall u \in N_{G^F}^-(v)\colon\ \mathrm{lab}_s(u, v) = \ominus$$

For instance,

$$\forall v \notin Z\colon s_v = \{u \in V\colon u \neq v\}\,\{v\}$$

On another hand, by Corollary 3.11, there exists an updated schedule $s \notin [s^p]_{G^F}$, such that $\mathcal{C} \in LC(F, s)$. Moreover, Theorem 3.10 allow us to construct it. In fact, there are five

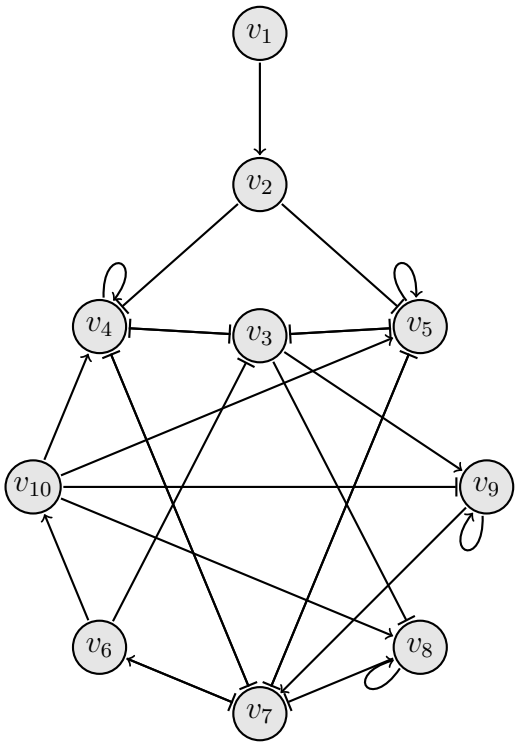


Figure 6.8: Interaction digraph of the fission yeast cell-cycle network.

of such update schedules, that we detail next:

$$\begin{aligned}
\forall v \in Z\colon s_v &= \{u \in V\colon u \neq v\}\,\{v\} \\
s_{v_1,v_2} &= \{u \in V\colon u \notin \{v_1, v_2\}\}\,\{v_1, v_2\} \\
s_{1,2} &= \{u \in V\colon u \notin \{v_1, v_2\}\}\,\{v_1\}\,\{v_2\} \\
s_{2,1} &= \{u \in V\colon u \notin \{v_1, v_2\}\}\,\{v_2\}\,\{v_1\}
\end{aligned}$$

We note that in this case, due to the topology of the interaction digraph, the update schedules s_{v_1,v_2}, s_{v_1} and $s_{2,1}$ belong to the parallel update schedule class. Besides, s_{v_2} and $s_{1,2}$ belong to the same class. Therefore, Theorem 3.10 only ensure us one different update schedule class that has $\mathcal{C}$ as a limit cycle.

On another hand, in Section 4.1.3 was defined another kind of equivalence classes, such that update schedules in the same class have a one-to-one correspondence between their limit cycles, maintaining even their length, but they do not necessarily have limit cycles in common. However, since in this case, the involved nodes are frozen in $\mathcal{C}$, they do share this limit cycle. In this way, according to Proposition 4.16, every cyclic permutation of the update schedules above will have $\mathcal{C}$ as a limit cycle.

Thus, we have at least two equivalent classes different from the parallel one that yield $\mathcal{C}$ as a limit cycle. They are, s_{v_2} and $s'_{v_2} = \{v_2\}\,\{u \in V\colon u \neq v_2\}$. In Figure 6.10 is shown the

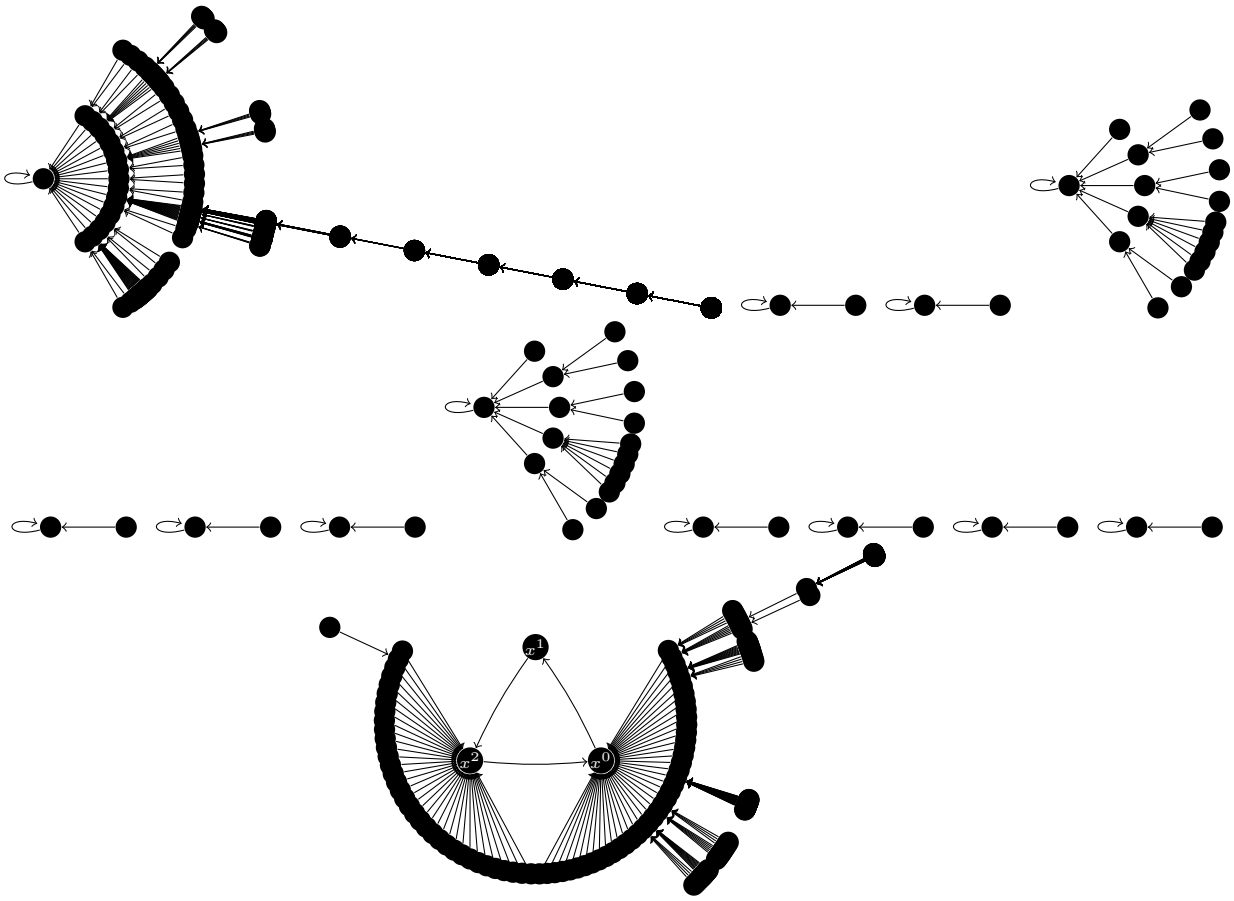


Figure 6.9: Dynamical behavior of the fission yeast cell-cycle network synchronously updated.

update digraph of $N_{v_2} = (F, s_{v_2})$ and in Figure 6.11 is shown its full dynamical behavior. In Figure 6.12 is shown the update digraph of $N'_{v_2} = (F, s'_{v_2})$ and in Figure 6.13 is shown its full dynamical behavior.

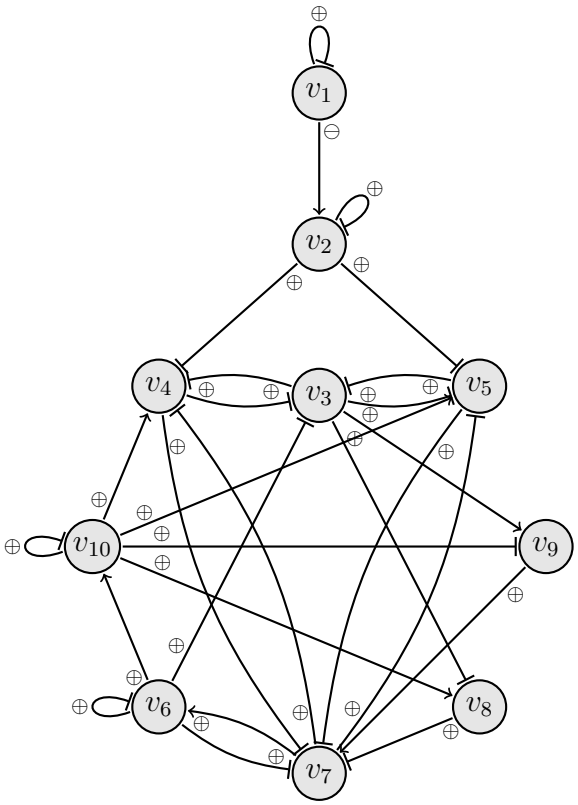


Figure 6.10: $G^F_{s_{v_2}}$.

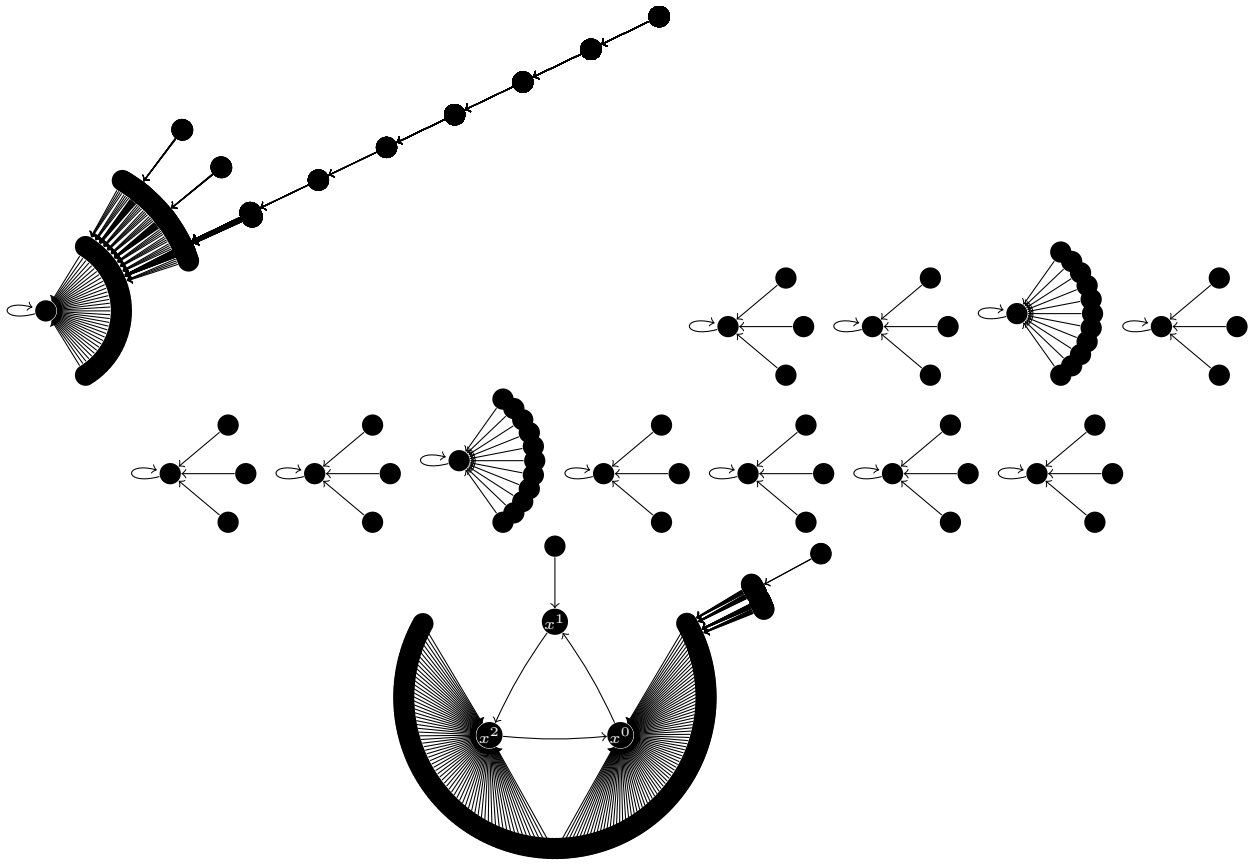


Figure 6.11: Dynamical behavior of N_{v_2}.

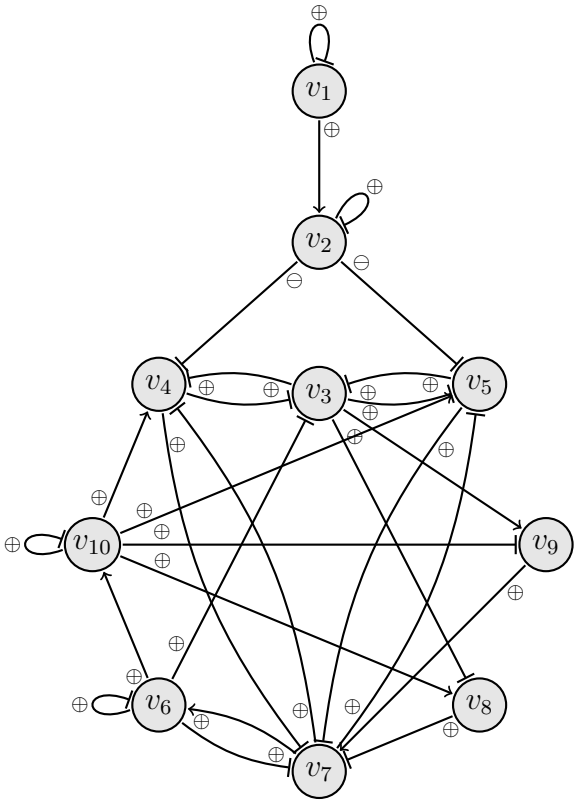


Figure 6.12: $G^F_{s'_{v_2}}$.

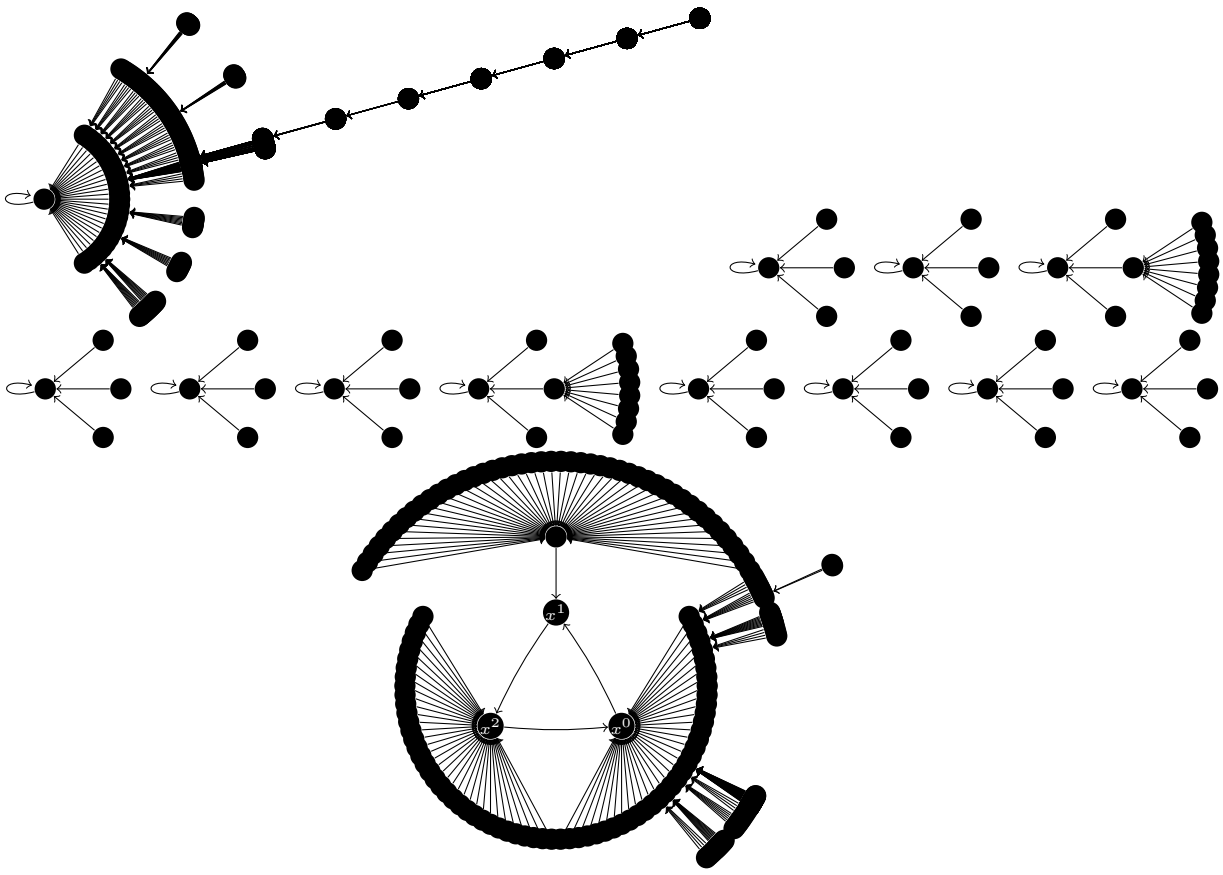


Figure 6.13: Dynamical behavior of N'_{v_2}.

Chapter 7

Conclusions

7.1. English version

In this thesis, we were interested in studying the dynamical behavior, mainly the limit cycles, of a given Boolean function updated under different deterministic update schedules (parallel, block sequential and sequential).

Our first subject of interest (Chapter 3), was to study the relationship between the update digraphs and the shared limit cycles of Boolean networks which differ only in their update schedule. This was motivated by a result proved in Aracena et al. (2009), which states that update schedules in the same equivalence class (equal update digraphs) yield the same dynamical behavior. Thus, our aim here was to define, if possible, new equivalence classes such that update schedules in the same class have the same set of limit cycles but not necessarily the whole dynamics. In this context, we first proved that the decision problems involved are all NP-hard (that is, intractable computationally problems). Next, we showed that it is not possible to define the desired equivalence classes, since for any two given different update digraphs, there always exists a Boolean function having as interaction digraph the given digraph and such that there is at least one limit cycle that is not common in both networks. This tell us that the information provided by the update digraphs is not sufficient to determine whether the networks have the same set of limit cycles.

Our next step was to study what information can provide the update digraph about the sharing of limit cycles. We first propose a polynomial algorithm that works as necessary condition for two Boolean networks to share at least one limit cycle. Furthermore, we show that the existence of update schedules preserving some limit cycle depends strongly on the global activation function and the structural properties of the network (see Example 3.1). In some Boolean networks, the only update schedules keeping a given limit cycle are those preserving the whole dynamical behavior of the network. This fact was observed previously by Goles and Salinas (2008) in the particular case of parallel and sequential update schedules. However, it is possible to define under certain conditions a set of non equivalent update schedules, that preserve some given limit cycles. These update schedules depend strongly on

the frozen nodes of the preserved limit cycles. We also give some trivial cases in which the related decision problems are polynomial. It is an open problem to know whether there are non trivial families of Boolean networks and interaction digraphs in which these problems are polynomial.

A more basic problem than deciding if there are two non equivalent update schedules that share a limit cycle for a given Boolean network, is to determine if there exists an update schedule that yields any limit cycle (referred in Chapter 4 as LCE problem). The related problem that consists in determining whether the parallel update schedule yields any limit cycle is known to be NP-Hard (Just, 2006). In this context, we proved that the LCE problem is NP-Hard even for AND-OR functions and in functions having a symmetric interaction digraph. This problem in symmetric OR functions is polynomial, this was proved in Goles and Noual (2012). Here we extend this result to symmetric AND-OR functions. In this case, we prove that such an update schedule exists if and only if the limit cycle set of the network updated in parallel is non empty. Furthermore, this last condition is characterized by a property in the interaction digraph that can be verified in polynomial time. This tell us that the simplicity of the Boolean function and its topological architecture are both necessary for establishing polynomiality. The general OR case remains open, nevertheless we give a characterization that describes the existence of solutions for it. This characterization define a new decision problem, that consists in labeling a given digraph such that the resulting labeled digraph is an update digraph, and the associated parallel digraph is non primitive. The complexity of this problem remains open, but we give some partial insights to solve it. In particular, we showed that if there is a digraph that has no solution, then neither any digraph that contains it. This allow us to search for forbidden configurations when analyzing the problem.

On the other hand, we also studied the problem of determining whether there exists an update schedule that does not generate any limit cycle and we proved that it is NP-Hard. In Goles and Salinas (2010) is shown a polynomial algorithm that constructs from a given monotonic network another network such that, when updated in parallel, it has the same fixed points but without limit cycles. Motivated by that work, we studied if it was possible to do the same but such that the constructed network was the associated parallel network. However, we show that this is not always possible by exhibiting an example of an AND-OR function that has limit cycles updated under any update schedule (see Example 4.5). Nevertheless, we proved that such an update schedule always exists in the OR case. Besides, an update schedule yielding this property can be found in polynomial time (Goles and Noual, 2012). It remains open the study of the complexity for other families of Boolean functions.

Our next subject of study is related to the inference of update schedules yielding a given dynamical property, mainly the limit cycles (Chapter 5). It is clear that as a prior step, we need to consider all the results developed in Chapter 4, since it would be pointless to search for an update schedule, that has a particular sequence as a limit cycles, if the Boolean network cannot cycle for any update schedule. For starting our analysis, we considered a more basic case, that is a single transition. We proved that it is NP-Complete even for OR functions. In this case, we give a characterization of the existence of solution in terms of

the interaction digraph and an algorithm (OR FT Algorithm) to test it, this algorithm is polynomial in the symmetric case. It remains open whether there is another non trivial class of OR networks in which this algorithm becomes polynomial. About the sequence problem, we also proved that it is NP-Complete even for OR networks. It would be no surprise that the symmetric OR case is once again polynomial. But just as in the case of the LCE problem, we proved that symmetry by it self is not enough for establishing polynomiality. In addition, we proved that the AND-OR case restricted to limit cycles of length two is also polynomial. However, it is not clear and it remains open what happens with limit cycles of a fixed length greater than two. It is known that the satisfiability problem of normal conjunctive formulas has a change of phase in its complexity when the number of clauses is increased, and we conjecture that the complexity of the problem for this case has a similar behavior. Finally, we also studied another related problems and we proved that all of them are NP-Complete even in the OR case.

Last but not least, we applied the results from Chapter 3 and Chapter 4 to study the robustness of limit cycles against changes in the update schedule of two known genetic regulatory networks: the mammalian cell cycle network and the fission yeast cell cycle network. The limit cycles of these networks updated in parallel have frozen nodes, which allow us to construct some non equivalent update schedules that share those limit cycles. Furthermore, using the cyclic equivalence classes of update schedules defined in Chapter 4 (that consists in cyclic permutations of the blocks of an update schedule that generates shifted limit cycles), and due to the frozen characteristic of the nodes involved, we were able to define a few more non equivalent update schedules that also generates the same limit cycles. Besides, we give some local conditions on the update digraphs, such that any update schedule that satisfies them, does not generate the same limit cycles.

7.2. Spanish version

En esta tesis, estuvimos interesados en estudiar el comportamiento dinámico, principalmente de los ciclos limites, de una función Booleana dada actualizada bajo diferentes esquemas de actualización deterministas (paralelo, secuencial, bloque-secuencial).

Nuestro primer tema de interés (Capítulo 3), fue estudiar la relación entre los dígrafos de actualización y los ciclos limites compartidos de redes Booleanas que solo difieren en su esquema de actualización. Esto motivado por un resultado probado en Aracena et al. (2009), que establece que los esquemas de actualización en la misma clase de equivalencia (dígrafos de actualización iguales) generan el mismo comportamiento dinámico. Así, nuestro objetivo fue determinar, si es que es posible, nuevas clases de equivalencia de forma tal que los esquemas de actualización en la misma clase produzcan el mismo conjunto de ciclos limites pero no necesariamente toda la dinámica. En este contexto, primero probamos que los problemas de decisión relacionados son todos NP-Hard (esto es, problemas computacionalmente intratables). A continuación, mostramos que no es posible definir las clases de equivalencia deseadas, puesto que para cualquier par de dígrafos de actualización distintos, siempre existe una función Booleana que tiene como dígrafo de interacción el dígrafo dado, y de forma tal que hay al menos un ciclo limite que no es común en ambas redes. Esto nos dice que la información proporcionada por los dígrafos de actualizacion no es suficiente para determinar si las redes tienen el mismo conjunto de ciclos limites.

Nuestro siguiente paso fue estudiar que información puede proveer el dígrafo de actualización acerca de la capacidad de compartir ciclos limites. Primero propusimos un algoritmo polinomial que trabaja como una condición necesaria para que dos redes Booleanas compartan al menos un ciclo limite. Mas aún, mostramos que la existencia de esquemas de actualización que preserven algunos ciclos limites depende fuertemente de las funciones locales de activación y de las propiedades estructurales de la red (ver Ejemplo 3.1). En algunas redes Booleanas, los únicos esquemas que preservan un ciclo limite dado son aquellos que preservan todo el comportamiento dinámico de la red. Este hecho fue observado previamente por Goles and Salinas (2008) en el caso particular de los esquemas paralelo y secuencial. Sin embargo, es posible definir bajo ciertas condiciones un conjunto de esquemas de actualización no equivalentes que preservan un conjunto dado de ciclos limites. Estos esquemas dependen fuertemente de los nodos "frozen" de los ciclos limites preservados. También, damos algunos casos triviales en el que los problemas de decisión relacionados son polinomiales. Queda abierto el saber si existen familias no triviales de redes Booleanas y dígrafos de interacción en el que estos problemas sean polinomiales.

Un problema mas básico que el decidir si hay dos esquemas de actualización no equivalentes que comparten un ciclo limite para una red Booleana dada, es determinar si existe un esquema de actualización que genere algún ciclo limite (referido como problema LCE en el Capítulo 4). El problema relacionado que consiste en determinar si el esquema paralelo genera ciclos limites es conocido ser NP-Hard (Just, 2006). En este contexto, probamos que el problema LCE es NP-Hard incluso para funciones AND-OR y en funciones que tienen un

dígrafo de interacción simétrico. En Goles and Noual (2012), se probó que este problema es polinomial en funciones OR simétricas. Aquí, extendemos este resultados a funciones AND-OR simétricas. En este caso, probamos que tal esquema existe si y solo si el conjunto de ciclos limites de la red iterada en paralelo es no vacío. Mas aún, esta última condición esta caracterizada por una propiedad en el dígrafo de interacción que puede ser verificada en tiempo polinomial. Esto nos dice que la simplicidad de la función Booleana y de su arquitectura topológica son ambas necesarias para establecer la polinomialidad. El caso OR general permanece abierto, pero damos una caracterización que describe la existencia de solución. Esta caracterización define un nuevo problema de decisión, que consiste en etiquetar un dígrafo dado de forma tal que el dígrafo etiquetado resultante sea un dígrafo de actualización, y que el dígrafo paralelo asociado sea no primitivo. La complejidad de este problema permanece abierta, pero damos algunos resultados parciales al respecto. En particular, mostramos que si hay un dígrafo en el cual este problema no tiene solución, entonces tampoco hay solución en cualquier dígrafo que lo contenga. Esto nos permite buscar por configuraciones prohibidas cuando lo analizamos.

Por otra parte, también estudiamos el problema de determinar si existe un esquema de actualización que no genere ciclos limites y probamos que es NP-Hard. En Goles and Salinas (2010) se da un algoritmo polinomial que construye a partir de una red monótona dada, otra red que, cuando es iterada en paralelo, tiene los mismos puntos fijos que la red original, pero sin ciclos limites. Motivados por este trabajo, estudiamos si era posible de realizar lo mismo, pero de forma tal que la red construida coincidiera con la red paralela asociada. Sin embargo, mostramos que esto no es siempre posible exhibiendo un ejemplo de función AND-OR que genera ciclos limites iterada bajo cualquier esquema de actualización (ver Ejemplo 4.5). No obstante, probamos que tal esquema siempre existe en el caso OR. Además, un esquema de actualización con esta propiedad puede ser encontrado en tiempo polinomial (Goles and Noual, 2012). Permanece abierto el estudio de la complejidad para otras familias de funciones Booleanas.

Nuestro siguiente tema de estudio esta relacionado con la inferencia de esquemas de actualización que producen alguna propiedad dinámica dada, principalmente ciclos limites (Capítulo 5). Es claro que como un paso previo, necesitamos considerar los resultados desarrollados en el Capítulo 4, puesto que no tendría sentido el buscar un esquema de actualización que tenga una secuencia particular como ciclo limite, si la red no genera ciclos limites con ningún esquema. Para comenzar nuestro análisis, consideramos un caso mas básico, el de una sola transición. Probamos que este problema es NP-Completo incluso para redes OR. En este caso, damos una caracterización de la existencia de solución en términos del dígrafo de interacción y un algoritmo (OR FT Algorithm) para testearlo, que se vuelve polinomial en el caso simétrico. Permanece abierto si existen otras clases de redes OR en el que este algoritmo se vuelve polinomial. Con respecto al problema de la secuencia, también probamos que es NP-Completo incluso para redes OR. No debería sorprender que el caso OR simétrico es nuevamente polinomial. Pero al igual que el caso del problema LCE, probamos que solo la simetría no es suficiente para establecer la polinomialidad. Además, probamos que el caso AND-OR restringido a ciclos limites de largo dos es polinomial. Sin embargo, no es claro

y permanece abierto que pasa con ciclos limites con largo fijo mayor que dos. Es conocido que el problema de satisfacibilidad de formulas normales conjuntivas tiene un cambio de fase en su complejidad cuando el número de clausulas es aumentado, y conjeturamos que la complejidad del problema en este caso debería presentar un comportamiento similar. Finalmente, también estudiamos otros problemas de decisión relacionados y probamos que todos son NP-Completos incluso en el caso OR.

Por último pero no menos importante, aplicamos los resultados de los Capítulos 3 y 4 para estudiar la robustez de los ciclos limites contra cambios en el esquema de actualización de dos conocidas redes de regulación génica: la red del ciclo celular mamífero y la red del ciclo celular de la levadura de fisión. Los ciclos limites de estas redes actualizadas en paralelo tienen nodos frozen, lo que nos permite construir algunos esquemas no equivalentes que comparten esos ciclos limites. Mas aún, usando las clases de equivalencia cíclica de esquemas de actualización definidas en el Capítulo 4 (que consiste en permutaciones cíclicas de los bloques de un esquema que genera ciclos limites desplazados), y a la característica frozen de los nodos involucrados, fuimos capaces de definir unos pocos más esquemas no equivalentes que también generan los mismos ciclos limites. Por otra parte, dimos algunas condiciones locales en los dígrafos de actualización, de forma tal que cualquier esquema que las satisfaga, no genera los mismos ciclos limites.

Appendix A

Algorithms

In this chapter, we are going to explicit the some algorithms presented in previous chapters.

A.1. Preliminary algorithms

In this section, we are going to introduce some existents algorithms that we are going to use in the following ones.

We recall that in Aracena et al. (2011) was proven that to decide if a partial labeled digraph is an update digraph can be done in polynomial time, as well as finding an update schedule that generates a certain update digraph. Besides, some algorithms were given that we will call `CheckingUD` with input a partial or total labeled digraph, and `UD2US` with input an update digraph, respectively. The output of the first one is a logical value, TRUE or FALSE, and the output of the second one is the found update schedule.

On another hand, in Aracena et al. (2011) was also proven the Extension Theorem, that states that if a partial labeled digraph is an update digraph, then there always exists an extension. Besides, we propose the following algorithm that can find one of such extensions in polynomial time:

Here, the input is the partial labeled digraph $G_{\widetilde{\text{lab}}}$ and the output is the total labeled digraph G_{lab}. The algorithm basically tries to label an unlabeled arc as $\oplus$ and test if the generated labeled digraph is an update digraph. If it is not, the Extension Theorem ensures us labeling the arc as $\ominus$, then the resulting labeled digraph it is an update digraph.

Procedure ExtendingLab($G_{\widetilde{\text{lab}}}$)

Output: G_{lab}

1 $\forall e \in A$: $\text{lab}(e) \leftarrow \bigcirc$
2 $\forall e \in Sup\left(G_{\widetilde{\text{lab}}}\right)$: $\text{lab}(e) \leftarrow \widetilde{\text{lab}}(e)$
3 **foreach** $e \notin \text{Sup}(G_{\text{lab}})$ **do**
4 $\text{lab}(e) \leftarrow \oplus$
5 **if** ¬`CheckingUD` (G_{lab}) **then**
6 $\text{lab}(e) \leftarrow \ominus$
7 **end**
8 **end**

A.2. Symmetric AND-OR Limit Cycle Existence problem

In Chapter 4 was found a polynomial testable condition that help us to decide whether there exists an update schedule such that a given symmetric AND-OR function generates limit cycles when it is updated under such update. Besides, one of the solutions of the problem is the parallel update schedule. The algorithm is straightforward, and we just need to consider a procedure, that we will call `GetAOA`, that gets as input the interaction digraph, an it gives as output a set containing each element of the AOA decomposition of the interaction digraph. In this way, the algorithm gets as follows:

SYMMETRIC AND-OR LCE Algorithm

Input: F a symmetric AND-OR function
Output: true or false

1 $AOA \leftarrow$`GetAOA` $(G^F, V_{\text{OR}}(G^F), V_{\text{AND}}(G^F))$
2 **foreach** $E \in AOA$ **do**
3 **if** E *is bipartite* **then**
4 **return** `true`
5 **end**
6 **end**
7 **return** `false`

A.3. OR Feasible Transition problem

We can summarize the results of Section 5.1.1 in the OR FT Algorithm presented is this one. Before that, we introduce two procedures:

- CNTreatment: it makes the transformation when there are constant nodes (see Lemma 5.3 and Remark 5.3).
- CheckingNC: it checks all necessary conditions presented in Theorem 5.9.

Besides, we are going to use the next procedures, from which we are not going to give an explicit algorithm:

- **GettingSets**(G, x, y)**:** used in Line 2 of the OR FT Algorithm, it constructs the respective node sets according to Definition 5.1, 5.2 and 5.3, and Remark 5.5.
- **SNSearch**$\left(N_1^0, \ldots, N_{q_2}^0\right)$**:** used in Line 7 of the OR FT Algorithm, it takes one node in each N_t^0 and one incoming arc from V_{10} for each node, label the arcs as $\oplus$ and check if the generated labeled digraph is an update digraph. If it is not, it check another incoming arcs combination, and then another set of nodes. If it find one, the output is the start nodes set V_0 and the arc set A_0. We note that checking if the labeled digraphs are update digraph can be done in polynomial time, but checking all the possible combination it is not.
- **SpanningTree**(i, G)**:** used in Line 15 of the OR FT Algorithm, is a polynomial procedure to get the nodes of an spanning tree in G starting at i.

Procedure CNTreatment$(G, V_{00}, V_{11}, V_{10}, V_{01})$

Output: $\left(\hat{G}, V_{01}, V_{10}, V_R\right)$

if $V_{00} \neq \emptyset$ **then**
  $V_{10} \leftarrow V_{00} \cup V_{10}$
  $\hat{A} \leftarrow A \setminus \{(i,j) \in A \colon i \in V_{00}\}$
else
  $\hat{A} \leftarrow A$
end
if $V_{11} \neq \emptyset$ **then**
  $V_{01} \leftarrow V_{11} \cup V_{01}$
  $\hat{A} \leftarrow \hat{A} \setminus \{(i,j) \in A \colon i \in V_{11}\}$
  $N_{11}^+ \leftarrow \bigcup_{i \in V_{11}} N_{G^F}^+(i)$
  $\hat{A} \leftarrow \hat{A} \setminus \left\{(i,j) \in A \colon j \in N_{11}^+ \cap V_{01}\right\}$
  $V_R \leftarrow N_{11}^+ \cap V_{01}$
else
  $V_R \leftarrow \emptyset$
end

Procedure CheckingNC($G, \hat{G}, V_{00}, V_{11}, V_{10}, V_{01}, L^-, O, N_1^0, \ldots, N_{q_2}^0$)

Output: true or false

if $\exists i \in V_{01}$: $N_G^-(i) \subseteq V_{00}$ **then**
  return false //Lemma 5.2
else if $\exists i \in V_{10}$: $N_G^-(i) \cap V_{11} \neq \emptyset$ **then**
  return false //Lemma 5.2
else if $\hat{G}[V_{10}]$ *has cycles* **then**
  return false //Lemma 5.6
else if $L^- = \emptyset$ **then**
  return false //Corollary 5.8
else if $\exists t \in \{1, \ldots, q_2\}$: N_t^0 *is non trivial and* $N_t^0 \subseteq O$ **then**
  return false //Lemma 5.7
else
  return true
end

OR FT Algorithm

Input: $G = (V, A)$; $x, y \in \{0,1\}^{|V|}$
Output: $\left(s, \hat{G}_{\widetilde{\text{lab}}}, G_{\text{lab}}\right)$ or false

`//initialization`
$\forall e \in \hat{A}$: $\widetilde{\text{lab}}\,(e) \leftarrow \bigcirc$
$\left(V_{00}, V_{11}, V_{10}, V_{01}, L^+, L^-, O, P_1^0, \ldots, P_q^0, N_1^0, \ldots, N_{q_2}^0\right) \leftarrow$ `GettingSets` (G, x, y)
`//constant nodes transformation`
$\left(\hat{G}, V_{01}, V_{10}, V_R\right) \leftarrow$ `CNTreatment` $(G, V_{00}, V_{11}, V_{10}, V_{01})$
if `CheckingNC` $\left(G, \hat{G}, V_{00}, V_{11}, V_{10}, V_{01}, L^-, O, N_1^0, \ldots, N_{q_2}^0\right)$ **then**
  `//labeling the arcs incoming to` V_{10}
  $\forall i \in V_{10},\ \forall j \in N_{\hat{G}}^-(i) \cap V_{10}$: $\widetilde{\text{lab}}(j,i) \leftarrow \ominus$
  $\forall i \in L^+,\ \forall j \in N_{\hat{G}}^+(i) \cap V_{10}$: $\widetilde{\text{lab}}(i,j) \leftarrow \oplus$
  `//start nodes search and labeling, the only non polynomial line in the algorithm!`
  $(V_0, A_0) \leftarrow$ `SNSearch` $\left(N_1^0, \ldots, N_{q_2}^0\right)$
  if $A_0 = \emptyset$ **then**
    return false
  else
    $\forall e \in A_0$: $\widetilde{\text{lab}}\,(e) \leftarrow \oplus$
    $V_R \leftarrow V_0 \cup V_R$
    `//spanning tree construction and labeling`
    $S_T \leftarrow \emptyset$
    foreach $r \in V_R$ **do**
      $S_T \leftarrow S_T \cup$ `SpanningTree` $\left(r, \hat{G}\left[\left(\{r\} \cup R_{\hat{G}}^+(r) \cap V_{01}\right) \setminus S_T\right]\right)$
    end
    $\forall i \in S_T,\ \forall j \in N_{\hat{G}}^+(i) \cap S_T$: $\widetilde{\text{lab}}(i,j) \leftarrow \ominus$
    `//getting the remaining labels`
    $G_{\text{lab}} \leftarrow$ `ExtendingLab(`$\hat{G}_{\widetilde{\text{lab}}}$`)`
    `//getting the update schedule`
    $s \leftarrow$ `UD2US(`G_{lab}`)`
  end
else
  return false
end

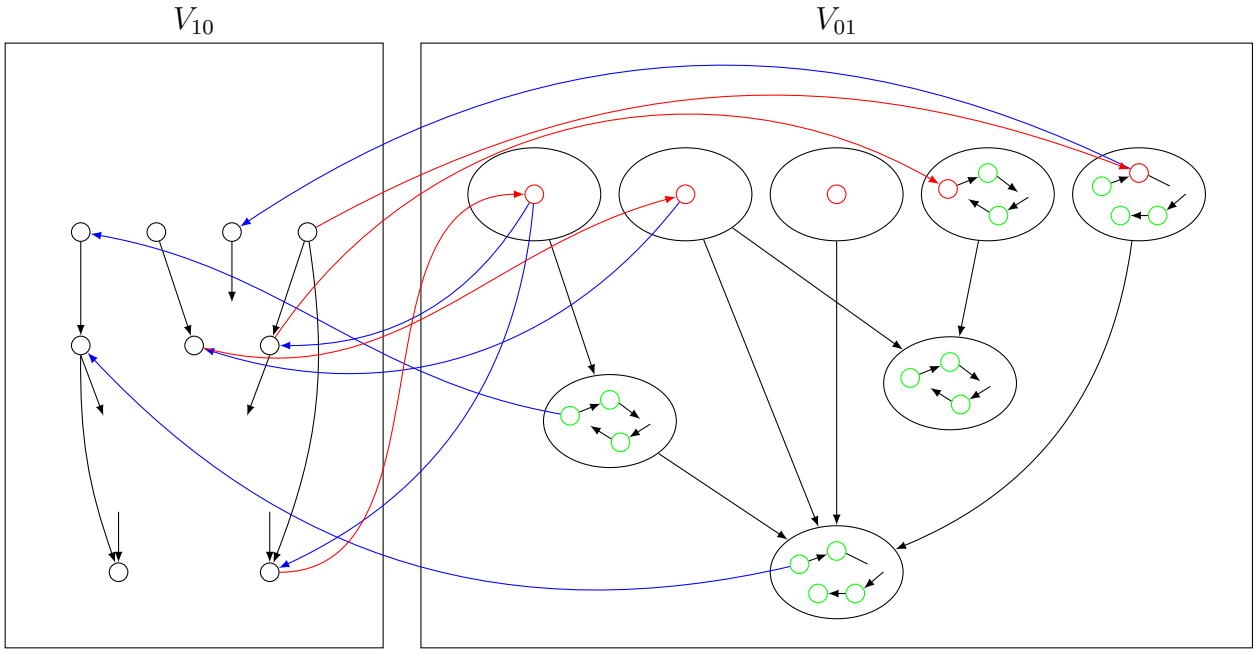


Figure A.1: Interaction digraph structure after applying the OR FT Algorithm.

We note that the OR FT Algorithm it is also valid for AND-OR functions, after applying the transformation described in Proposition 5.4.

On another hand, we recall that in Proposition 5.12 was proven that SYMMETRIC OR FT is polynomial. In terms of the OR FT Algorithm, we just need to change Line 7 by:

$$(V_0, A_0) \leftarrow \texttt{SNElection}\left(N_1^0, \ldots, N_{q_2}^0\right)$$

where the `SNElection` procedure can be described as follows:

Procedure SNElection$(N_1^0, \ldots, N_{q_2}^0)$

Output: (V_0, A_0)

$V_0 \leftarrow \emptyset$
$A_0 \leftarrow \emptyset$
foreach $t \in \{1, \ldots, q_2\}$ **do**
    Choose a node $v_t \in N_t^0$
    Choose a node $u_t \in N_{\hat{G}}^-(v_t) \cap V_{10}$
    $V_0 \leftarrow V_0 \cup \{u_t\}$
    $A_0 \leftarrow A_0 \cup \{(u_t, v_t)\}$
end

A.4. Feasible Limit Cycle problem

In Proposition 5.17 was proven that SYMMETRIC OR FLC is polynomial, besides the parallel update schedule is the only possible solution. The algorithm is straightforward and can be described as follows:

SYMMETRIC OR FLC Algorithm

Input: F symmetric OR function; $x^1, \ldots, x^p = x^0$
Output: true or false

```
1  if p = 2 then
2      if G^F is bipartite then
3          if F(x^0) ≠ x^1 then
4              return false
5          else if F(x^1) ≠ x^0 then
6              return false
7          else
8              return true
9          end
10     else
11         return false
12     end
13 else
14     return false
15 end
```

On another hand, it was proved in Proposition 5.19 that OR 2-FLC is also polynomial. The algorithm is described in the proof, and here we explicit it in the OR 2-FLC Algorithm described next.

We note that the `for` loop in Line 1 of the OR 2-FLC Algorithm initializes the partial label $\widetilde{\text{lab}}$ with $Sup\left(G^F_{\widetilde{\text{lab}}}\right) = \emptyset$. Besides, the `GetVpq` procedure used in Line 4 obtains the respective sets according to Definition 5.1.

Finally, we note that this algorithm is easily extensible to the AND-OR case, making the analogous analysis to the AND nodes, and checking the respectives connections between AND and OR nodes.

OR 2-FLC Algorithm

Input: F an OR function; $x^0, x^1, x^2 \equiv x^0$
Output: s or false

for $a \in A\left(G^F\right)$ **do**
  $\widetilde{\text{lab}}(a) \leftarrow \bigcirc$
end
$(V_{00}, V_1, V_{10}, V_{01}, V_c) \leftarrow \texttt{GetVpq}(x^0, x^1)$
foreach $v \in V_{11}$ **do**
  if $\exists u \in R^+_{G^F}(v)\colon u \notin V_{11}$ **then**
    return `false`
  end
end
foreach $v \in V_{00}$ **do**
  if $\exists u \in R^-_{G^F}(v)\colon u \notin V_{00}$ **then**
    return `false`
  end
end
foreach $v \in V_{10}$ **do**
  foreach $u \in N^-_{G^F}(v) \cap V_{01}$ **do**
    $\widetilde{\text{lab}}(u, v) \leftarrow \oplus$
  end
  foreach $u \in N^-_{G^F}(v) \cap V_{10}$ **do**
    $\widetilde{\text{lab}}(u, v) \leftarrow \ominus$
  end
end
foreach $v \in V_{01}$ **do**
  foreach $u \in N^-_{G^F}(v) \cap V_{01}$ **do**
    $\widetilde{\text{lab}}(u, v) \leftarrow \ominus$
  end
  foreach $u \in N^-_{G^F}(v) \cap V_{10}$ **do**
    $\widetilde{\text{lab}}(u, v) \leftarrow \oplus$
  end
end
if `CheckingUD` $(G^F_{\widetilde{\text{lab}}})$ **then**
  $G_{\text{lab}} \leftarrow \texttt{ExtendingLab}(G^F_{\widetilde{\text{lab}}})$
  $s \leftarrow \texttt{UD2US}(G_{\text{lab}})$
else
  return `false`
end

Bibliography

Abou-Jaoudé, W., Ouattara, D., Kaufman, M., 2009. From structure to dynamics: frequency tuning in the p53-mdm2 network: I. logical approach. Journal of Theoretical Biology 258, 561–577.

Abou-Jaoudé, W., Ouattara, D., Kaufman, M., 2010. From structure to dynamics: frequency tuning in the p53-mdm2 network: Ii. differential and stochastic approaches. Journal of Theoretical Biology 264, 1177–1189.

Akutsu, T., Miyano, S., Kuhara, S., et al., 1999. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In: Pacific Symposium on Biocomputing. Vol. 4. pp. 17–28.

Albert, R., Othmer, H. G., 2003. The topology of the regulatory interactions predicts the expression pattern of the drosophila segment polarity genes. Journal of Theoretical Biology 223, 1–18.

Aracena, J., Demongeot, J., Fanchon, E., Montalva, M., 2013a. On the number of different dynamics in boolean networks with deterministic update schedules. Mathematical biosciences 242 (2), 188–194.

Aracena, J., Fanchon, E., Montalva, M., Noual, M., 2011. Combinatorics on update digraphs in Boolean networks. Discrete Applied Mathematics 159, 401–409.

Aracena, J., Goles, E., Moreira, A., Salinas, L., 2009. On the robustness of update schedules in Boolean networks. Biosystems 97, 1–8.

Aracena, J., Gómez, L., Salinas, L., 2013b. Limit cycles and update digraphs in Boolean networks. Discrete Applied Mathematics 161, 1–2.

Bang-Jensen, J., Gutin, G., 2007. RDigraphs Theory, Algorithms and Applications. Springer-Verlag, Berlin.

Berman, A., Plemmons, R. J., 1994. Nonnegative matrices in the mathematical sciences. Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

Bornholdt, S., 2008. Boolean network models of cellular regulation: prospects and limitations. Journal of the Royal Society Interface 5 (Suppl 1), S85–S94.

Brualdi, R. A., Ryser, H. J., 1991. Combinatorial matrix theory. Cambridge University Press, Cambridge.

Chaves, M., Réka, A., Sontag, E., 2005. Robustness and fragility of Boolean models for genetic regulatory networks. Journal of Theoretical Biology 235, 431–449.

Chaves, M., Tournier, L., Gouzé, J., 2010. Comparing Boolean and piecewise affine differential models for genetic networks. Acta Biotheorica 58, 217–232.

Christoph Schmal, T. P. P., Drossel, B., 2010. Boolean networks with robust and reliable trajectories. New Journal of Physics 12, 113054.

Ciliberti, S., Martin, O., Wagner, A., 2007. Robustness can evolve gradually in complex regulatory gene networks with varying topology. PLoS Comput. Biol. 3, e17.

Davidich, M., Bornholdt, S., 2008. Boolean model predicts cell cycle sequence of fission yeast. PLos ONE 3, e1672.

Demongeot, J., Elena, A., Sené, S., 2008. Robustness in regulatory networks: a multi-disciplinary approach. Acta Biotheoretica 56 (1-2), 27–49.

Elena, A., 2009. Robustesse des réseaux d'automates booleéns a seuil aux modes d'itération. Application a la modélisation des réseaux de régulation génétique, PhD thesis. Université Joseph Fourier (Grenoble I), Grenoble, France.

Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D., 2006. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. Bioinformatics 22, 124–131.

Fogelman, F., Goles, E., , Pellegrin, D., 1985. Decreasing energy functions as a tool for studying threshold networks. Discrete Applied Mathematics 12, 261–277.

Goles, E., 1980. Comportement oscillatoire d'une famille d'automates cellulaires non uniformes. These docteur ingénieur. IMAG, Université de Grenoble, Grenoble, France.

Goles, E., Montalva, M., Ruz, G., 2013. Deconstruction and dynamical robustness of regulatory networks: Application to the yeast cell cycle networks. Bull Math Biol 75, 939–966.

Goles, E., Noual, M., 2010. Block-sequential update schedules and Boolean automata circuits. DMTCS Proceedings (01), 41–50.

Goles, E., Noual, M., 2012. Disjunctive networks and update schedules. Advances in Applied Mathematics 48, 646–662.

Goles, E., Salinas, L., 2008. Comparison between parallel and serial dynamics of Boolean networks. Theoretical Computer Science 396, 247–253.

Goles, E., Salinas, L., 2010. Sequential operator for filtering cycles in boolean networks. Advances in Applied Mathematics 45 (3), 346–358.

Goles, S., Matamala, M., 1993. Complexity of block-sequential update for symmetric neural networks. In: Proceedings of 1993 International Joint Conference on Neural Networks. IEEExplore, Nagoya, pp. 1469–1472.

Gómez, L., 2009. Robustez de ciclos dinámicos en redes Booleanas, Engineering thesis. Universidad de Concepción, Concepción, Chile.

Green, D. G., Leishman, T. G., Sadedin, S., 2007. The emergence of social consensus in Boolean networks. In: Artificial Life, 2007. ALIFE'07. IEEE Symposium on. IEEE, pp. 402–408.

Greil, F., Drossel, B., Sattler, J., 2007. Critical kauffman networks under deterministic asynchronous update. New Journal of Physics 9 (10), 373.

Hansson, A., Mortveit, H., Reidys, C., 2005. On asynchronous cellular automata. Advances in Complex Systems 8 (4), 521–538.

Huang, S., 1999. Gene expression profiling, genetic networks and cellular states: an integrating concept for tumorigenesis and drug discovery. Journal of Molecular Medicine 77, 469–480.

Jarrah, A. S., Laubenbacher, R., Veliz-Cuba, A., 2010. The dynamics of conjunctive and disjunctive Boolean networks. Bulletin of Mathematical Biology 72, 1425–1447.

Just, W., 2006. The steady state system problem is np-hard even for monotone quadratic Boolean dynamical systems. pre-print.

Kauffman, S., 1969. Metabolic stability and epigenesis in randomly connected nets. Journal of Theoretical Biology 22, 437–67.

Kauffman, S., 1990. Requirements for evolvability in complex systems: orderly dynamics and frozen components. Physica D: Nonlinear Phenomena 42, 135–152.

Kauffman, S., 1993. The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, New York.

Macauley, M., Mortveit, H. S., 2009. Cycle equivalence of graph dynamical systems. Nonlinearity 22 (2), 421.

Mendoza, L., Alvarez-Buylla, E., 1998. Dynamics of the genetic regulatory network for arabidopsis thaliana flower morphogenesis. Journal of Theoretical Biology 193, 307–319.

Meng, M., Feng, J., 2014. Function perturbations in Boolean networks with its application in a d. melanogaster gene network. European Journal of Control 20 (2), 87 – 94.
URL http://www.sciencedirect.com/science/article/pii/S0947358014000028

Mortveit, H., Reidys, C., 2001. Discrete, sequential dynamical systems. Discrete Mathematics 226, 281–295.

Noual, M., 2011. Dynamics in parallel of double Boolean automata circuits. Tech. rep., LIP, École Normale Supérieure de Lyon.

Robert, F., 1986. Discrete Iterations: A Metric Study. Springer-Verlag, Berlin.

Robert, F., 1995. Les Systèmes Dynamiques Discrets. Mathématiques et Applications. Springer.
URL http://books.google.cl/books?id=Oztl9R30BW8C

Ruz, G., Goles, E., Montalva, M., Fogel, G., 2014. Dynamical and topological robustness of the mammalian cell cycle network: A reverse engineering approach. BioSystems 115, 23–32.

Ruz, G. A., Goles, E., 2013. Learning gene regulatory networks using the bees algorithm. Neural Computing and Applications 22 (1), 63–70.

Salinas, L., 2008. Estudio de modelos discretos: estructura y dinámica, PhD thesis. Universidad de Chile, Santiago, Chile.

Schaefer, T. J., 1978. The complexity of satisfiability problems. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing. STOC '78. ACM, New York, NY, USA, pp. 216–226.
URL http://doi.acm.org/10.1145/800133.804350

Schutter, B. D., Moor, B. D., 2000. On the sequence of consecutive powers of a matrix in a Boolean algebra. SIAM Journal on Matrix Analysis and Applications 21, 328–354.

Shmulevich, I., Lähdesmäki, H., Dougherty, E., Astola, J., Zhang, W., 2003. The role of certain post classes in Boolean network models of genetic networks. Proceedings of the National Academy od Sciences USA 100, 10734–10739.

Shmulevich, I., Saarinen, A., Yli-Harja, O., Astola, J., 2002. Inference of genetic regulatory networks via best-fit extensions. In: Computational and Statistical Approaches to Genomics. Springer, pp. 197–210.

Thomas, R., 1973. Boolean formalization of genetic control circuits. Journal of Theoretical Biology 42, 563–585.

Tocci, R., Widmer, N., 2001. Digital Systems: Principles and Applications, seventh Edition. Prentice-Hall.

Veliz-Cuba, A., Stigler, B., 2011. Boolean models can explain bistability in the lac operon. J. Comput. Biol. 18, 783–794.