FACULTY OF ENGINEERING AND SCIENCES
PHD. IN COMPLEX SYSTEMS ENGINEERING

On the Complexity of Predicting Majority and Sandpile Cellular Automata

PABLO CONCHA-VEGA

Thesis submitted to the Faculty of Engineering and Sciences of Adolfo Ibáñez University to qualify for the academic degree of Doctor of Philosophy in Complex System Engineering

DIRECTORS: PEDRO MONTEALEGRE, ERIC GOLES

CO-DIRECTOR: KÉVIN PERROT

June 2024
Santiago, Chile

Aix*Marseille université
Socialement engagée

UNIVERSIDAD ADOLFO IBÁÑEZ

NNT : 2023AIXM0001

THÈSE DE DOCTORAT

Soutenue à Universidad Adolfo Ibáñez en cotutelle avec Aix-Marseille Université

le 12 juin 2024 par

Pablo CONCHA-VEGA

On the Complexity of Predicting Majority and Sandpile Cellular Automata

Discipline
Informatique

École doctorale
ED 184 Mathématiques et informatique
Doctorado en ingeniería de sistemas complejos

Laboratoire
Laboratoire d'informatique et des systèmes (LIS)
Facultad de ingeniería y ciencias

Composition du jury

Enrico FORMENTI PU, Université Côte d'Azur	Examinateur
Eric GOLES Professeur titulaire, Universidad Adolfo Ibáñez	Co-directeur
Luca MANZONI Professeur associé, University of Trieste	Rapporteur
Marco MONTALVA Professeur associé, Universidad Adolfo Ibáñez	Examinateur
Kévin PERROT MCF HDR, Aix Marseille Université	Directeur
Gonzalo RUZ Professeur titulaire, Universidad Adolfo Ibáñez	Président du jury
Sylvain SENÉ PU, Aix Marseille Université	Examinateur
Véronique TERRIER MCF HDR, Université de Caen Normandie	Rapporteure

Invités

Pedro MONTEALEGRE Professeur associé, Universidad Adolfo Ibáñez	Co-encadrant
Guillaume THEYSSIER CR, CNRS, I2M	Examinateur

LIS
LABORATOIRE D'INFORMATIQUE & DES SYSTÈMES
UMR 7020

FACULTAD DE INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

Affidavit

I, undersigned, Pablo Concha Vega, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific supervision of Eric Goles, Pedro Montealegre and Kévin Perrot, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the french national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Santiago, Chile, the 11th of March 2024.

Publications

Journals:

1. Concha-Vega, P., Goles, E., Montealegre, P., & Ríos-Wilson, M. (2022). On the complexity of stable and biased majority. Mathematics, 10(18), 3408.

Preprints:

1. Concha-Vega, P., Goles, E., Montealegre, P., & Perrot, K. (2024). Sandpiles prediction and crossover on $\mathbb{Z}^2$ within Moore neighborhood. https://www.researchsquare.com/article/rs-3872054/latest.pdf

2. Concha-Vega, P., Goles, E., Montealegre, P., & Perrot, K. (2024). Sandpiles timed prediction and crossover on $\mathbb{Z}^2$ within Moore neighborhood.

3. Concha-Vega, P., Goles, E., Montealegre, P., & Perrot, K. (2024). On the Complexity of the Freezing Majority Rule with L-shaped Neighborhood.

Conferences:

1. Concha-Vega, P., Goles, E., & Montealegre, P., (2022). On the Complexity of the Freezing Majority Rule with L-shaped Neighborhood. Conference on Complex Systems.

Résumé et mots clés

Un automate cellulaire est composé d'une grille de cellules, chacune ayant un état (issu d'un ensemble fini) qui évolue selon une règle locale (homogène dans l'espace et le temps).

Cette thèse se concentre sur deux classes particulières d'automates cellulaires : les automates cellulaires majoritaires et le modèle de pile de sable. Ces deux classes d'automates cellulaires sont liées par plusieurs aspects théoriques discutés dans ce travail. Notre attention se porte sur un problème de décision connu sous le nom de *problème de prédiction*, consistant à déterminer si une cellule donnée change d'état au cours de la simulation d'un automate cellulaire. La complexité algorithmique de ce problème offre des indications précieuses sur les algorithmes les plus efficaces pour calculer la dynamique de l'automate. Pour le cas bidimensionnel, la complexité du problème de prédiction demeure une question ouverte tant pour les automates cellulaires majoritaires que pour les piles de sable.

La principale contribution de cette thèse réside dans la classification systématique des variations des automates cellulaires majoritaires et du modèle de pile de sable, en fonction de la complexité algorithmique de leurs problèmes de prédiction. Notre première approche implique la fusion de deux règles majoritaires distinctes, stables et biaisées, donnant naissance à ce que nous appelons des automates cellulaires majoritaires hétérogènes. Nous prouvons que, pour les automates cellulaires majoritaires hétérogènes unidimensionnels, le problème de prédiction est dans la classe de complexité NL, mais devient P-complet pour des dimensions supérieures.

Ensuite, nous examinons une variante appelée majorité gelée et introduisons le concept de voisinages en forme de L. Notamment, Nous démontrons que, pour le plus petit voisinage en forme de L, le problème de prédiction est dans la classe de complexité NC. En revanche, pour tout voisinage plus grand, le problème de prédiction devient P-Complet.

Enfin, nous étudions le problème de prédiction pour les piles de sable pour chaque sous-voisinage du voisinage de Moore. Nous prouvons que 12 d'entre eux ont un problème de prédiction P-complet. Pour les autres, nous prouvons qu'ils ne peuvent pas croiser l'information, si le bit d'information est la présence (ou l'absence) d'une avalanche. Nous introduisons également le concept de problème de prédiction chronométré, une variation du problème de prédiction canonique où un pas de temps précis fait partie de l'entrée. Nos recherches révèlent que le problème de prédiction chronométré est P-complet pour 52 voisinages différents.

Mots clés: Automates cellulaires majoritaires, modèle de pile de sable, prédiction, systèmes dynamiques discrets, complexité algorithmique.

Resumen y palabras clave

Un automata celular consiste en una grilla de celdas, cada una con un estado (de un conjunto finito) que evoluciona según una regla local (homogénea en espacio y tiempo).

Esta tesis se centra en dos clases de autómatas celulares: los Autómatas Celulares de Mayoría y el Modelo de Pila de Arena, que están relacionados por varios aspectos teóricos discutidos aquí. Abordamos el *problema de predicción*, que busca determinar si una celda cambia de estado durante la simulación. La complejidad de este problema proporciona información valiosa sobre los algoritmos eficientes para calcular la dinámica del autómata, siendo un problema abierto en el caso bidimensional tanto para los Autómatas Celulares de Mayoría como para la Pila de Arena..

La contribución principal de esta tesis radica en la clasificación sistemática de variaciones del Autómata Celular de Mayoría y del modelo de pila de arena, según la complejidad de sus problemas de predicción. Específicamente, nuestra primera aproximación implica la combinación de dos reglas de mayoría distintas, denominadas estable y sesgada, dando lugar a lo que llamamos Autómatas Celulares de Mayoría Heterogéneos. Demostramos que, para los Autómatas Celulares de Mayoría Heterogéneos unidimensionales, el problema de predicción está en la clase de complejidad NL. Sin embargo, se vuelve P-completo para dimensiones mayores.

A continuación, investigamos una variante conocida como la mayoría congelante e introducimos el concepto de vecindades en forma de L. Demostramos que, para la vecindad en forma de L más pequeña, el problema de predicción está en la clase de complejidad NC. Por el contrario, para cualquier vecindad en forma de L de mayor tamaño, el problema de predicción se vuelve P-completo.

Finalmente, estudiamos el problema de predicción para pilas de arena para cada subvecindad de la vecindad de Moore. Demostramos que 12 de ellas tienen un problema de predicción P-completo. Para el resto, demostramos que no pueden cruzar información, si el bit de información es la presencia (o ausencia) de una avalancha. También, introducimos el problema de predicción cronometrado, una variación del problema de predicción donde un paso de tiempo particular es parte de la entrada. Nuestra investigación revela que el problema de predicción cronometrado es P-completo para 52 vecindades, lo que sugiere que algunas vecindades pueden simular eficientemente la evaluación de circuitos solo en su dinámica transitoria.

Palabras clave: automatas celulares de mayoría, modelo de pilas de arena, predicción, sistemas dinámicos discretos, complejidad computacional.

Abstract and keywords

A cellular automaton consists of a grid of cells, each of which has a state (from a finite set) that evolves following a local rule (homogeneous in space and time).

In this thesis we center our attention on two particular classes of cellular automata: The Majority Cellular Automata and the Sandpile Model. These two classes of cellular automata are related by several theoretical aspects discussed in this work. We focus on a decision problem known in the literature as the *prediction problem*. This problem consist in determine whether a given cell of a cellular automaton changes it state during it simulation. Depending on the computational complexity of solving this problem, it offers valuable insights into the most efficient algorithms that can be employed for computing the automaton dynamics. For the 2-dimensional case, to determine the complexity of the prediction problem remains an open problem for both the Majority Cellular Automata and Sandpile.

The key contribution of this thesis lies in the systematic classification of variations within both the Majority Cellular Automaton and the sandpile model, contingent upon the computational complexity of their prediction problems. Specifically, our first approach involves mixing two majority rules herein referred to as stable and biased, resulting in what we term Heterogeneous Majority Cellular Automata. We prove that for the 1-dimensional Heterogeneous Majority Cellular Automata, the prediction problem is in NL. However it becomes P-complete for greater dimensions.

Secondly, we investigate a variant known as the freezing majority and we introduce the concept of L-shaped neighborhoods. Notably, we demonstrate that for the smallest L-shaped neighborhood the prediction problem resides within the complexity class NC. Conversely, for any L-shaped neighborhood of a larger size, the prediction problem becomes P-Complete.

Finally, we study the prediction problem for sandpiles for every sub-neighborhood of the Moore neighborhood. We prove that 12 of them have a P-complete prediction problem. For the rest of them, we prove that they cannot cross information, if the bit of information is the presence (or absence) of an avalanche. We go further by introducing the concept of timed prediction problem, a variation of the canonical prediction problem where a precise time step is part of the input. Our research reveals that the timed prediction problem is P-complete for 52 different neighborhoods, suggesting that some neighborhoods may efficiently simulate circuit evaluation only in their transient dynamics.

Keywords: Majority cellular automata, sandpile models, prediction, discrete dynamical systems, computational complexity.

Agradecimientos

Quiero comenzar expresando mi profundo agradecimiento a mis padres, cuyo apoyo inquebrantable ha sido el pilar de cada etapa de mi vida, incluida esta. Su constante respaldo ha sido invaluable.

También deseo extender mi gratitud a los miembros de mi familia que siempre han estado presentes cuando los he necesitado. Su apoyo ha sido un sostén fundamental en mi camino.

A mis amigos, tanto los de larga data como los más recientes, les agradezco de todo corazón. Cada uno de ustedes ha dejado una huella significativa en mi vida, aportando con su presencia y apoyo en diferentes aspectos.

Quiero reconocer especialmente a mis directores de tesis, Pedro y Eric, quienes desde el principio han mostrado un compromiso excepcional con mi trabajo. Je tiens également à remercier Kévin, pour sa grande volonté et son vif intérêt pour notre recherche.

No puedo pasar por alto a aquellas personas que, aunque ya no están tan cerca como antes, han dejado una marca en mi vida. A todos ellos les envío mi sincero agradecimiento.

Por último, quiero expresar mi gratitud a los proyectos FONDECYT 1230599, dirigido por Pedro Montealegre, y FONDECYT-ANID 1200006, dirigido por Eric Goles, que han financiado parcialmente este trabajo.

Contents

List of Figures

Introduction

In the field of computer science, the study of discrete dynamical systems has emerged as a fascinating tool that offers a unique lens through which we can understand complex phenomena. One of the most iconic and influential branches of these systems, characterized by their discrete space and time, is cellular automata. Cellular automata are simple yet powerful mathematical constructs that have captured the imagination of computer scientists, mathematicians, and physicists for decades. A cellular automaton consists of a grid of cells, each of which has a state (from a finite set) that evolves following a local rule (homogeneous in space and time). The study of discrete dynamical systems, and specially cellular automata, has not only enriched our understanding of fundamental computational principles but has also paved the way for the development of innovative models and applications across various domains AGUR et al. 2002a; C. CASTELLANO et al. 2009; R. HEGSELMANN 1998; KŮRKA 1997; ROY et al. 2021; T. C. SCHELLING 1978.

In this thesis we center our attention on two particular classes of cellular automata : The Majority Cellular Automata and the Sandpile Cellular Automata. The Majority Cellular Automata (MCA) can be defined as two-states cellular automata (namely with states -1 and $+1$), where in each step, each site takes the most represented state in its neighborhood. The majority rule is one of the simplest defined rules, and appears naturally in models from physics, biology, social phenomena, and elections systems Thomas C SCHELLING 2006; C. CASTELLANO et al. 2009; Rainer HEGSELMANN 1998; ANCONA et al. 2022; AGUR et al. 2002a; YASSINE 2012. On the other hand, the sandpile CA, originally introduced by Bak, Tang and Wiesenfeld as the first example of a dynamical system exhibiting self-organized criticality BAK et al. 1987, simulates the behavior of sand grains on a 2-dimensional grid. Each grid cell has a capacity threshold. When a cell accumulates more sand grains than its capacity, it becomes *unstable* and redistributes some of its grains to its neighboring cells. This redistribution process, often referred to as an *avalanche*, occurs iteratively, with unstable cells toppling and causing their neighbors to become unstable in a cascading fashion. Although these models are usually formulated over 2-dimensional lattices with the von Neumann neighborhood (the cells surrounding a cell in the four cardinal directions), they can be naturally extended to arbitrary graph topologies, hence to d-dimensional grids. The remarkable feature of these automata over arbitrary graphs is its Turing universality E. GOLES et MARGENSTERN 1997; Eric GOLES et Pedro MONTEALEGRE 2014. This means that it has the capacity to simulate any given Turing machine, a foundational concept in computer science representing a general-purpose computing device. Knowing this, a pivotal question in the field of computer science arose, as articulated by Moore and Nilsson C. MOORE et al. 1999 for the sandpile CA : *given an initial distribution of*

grains in a sandpile, what is the eventual state it converges to after all possible topplings have occurred? This problem, commonly referred to as the *prediction problem* for sandpiles, serves as a cornerstone in the study of sandpile dynamics. As one might envision, there exists a variant of the prediction problem for the MCA initially studied by Cristopher MOORE 1997. Depending on the computational complexity of solving this problem, it offers valuable insights into the most efficient algorithms that can be employed for computing the automaton dynamics. The initial results of Moore and Nilsson unveil fundamental differences in the computational capabilities of both the MCA and the sandpile CA depending on dimensionality. They demonstrated that in three or more dimensions, the dynamics is intrinsically sequential (P-complete), implying that solving the prediction problem inherently requires a naive simulation approach (unless P = NC). On the other hand, in the one-dimensional case, they presented a fast parallel algorithm (NC). Determining the computational complexity of the 2-dimensional cases are still open problems today. This dimensionality-sensitive results also applies for the *biased majority automata* (called *Half-or-More automata* ibid.) which corresponds to the majority automata where the sites do not consider their own state in the neighborhood, and privilege state +1 in tie cases.

When establishing the P-completeness of the prediction problem associated with a cellular automaton, the prevailing framework in the literature is the well-known *Banks approach* BANKS 1971. This approach hinges on reducing from the *monotone circuit value problem* (**MCVP**), via constructing a series of circuitry gadgets. These gadgets typically consist of *wires* and *logic gates* (AND and OR). Nevertheless, when the support of the dynamics is symmetric and planar, it may be necessary to design more devices, such as *diodes* and *crossover gates* E. GOLES, P. MONTEALEGRE, K. PERROT et G. THEYSSIER 2017; E. GOLES, TSOMPANAS et al. 2020; MODANESE et al. 2022. Crossover gates, in particular, hold a significant role as they often present the greatest challenge in terms of construction in a 2-dimensional context. In essence, crossover gates consist of a rectangular finite configuration that facilitates the unimpeded transmission of information (commonly referred to as a *signal*) from one edge to the opposite (e.g., from north to south), without interfering with information traveling in the perpendicular direction (e.g., from west to east). The underlying challenge in the context of 2-dimensional MCA and sandpile models is the absence of an evident construction of a crossover gate. Furthermore, the work of Goles and Gajardo GAJARDO et al. 2006 has sustained this difficulty, proving that a crossover gate cannot exist for the 2-dimensional sandpile, whether one considers the von Neumann neighborhood or the Moore neighborhood (the eight adjacent cells surrounding a cell). There is no equivalent result for the MCA. It is crucial to emphasize that the impossibility to construct a crossover gate solely signifies the limitation of the Banks approach, therefore the classification of the prediction problem remains unsolved, as we mentioned before.

The key contribution of this thesis lies in the systematic classification of variations within both the Majority Cellular Automaton (MCA) and the sandpile CA, contingent upon the computational complexity of their prediction problems.

Thesis Outline

This thesis is structured into five Chapters. Chapter 1 contains primary definitions used throughout this work. It introduces Cellular Automata, focusing on the variants studied in this thesis, *i.e.*, the Majority Cellular Automata and the sandpile CA. The Chapter also delves into the essential elements of computational complexity theory for understanding the ensuing results, including the definition of prediction and timed prediction problems. Additionally, it provides a *toolbox* containing existing techniques sourced from the literature.

In Chapter 2, we expose and overview of existing results in the literature regarding the Majority Cellular Automata, the Sanpile Model and their various adaptations. Our work is is line with a series of previous works that aim to understand the computational complexity of both, the majority rule and the sandpile CA. This includes findings on freezing dynamics, generalization to arbitrary topologies, and the interrelations between the Majority Cellular Automata and the sandpile CA. There are several variations of these two models that have NC, P-complete or even NP-complete prediction problems.

Our contributions start at Chapter 3. Here we unveil our results concerning the variations proposed of the Majority Cellular Automata. Specifically, our initial approach involves mixing the canonical majority and the biased majority, resulting in what we term *Heterogeneous Majority Cellular Automata* (HMCA). Significantly, we establish that, for dimension 1, the prediction problem for HMCA falls within the complexity class NL, whereas for higher dimensions, it becomes P-complete. Furthermore, in the context of the MCA, we investigate a variant known as the freezing majority, where a cell in state 1 remains unchanged in every subsequent step. To analyze these cellular automata, we introduce the concept of L-shaped neighborhoods. Notably, we demonstrate that for the conventional L-neighborhood (recognized in literature as the Toom neighborhood TOOM 1980), considering the center site along with the upper and right sites, the prediction problem resides within the complexity class NC. Conversely, for any L-neighborhood of a larger size, the prediction problem becomes P-Complete.

Moving to Chapter 4, we extend notions in relation to the complexity of 2-dimensional sandpiles by meticulously studying all the sub-neighborhoods within the Moore neighborhood, *i.e.*, all the 256 possible combinations of the eight cells included in the Moore neighborhood. Surprisingly, we found that there are 12 neighborhoods that admit a crossover gate. For them, we construct all the necessary circuitry for proving the P-completeness of their prediction problem. In contrast, for the remaining sub-neighborhoods, we extend the findings of Goles and Gajardo by establishing the impossibility of a crossover gate. While some of these proofs present fairly straightforward conclusions, others require deeper analysis and geometric arguments. We extend these results by studying the timed prediction problem for all these subsets. Our research reveals that the timed prediction problem is P-complete for 52 different neighborhoods : despite the fact that most of them do not permit a regular crossover gate, their inner dynamics is provably inherently sequential.

Finally, in Chapter 5 we discuss the outcomes derived from this work and the current state of the art. Additionally, we put forth perspectives on the studied topics to guide future investigations.

1 Preliminaries

1.1 Cellular Automata

Cellular automata are discrete dynamical systems defined on a d-dimensional regular grid of cells, where each cell changes its state by the action of a local function or automata rule, which depends on the state of the cell and the state of its neighbors. In this model, each cell can only have a finite number of states taken from a set Q.

A *configuration* of the grid is a function c that assigns values in Q to every cell of the d-dimensional grid. The value of the cell u in the configuration c is denoted as c_u. A *periodic configuration* is a configuration defined by a repeating d-dimensional pattern of cell states in the shape of a d-cube of side n, which is repeated all over de d-dimensional grid. For instance, if $d = 2$, a periodic configuration of the two-dimensional grid is given by a square area of $n \times n$ cells.

Let $\subset_{\texttt{fin}}$ denote finite subsets. For every cell $u \in \mathbb{Z}^d$ we refer as $\mathcal{N}(u) \subset_{\texttt{fin}} \mathbb{Z}^d$ to the neighborhood of u. In addition, for a cell $u \in \mathbb{Z}^d$, we call $c_{\mathcal{N}(u)}$ the restriction of c to the neighborhood of u. For a cellular automaton, the size of the neighborhood of a cell is uniform, i.e. $|\mathcal{N}(u)|$ is the same for each cell. Morever, we have that $\mathcal{N}(u) = \mathcal{N}(0) + u$. We consider $\mathcal{N} = \mathcal{N}(0)$. The most famous neighborhoods, for their simplicity and symmetries, are *von Neumann* and *Moore*, respectively defined for 2-dimensional grids as :

$$\mathcal{N}_{\texttt{vn}} = \{(x, y) \in \mathbb{Z}^2 \mid |x| + |y| = 1\} \quad \text{and} \quad \mathcal{N}_{\texttt{m}} = \{(x, y) \in \mathbb{Z}^2 \mid 1 \leq x^2 + y^2 \leq 2\}.$$

Formally, a *cellular automaton* (CA) with states Q and *local function* $f : Q^{|\mathcal{N}(u)|} \to Q$, is a map $F : Q^{n^d} \to Q^{n^d}$, such that $F(c)_u = f(c_{\mathcal{N}(u)})$, for all $u \in \mathbb{Z}^d$. We call F the *global function* or the *global rule* of the CA. The dynamic is defined by assigning to each state of the configuration c a new state given by the synchronous update of the local function on c. In addition, it is also possible to define a function F by assigning to each cell a local function. In this case, it is possible to assign different rules to each cell.

1.1.1 Majority Cellular Automata

Great part of this work is based on the *Majority Cellular Automata* (MCA). In this cellular automaton, each cell has one of two possible internal states (usually represented by $\{-1, 1\}$). Each cell will change its state according to a majority local rule. In each time step, this local rule forces each cell to take the most represented state in its neighborhood. Depending on the neighborhood, there may be an even number of neighbors, therefore there are some cases in which exactly one half of the neighbors

are in one state and the other half are in the other. In this *tie* case scenario, the rule is not a priori defined. However, in the literature, different approaches are considered to define transitions in tie case scenarios.

The first case is the one known as *stable* majority, in which, in the tie case scenario, the rule preserves the original state of the cell. So, for example, if a cell is in state 1 and there is a tie case scenario, it will remain in the state 1 (see Figure 1.1 for an example). Formally, the stable majority local rule is defined by :

$$f(c)_u = \begin{cases} 1 & \text{if } \sum_{v \in \mathcal{N}(u)} c_v > 0, \\ c_u & \text{if } \sum_{v \in \mathcal{N}(u)} c_v = 0, \\ -1 & \text{if } \sum_{v \in \mathcal{N}(u)} c_v < 0. \end{cases}$$

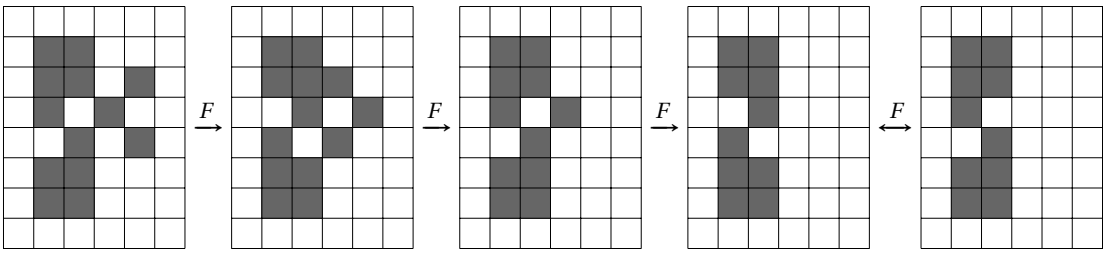


FIGURE 1.1 : Example of majority cellular automata with the von Neumman neighborhood $\mathcal{N}_{\mathtt{vn}}$. Gray cells represent the $+1$ state, while white cells are in the -1 state. A cell in state -1 (resp. $+1$) changes to $+1$ (resp. -1) only if it has at least three neighbors in state $+1$ (resp. -1). Notice that the configuration illustrated in the last image changes to the one on its left, then the initial configuration converges to a limit-cycle of length 2.

The second case that is considered in this work is the one of the *biased* majority in which in a tie case scenario a cell changes to a fixed state. Formally, for $s \in \{-1, 1\}$, the s-biased majority local rule is defined by :

$$f^s(c)_u = \begin{cases} 1 & \text{if } \sum_{v \in \mathcal{N}(u)} c_v > 0, \\ s & \text{if } \sum_{v \in \mathcal{N}(u)} c_v = 0, \\ -1 & \text{if } \sum_{v \in \mathcal{N}(u)} c_v < 0. \end{cases}$$

From now on, we call the 1-biased majority rule simply as biased majority rule. In general terms, a d-dimensional *heterogeneous automata* corresponds to a function $F : \{-1,1\}^{\mathbb{Z}^d} \mapsto \{-1,1\}^{\mathbb{Z}^d}$ together with an assignation of local functions $g : \mathbb{Z}^d \to \{f_1, \ldots, f_k\}$ such that $F(c)_u = f_{g(u)}(c_{\mathcal{N}(u)})$, *i.e.*, each cell is alway governed by a particular local function from $\{f_1, \ldots, f_k\}$. Thus, the d-dimensional *heterogeneous majority automata* is a d-dimensional heterogeneous automata where the set of available local functions

corresponds to the majority and biased majorities ($\{f, f^1\}$).

1.1.2 Freezing Majority Cellular Automata

There is widely studied class of cellular automata called *Freezing Cellular Automata*. In a *freezing* dynamic, states can only increase according to a given order. For example, in the particular case of the binary states $Q = \{-1, 1\}$ (or $Q = \{0, 1\}$), there is no transition from 1 to -1 (resp. 0), *i.e.*, a cell that reaches state 1 stays frozen.

Formally, the *Freezing Majority Cellular Automata* (FMCA) are defined by the local function

$$f(c)_u = \begin{cases} 1 & \text{if } x_u = 1 \text{ or } \sum_{v \in \mathcal{N}(u)} x_v > 0 \\ -1 & \text{otherwise.} \end{cases}$$

1.1.3 Sandpile Cellular Automata

As its name suggests a *sandpile cellular automaton* simulates the behavior of sand grains falling down on a d-dimensional grid. A *sandpile cellular automaton* is defined by a *neighborhood* $\mathcal{N} \subset_{\mathtt{fin}} \mathbb{Z}^d$. We assume this neighborhood $\mathcal{N}$ spans $\mathbb{Z}^d$, otherwise we have independent subdynamics. We denote its size by $\theta_{\mathcal{N}} = |\mathcal{N}|$. A *configuration* $c : \mathbb{Z}^d \to \mathbb{N}$ assigns $c(u)$ grains to cell $u \in \mathbb{Z}^d$. Let $\mathtt{H}(\alpha) = 1$ if $\alpha \geq 0$ and $\mathtt{H}(\alpha) = 0$ if $\alpha < 0$ denote the Heaviside step function, then the sandpile dynamics $F_{\mathcal{N}} : \mathbb{N}^{\mathbb{Z}^d} \to \mathbb{N}^{\mathbb{Z}^d}$ is defined as :

$$\forall u \in \mathbb{Z}^d : F_{\mathcal{N}}(c)(u) = c(u) - \theta_{\mathcal{N}} \cdot \mathtt{H}(c(u) - \theta_{\mathcal{N}}) + \sum_{v \in \mathcal{N}} \mathtt{H}(c(u - v) - \theta_{\mathcal{N}}) .$$

In words, each cell reaching the threshold $\theta_{\mathcal{N}}$ *topples* or *fires*, sending one grain to each out-neighbor which relative positions are given by $\mathcal{N}$. This local rule is applied in parallel at every cell of the lattice (See an example on Figure 1.2). However, it is worth mentioning that even in the case of applying a sequential update schedule, the sandpile CA always converges to the same stable configuration, *i.e.*, the final configuration is independent of the order in which the grain topplings occur. This was demonstrated by Dhar 1990 and it is commonly refeared as the *Abelian* property of sandpiles. This is the reason why the sandpile CA is widely known as the *Abelian Sandpile Model*.

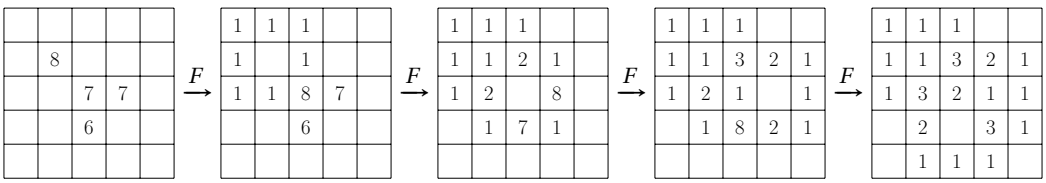


FIGURE 1.2 : Example of sandpile dynamics for Moore neighborhood $\mathcal{N}_{\mathtt{m}}$. The numbers indicate the amount of grains in every cell. A blank site represents an empty cell.

When it is clear from the context, we drop the subscript notations and simply denote θ and F the threshold and the evolution rule. The quantity of sand grains in a configuration is denoted $\mathscr{G}(c) = \sum_{u \in \mathbb{Z}^d} c(u)$, it is invariant by F for any sandpile CA : $\mathscr{G}(c) = \mathscr{G}(F(c))$. A configuration c is *finite* when it contains a finite number of sand grains ($\mathscr{G}(c) \in \mathbb{N}$), and *stable* when no toppling occurs ($\forall u \in \mathbb{Z}^d : c(u) < \theta$). From any finite configuration c, the dynamics converges to a stable configuration denoted $c^\circ = \lim_{t \to \infty} F^t(c)$, see for example FORMENTI et K. PERROT 2019. It is worth mentioning that all the results presented in this thesis pertain to 2-dimensional sandpile CA.

1.2 Computational Complexity Theory

In this section we briefly define the main concepts of computational complexity theory used in this work. However, we encourage the interested reader to consult GREENLAW et al. 1995 for more detailed insights in the topic.

As this section functions as an introduction to the computational complexity concepts used *in this thesis*, we only focus in the P complexity class and its subclasses.

1.2.1 Decision Problems

In the field of computer science, a *decision problem* is roughly a question with a binary response, denoted either as 1 (YES) or 0 (NO). If a decision problem can be solved by a *Turing machine*, it is termed *decidable*, otherwise it is *undecidable*. Clearly, determining whether a decision problem is decidable or undecidable requires a formal proof. *Decidability* holds tremendous importance in the field of computer science. Proofs for demonstrating that a decision problem is decidable usually consist in proposing an algorithm that solves it. The latter leads to a natural categorization of the decidible problems, which will be further addressed in the following subsection.

1.2.2 Complexity Classes

Given a decidable problem, its complexity can be defined as the amount of resources, like time or space, needed to solve it as a function of the size of the input, which is usually denoted by n. One fundamental set of computational decision problems is the class P, which is the class of problems solvable in polynomial time ($\mathcal{O}(n^i)$ for some constant i) on a deterministic Turing machine. The class P is informally known as the class of problems that admit an efficient algorithm. However, within P there exist other complexity classes that denote problems solvable with even greater efficiency. One of these classes is the NC class, which contains the problems solvable in poly-logarithmic time by a Parallel Random Access Machine (PRAM) (see JÁJÁ 1992 for a detailed definition), with a polynomial number of processors. In other words, NC is the class of problems which have a fast parallel algorithm. One can be more specific and define the NC^i class as the subset of NC problems that can be solved in $\mathcal{O}((\log n)^i)$ parallel time. Clearly we have :

$$\mathsf{NC}^1 \subseteq \mathsf{NC}^2 \subseteq \cdots \subseteq \mathsf{NC}^i \subseteq \cdots \subseteq \mathsf{NC}$$

Another of these classes are the L and NL classes. The L class consists in the decision problems that can be solved by a deterministic Turing machine in logarithmic space. On the other hand, NL is the non-deterministic version of L. The hierarchy for these complexity classes is the following one :

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{P}$$

Additionally to the complexity classes themselves, there is an important set of problems within each class. Given a complexity class C, a problem is said to be C-complete if

1. it belongs to C and
2. there exists a reduction from any problem in C to this specific problem.

If only the second condition is satisfied, the problem is termed C-hard. The concept of *hardness* is relevant as none of the previously mentioned classes (and including others like NP) are known to be distinct from one another. It is important to mention that the reduction is expected to be efficient in the sense that it should not have a higher complexity than the class C itself. This efficiency criterion ensures that the reduction process aligns with the complexity level inherent to class C.

Throughout this work, we consider logarithmic space many-one reductions when proving P-completeness. Let us consider two decision problems A and B. A logarithmic space many-one reduction from A to B is a logarithmic space computable function f that maps instances of A to instances of B. In this fashion, if we have an efficient algorithm for deciding B (let us say polynomial), then naturally we have an efficient algorithm for deciding A. Thus, reductions provide a method for evaluating the relative computational complexity of two problems (in our example B is at least as hard to solve as A).

A frequently explored concept in the literature is that of *inherently sequential problems*. In simple terms, a problem is called inherently sequential if the time needed to decide it cannot be substantially accelerated, regardless of the number of processors employed. A problem classified in NC is considered *efficiently parallizable*, in contrast to the inherently sequential problems. Conversely, the P-complete problems are the more probable candidates for being inherently sequential.

1.2.3 Circuit Value Problem

The *Circuit Value Problem* (CVP) is a decision problem that Ladner demonstrated to be P-complete shortly after the formulation of Cook-Levin's theorem LADNER 1975. The CVP can be viewed as analogous to the NP-complete problem *Satisfability*. The definitions and results exposed here are taken from GREENLAW et al. 1995; JÁJÁ 1992. CVP asks whether a given Boolean circuit outputs True or False with a given input. A

Boolean circuit is an idealization of real electronic device and, as an abstraction of a real circuit, it ignores a host of important practical considerations such as circuit area, volume, pin limitations, power dissipation, packaging, and signal propagation delay. A circuit is simply a formal model of a combinational logic circuit. It is an acyclic directed graph in which the edges carry unidirectional logical signals and the vertices compute elementary logical functions (and, or, not). The entire graph computes a Boolean function from the inputs to the outputs in a natural way. An example of a circuit is given in Figure 1.3 for a better understanding of the concept.

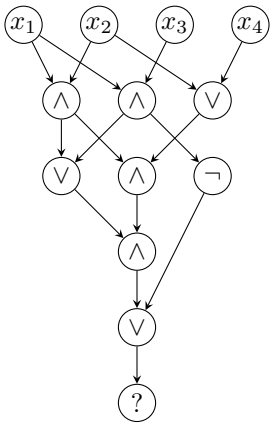


FIGURE 1.3 : Example of a Boolean circuit. The nodes labeled as x_i with $\{1,2,3,4\}$ represent de inputs. The node labeled with a question mark represents the circuit output.

Formally, the CVP is defined as follows :

Circuit Value Problem (**CVP**)
Input:

- An encoding $\bar{\alpha}$ of a Boolean circuit α
- an input $x_1, \ldots, x_n$
- a designated output y

Question: is output y of α True on input $x_1, \ldots, x_n$?

Theorem 1.1 (GREENLAW et al. 1995)**.** *The Circuit Value Problem is* P*-complete.*

The formal proof of this fact is quite technical. However, it consists in simulating an arbitrary Turing machine using Boolean circuits.

In addition to the canonical CVP, there are a multitude of variations of it. One of them, of special importance in this work, is the *Monotone Circuit Value Problem* (MCVP), which consists only in *monotone circuits*, *i.e.*, Boolean circuits composed only of AND and OR gates. The MCVP turns out to be also a P-complete problem.

1.2.4 Prediction Problems

In this thesis, we focus on prediction problems. A *prediction problem* is a decision problem such that one asks if a given cell of a cellular automaton changes its state at some point of the computation. We also define what we call a *timed prediction problem*, where a timestep t is given as part of the problem input and one asks whether the given cell has a different state at timestep t than at the beginning (timestep 0).

Formally, the prediction problem associated with a cellular automaton is defined as follows :

Prediction problem (**PRED**)
Input:

- a finite configuration $c \in \mathbb{N}^{\mathbb{Z}^d}$
- a cell $u \in \mathbb{Z}^d$

Question: does $\exists t \in \mathbb{N} : F^t(c)_u \neq c_u$?

Notice that a global function is not required as an input because the prediction problem is defined for a specific, predetermined class of cellular automata, contingent upon the context. As we mention, we also consider the timed prediction problem, defined formally as :

Timed prediction problem (**TIMED-PRED**)
Input:

- a finite configuration $c \in \mathbb{N}^{\mathbb{Z}^d}$
- a cell $u \in \mathbb{Z}^d$
- a timestep t

Question: does $F^t(c)_u = c_u$?

It is not clear which of these two problems is (computationally) more difficult.

The core of this work is a classification of the prediction problems associated with several instances of the MCA and the sandpile CA into complexity classes.

Evidently, the prediction problem can be solved by simply simulating the dynamics of the cellular automata until reaching an attractor (or for the given number of time-steps in the case of the timed prediction problem). This is called the *trivial algorithm*. In Eric Goles et Olivos 1980 Goles *et al.* showed that the dynamics of the majority automaton reaches an attractor in a number of time-steps that is linear in the size of the configuration. Moreover, the reached attractor is either a fixed-point or a limit cycle of period two. This result implies that the prediction problem associated to a d-dimensional MCA can be solved in polynomial time. Similar results for the sandpile cellular automata are shown in Formenti et K. Perrot 2019, *i.e.*, the prediction

problem associated with a d-dimensional sandpile CA can be solved by applying the global function a polynomial number of times. Therefore, if we aim to solve the prediction problem more efficiently than the trivial algorithm, we would have to classify it in a proper subclass of P.

1.3 Toolbox

1.3.1 The Banks' Approach

When it comes to prove that the prediction problem of a certain cellular automaton is a P-complete problem, the *Banks' approach* BANKS 1971 is widely used, moreover it is the only known approach for proving P-completeness in this context. Roughly, the Banks' approach consists in simulating an arbitrary boolean circuit on the grid by using a series of *gadgets*, constructed specifically for a given cellular automaton, in order to reduce from the MCVP (see Figure 1.4). These gadgets consist in all the necessary circuitry for performing universal computation. The latter usually consists in *wires*, *turns*, AND gates, OR gates, *diodes* and *crossover* gates. It turns out that all gates are usually straightforward to design, except crossover gates. The inexistence of crossover gates is therefore a strong indication that such a reduction may not be possible, but it is relative to a precise definition of crossover gate (to prove that none exist) which enforces a precise (hence restrictive) consideration of what a wire, or more generally a signal is. Providing a general definition of signal is far from obvious DELORME et al. 2002.

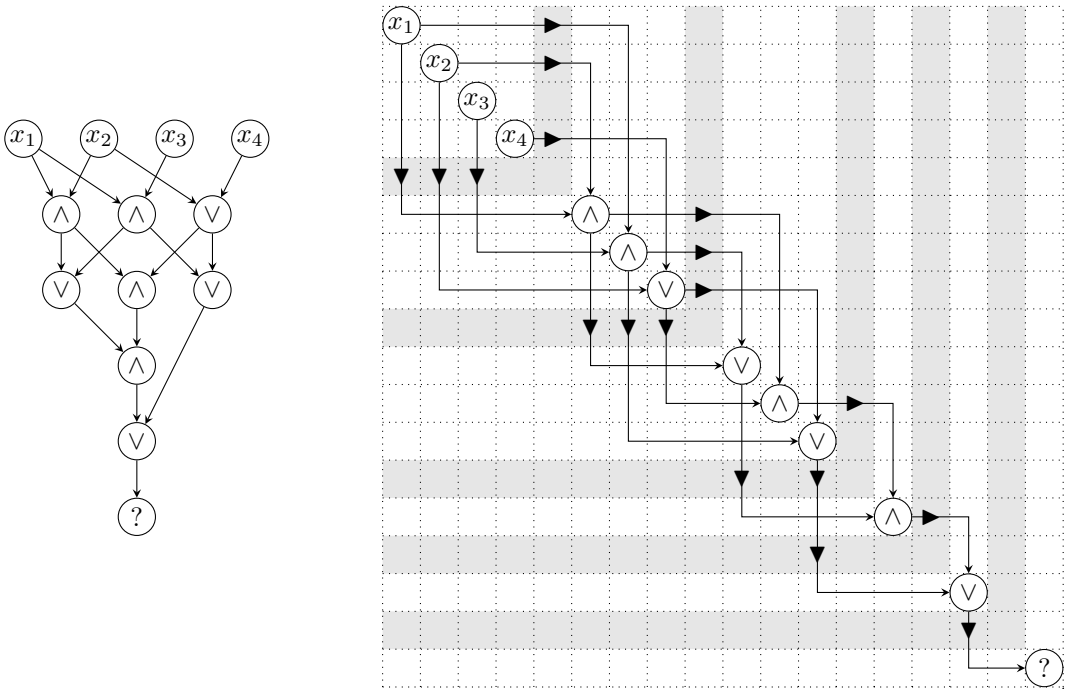


FIGURE 1.4 : Banks' approach. On the left, an illustration of a logic circuit. On the right, a schematic representation of how the Banks' approach is employed to simulate this circuit. Each cell in this grid represents a *gadget*. All the circuits inputs and gates are positioned on the diagonal, connected by *wires*. Crossover gates must be used when two wires intersect. The blank cells represent an stable background.

1.3.2 NC subroutines

As we mention before, the complexity class NC contains all the problems with an efficient parallel solution (polylogarithmic time with a polynomial number of processors). In this subsection we formalize the concept of *parallel algorithm* and mention a few fast parallel algorithms used in this thesis. All the material presented here is taken from GREENLAW et al. 1995 ; JÁJÁ 1992.

A universally accepted model of sequental computing is the *Random Access Machine* (RAM) model, which can be characterized in mainly three components :

1. Memory structure : The memory is divided into individual cells, each of them identified by a unique address. This memory is typically assumed to be large enough to hold the input data and any other variables required by an algorithm.

2. Operations : This model supports basic arithmetic operations (addition, subtraction, multiplication, and division), logical operations (AND, OR, NOT), memory access operations (reading from and writing to memory cells), assignment of a value to a memory cell, conditionnal statements and repetition

structures (loops). Each operation is assumed to take a constant amount of time ($\mathcal{O}(1)$).

3. Sequential execution : Every instruction is executed in a sequential manner one after the other one. There is no notions of parallelism or concurrency.

A natural generalization of the RAM model is the Parallel RAM (PRAM). As it name indicates, it extends the RAM model to include parallelism. The PRAM model assumes that multiple processors operate in parallel, each of them identified by an index i. Each processor has it own local memory, and all processors have access to a shared global memory. Also, all the processors work in a synchronized manner under a general global clock. PRAM is usually categorized based on how processors access the shared memory as follows :

- Exclusive Read Exclusive Write (EREW) : No two processors can access one memory cell simultaneously for reading nor writing operations.
- Concurrent Read Exclusive Write (CREW) : Simoultaneous reading is allowed but only one processor can write on an specyfic memory cell at a time.
- Concurrent Read Concurrent Write (CRCW) : Simoultaneous reading and writing is allowed in this variant. Hence, some method of simoultaneous writing has to be defined (*e.g.* lowest processor index).

The reason why we introduce the PRAM model is because it is closely related to NC. Let us define EREW^i, CREW^i and CRCW^i as the classes of decision problems solvables by the EREW, CREW and CRCW PRAMs, respectively, in $\mathcal{O}((\log n)^i)$ time, using a polynomial number or processors. Then the following holds :

Proposition 1.1. *let* $i \geq 1$, *then*

$$\mathsf{NC}^i \subseteq \mathsf{EREW}^i \subseteq \mathsf{CREW}^i \subseteq \mathsf{CRCW}^i \subseteq \mathsf{NC}^{i+1}$$

This basically means that solving a problem under the PRAM model in polylogarithmic time using a polyomial number of processors, means the problem is in NC. The NC class can also be defined as the set of problems that can be decided by a family of *logspace-uniform circuits* of polylogarithmic depth and a polynomial number of gates. A family of circuits $\{\alpha_n\}$ is termed logspace-uniform if the transformation from 1^n to the α_n circuit description can be done in $\mathcal{O}(\log(\text{size}(\alpha_n)))$ space on a deterministic Turing machine, where $\text{size}(\alpha_n)$ is the number of gates of α_n. Note that this implies that NC can be formulated without using the PRAM classes as follows :

$$\mathsf{NC} = \bigcup_{i \geq 1} \mathsf{NC}^i$$

In fact, as stated in JÁJÁ 1992, the class NC remains invariant across various parallel models, underscoring its fundamental importance in complexity theory.

Later in this work, we employ a pair of NC subroutines, which we articulate in the following propositions.

Proposition 1.2 (Prefix-sum)**.** *The following problem can be solved by a CREW PRAM algorithm with $\mathcal{O}(n)$ processors in $\mathcal{O}(\log n)$ time : given a finite set $A = \{x_1, \ldots, x_n\}$, $k \leq n$ and a binary associative operation $\oplus$ in A, compute $\bigoplus_{i=1}^{k} x_i$.*

It is noteworthy that the elements of A can be any objects, as long as $\oplus$ is an binary associative operation.

Proposition 1.3 (Matrix multiplication)**.** *Let A and B two $n \times n$ matrices. The matrix product $A \times B$ can be calculated by a CRCW PRAM algorithm in $\mathcal{O}(\log n)$ time using $\mathcal{O}(n^2)$ processors.*

This can be extended to rectangular matrices.

For the interested reader, JÁJÁ 1992 contains several parallel techniques that solve non-trivial problems, such as the *pointer jumping* technique.

2 State of the Art

Our work is is line with a series of previous works that aim to understand the computational complexity of both, the majority rule and the sandpile CA, by studying the prediction problem for different variants of these models. We mention these works in this Section as well as the relation between the MCA and the sandpile CA.

As previously discussed, the complexity of the prediction problem can be categorized within P through the use of the trivial algorithm. Nevertheless, more accurate results emerge when studying a grid of dimensions greater than 2. In such instances, the prediction problem is P-complete. Moreover, the prediction problem has been proven to be in NC for the one dimensional case. The latter applies for the Stable MCA, the Biased MCA (both in Cristopher MOORE 1997) and the sandpile CA C. MOORE et al. 1999 considering the von Neumann neighborhood. Note that these results leave the two-dimensional case as an open problem, introducing an intriguing element. This particular case seems to represent a sort of boundary or limit between the P-complete problems and P with its internal complexity classes.

Another interesting fact is shown in Eric GOLES, Pedro MONTEALEGRE et Kévin PERROT 2021, where the authors studied the freezing version of the sandpile CA. The notable outcome of this work in this context is that it gives insights on how to simulate two-dimensional freezing sandpiles using two-dimensional MCA. This establishes a key relationship between these two models in the context of our work and opens several interesting questions : can the majority simulate the canonical sandpile CA? can a sandpile simulate a (freezing) MCA? Are these two classes of CA equivalent in some sense?

2.1 The Majority Rule Variants

Freezing majority. Freezing automata model forest fires KARAFYLLIDIS et al. 1997, infection spreading FUENTES et al. 1999, bootstrap percolation CHALUPA et al. 1979 and voting systems Cristopher MOORE 1997. Theoretical facts about these automata can be seen in Eric GOLES, OLLINGER et al. 2015.

The prediction problem for the *freezing stable majority automata* was studied by Goles et al. in E. GOLES, MONTEALEGRE-BARBA et al. 2013, where the authors show that in two dimensions the prediction problem for the freezing majority automata is in NC. At the same time, in three or more dimensions the prediction problem in P-complete. Later, in E. GOLES, D. MALDONADO et al. 2017; Eric GOLES, Diego MALDONADO et al. 2020 the authors showed an analogous result for the *freezing biased majority automata*. In both articles, the efficient algorithms are based in some topological properties of

the set of *stable* sites, that is to say, the set of nodes initially in -1 that never switch their state. Unfortunately, these topological properties are not preserved when the freezing property is lifted. Hence, it is unclear how to use the algorithms for the freezing cases into the non-freezing cases.

Majority automata networks. Another approach consists in generalizing the majority cellular automata from grids into an arbitrary graph. In that context, two perspectives have been taken in order to show the P-Completeness. In Eric GOLES et Pedro MONTEALEGRE 2015 it is shown that the prediction problem for the majority rule is P-Complete, even when the topology is restricted to planar graphs where every node has an odd number of neighbors. The result is based in a crossover gadget that uses a sort of *traffic lights*, that restricts the flow of information depending on the parity of the time-step. Then in Eric GOLES et Pedro MONTEALEGRE 2014 it is shown that the prediction problem for the majority rule is P-Complete when the topology is restricted to regular graphs of degree 3 (i.e. each node has exactly three neighbors). Both results are valid for the stable and biased majorities, as these rules are equal when the nodes have and odd number of neighbors.

Signed majority. In E. GOLES, P. MONTEALEGRE, K. PERROT et G. THEYSSIER 2017 the authors study the majority rule in two dimensional grids where the relations between neighboring cells have a *sign*. The *signed majority* consists in a modification of the majority rule, where the most represented state in a neighborhood is computed multiplying the state of each neighbor by the corresponding sign in the relation. The authors show that when the configuration of signs is the same on every site (i.e. we have an homogeneous cellular automata) then the dynamics and complexity of the signed majority is equivalent to the standard majority. Interestingly, when the configuration of signs may differ from site to site, the prediction problem is P-complete.

Asynchronous prediction. Another variant considers the prediction problem under a sequential updating scheme. More precisely, the *asynchronous prediction* problem asks for the existence of a permutation of the cells that produces a change in the state of a given cell, in a given time-step. In fact, in Cristopher MOORE 1997 Moore suggested in this case it holds a similar dichotomy than in the synchronous case : namely, the complexity in the two-dimensional case is lower than in three or more dimensions. This conjecture was proven in Eric GOLES, Diego MALDONADO et al. 2020 where it was shown that the asyncrhonous prediction in two dimensions is in NC, while it is NP-complete in three or more dimensions.

2.2 The Sandpile Variants

Sandpiles on lattices. The study of the computational complexity of sandpile CA started with the work of Moore and Nilssen in 1999 C. MOORE et al. 1999, where it is proven that the prediction problem with von Neumann neighborhood is in NC in

dimension one (the bound has been improved to AC^1 in MILTERSEN 2007), and is P-complete in dimension three and above. The two dimensional case is left open in this preliminary work, and has been the subject of subsequent works in many directions. A survey FORMENTI et K. PERROT 2019 proves than all prediction problems on lattices ($\mathbb{Z}^d$) are solvable in polynomial time, generalizing a result of Tardos TARDOS 1988 on finite undirected graphs. The survey FORMENTI et K. PERROT 2019 also generalizes the dimension sensitivity, by proving that the prediction problem is in NC for all one dimensional sandpile CA, and is P-complete for all three or more dimensional sandpile CA.

Crossover impossibilities. Towards the understanding of the computational complexity of the two dimensional sandpile CA with von Neumann neighborhood, Goles and Gajardo GAJARDO et al. 2006 proved that it is impossible to contruct a crossover gate (with elementary signals following the approach initiated by Banks on cellular automata BANKS 1971). In the present work, we develop on this standard approach and on obstacles to the simulation of monotone circuit value problem within sandpiles (crossover impossibility). Out of this context, the design of unconventional signals and unconventional circuit simulation is largely open DELORME et al. 2002. The authors of GAJARDO et al. 2006 also introduced the firing graph, which is a central concept of Section 4.2 (Definition 4.3). Despite the fact that crossover gates are known to be impossible in the two dimensional sandpile model with von Neumann neighborhood of radius one $\mathcal{N}_{\mathtt{vn}}$, no NC algorithm is known. The situation is identical for the Moore neighborhood $\mathcal{N}_{\mathtt{m}}$ GAJARDO et al. 2006; NGUYEN et al. 2018.

Neighborhood shapes. More two dimensional sandpile CA have been proven to have P-complete prediction problems, by reduction from **MCVP**. It turns out that slight modifications of von Neumann neighborhood, such as that of radius two or more, are P-complete GAJARDO et al. 2006. It is also the case for Kadanoff sandpile CA of radius two or more FORMENTI, E. GOLES et al. 2012. More generally, the shape of the neighborhood is not an obstacle to the existence of a crossover gate. Indeed, it is proven in NGUYEN et al. 2018 that increasing the radius of any shape leads to a neighborhood that is eventually able to perform a crossover. This is in particular the case of Moore neighborhood of big enough radius, and even of a discrete circular neighborhood of big enough radius.

Fungal sanpiles. Fungal sandpiles received a recent interest, it consists in splitting the rule application into an horizontal and a vertical phase. Crossover gates have been proven to exist even on very restricted version of the model, which has required subtle considerations on delays in the composition of semi-crossover gates E. GOLES, TSOMPANAS et al. 2020; MODANESE et al. 2022.

Reusable wires. A general framework for the design of P-completeness reductions in discrete dynamical systems is presented in E. GOLES, P. MONTEALEGRE, K. PERROT et G. THEYSSIER 2017, in terms of cellular automata and threshold networks. A focus is put on intrinsic universality, and the power of reusable wires (which is presumably

impossible to implement in sandpiles, because of the abelian property that the order of topplings commute) is emphazised through the discovery of a planar crossover gate in this context.

On overall, many neighborhoods in dimension two are known to have a P-complete sandpile prediction problem, and are therefore intrisically sequential. On the other hand, few others are known to have a sandpile prediction problem in NC, and are therefore efficiently parallelizable. More precisely, it is known that there are infinitely many neighborhoods $\mathcal{N} \subset_{\texttt{fin}} \mathbb{Z}^2$ having a P-complete prediction problem. To the best of our knowledge, it is currently not known whether there is an infinity of neighborhoods $\mathcal{N} \subset_{\texttt{fin}} \mathbb{Z}^2$ having a prediction problem in NC. Also, only finitely many neighborhoods are known to have a crossover impossibility.

Freezing sandpiles. The freezing world also received a dedicated attention. In the context of sandpiles, the complexity of the prediction problem is still open for the von Neumann neighborhood of radius one, although important restrictions are proven to be either equivalent, or in NC E. GOLES, P. MONTEALEGRE et K. PERROT 2021. For example, the prediction problem for 2-dimensional freezing sanpiles with the von Neumann neighborhood is still as computationally hard for initial configurations restricted to cells with 0, 2 and 4 grains only.

3 Majority Cellular Automata

This Chapter is structured into three sections. In Section 3.1, our focus is on the study of the Heterogeneous Majority Cellular Automata. We establish that the prediction problem for the 1-dimensional HMCA belongs to NL, while it turns out to be a P-complete problem for greater dimensions. Moving on to Section 3.2, our attention shifts to the study of the Freezing L-shaped MCA. We prove that, when considering the smallest L-shaped neighborhood (comprising the top and right sites), the problem can be solved efficiently with a parallel algorithm (NC). In contrast, it is P-complete for any larger L-shaped neighborhood. Finally, in Section 3.3 we provide the chapter's conclusions and outline potential directions for future research.

3.1 Heterogeneous Majority Cellular Automata

In this section, we study the prediction problem in heterogeneous majority automata setup, *i.e.*, mixing both the stable and biased majority rules. Note that the fact that the prediction problem for the canonical majority cellular automata has been proven P-complete for dimensions ≥ 3, implies the same complexity for the heterogeneous majority for dimensions ≥ 3 as one can choose the same rule everywhere. On the other hand, it is known that for 1-dimensional majority cellular automata, the prediction problem is in NC. Differently as before, this result does not direclty apply to the 1-dimensional heterogeneous majority CA. The results presented in this work show that the prediction problem for the 2-dimensional heterogeneous majority cellular automata is a P-complete problem (recall that the 2-dimensional case is an open problem for the canonical MCA). On the other hand, we prove that the prediction problem for the 1-dimensional heterogeneous majority is in NL.

3.1.1 1-dimensional Case

In this section, we show that restricted to one dimension, the heterogeneous majority cellular automata can be efficiently predicted. Indeed, we show that restricted to $d = 1$, the prediction problem belongs to NL. In fact, we show a stronger result : we show that if we consider the dynamics given by two consecutive iterations of an arbitrary 1-dimensional HMCA, i.e., if we study the dynamics given by F^2 where F is the global rule of the HMCA then, we have that it defines a bounded-change cellular automata.

Proposition 3.1 (Guillaume THEYSSIER et al. 2022)**.** *Restricted to one-dimensional bounded-change cellular automata, the prediction problem is in* NL.

Obviously, an HMCA is not bounded-change. For instance, if we take $d = 1$, n even and we define the configuration c that spatially alternates 1 and -1, we have that it induces a limit cycle of period two, independently of the local rule (biased or stable) that we choose. More precisely, if $d = 1$ and n is even, we can define for $i \in \{1, \ldots, n\}$:

$$c_i = \begin{cases} -1 & \text{if } i \text{ is even} \\ 1 & \text{otherwise} \end{cases}$$

We have that $F^{t+2}(c) = F^t(c)$ for all $t \geq 0$. Nevertheless, the situation is different when we look at the evolution of the rule every two time-steps. More precisely, let F be the global function of a one-dimensional HMCA, and let us call $F^2 = F \circ F$. In the following, we show that F^2 is bounded-change.

Lemma 3.1. *Let F be a one-dimensional HMCA. Then F^2 is a two-change one-dimensional cellular automata.*

Proof. For simplicity, in the following we identify state -1 with 0. Our proof consists in showing that, if a cell has a state transition from 0 to 1 under F^2, then this cell will remain fixed in state 1 on every following iteration of F^2. This means that in total a cell can have at most two state transitions under F^2.

Let us denote by u an arbitrary cell such that there exists an even time-step t so that $c_u^t = 0$ and $c_u^{t+2} = 1$. We claim that $c_u^{t+2k} = 1$ for every $k > 1$. We separate the proof of our claim in five cases, depending on the local functions of the cells adjacent to u. Let us call ℓ and r, respectively the left and right neighbors of u.

In our figures, cell u is highlighted, and the local rules of the cells are specified by a "b" or an "s" meaning "biased" and "stable" majorities, respectively. For instance, the case bbs is the case where u and ℓ have the biased majority, and r has the is stable majority.

In our cases we repeatedly use the following observations. First, if in a given time-step two adjacent cells are in state 1, then they will remain on state 1 on every future time-step independently of the local functions. Second, if in a given time-step two adjacent cells are in state 0, and the local function of both cells have is stable majority, then both cells will remain in state 0 on every future time-step. Now we are ready to tackle the cases.

Case 1 : bbb, bbs and sbb. Let us consider the case where u and ℓ have the biased majority rule (case sbb is symmetric to bbs). Since u is in state 1 at time-step $(t+2)$, then at time-step $t+3$ cell ℓ will be in state 1 making u to stay in state 1 in time-step $t+4$. Inductively, we conclude that $c_u^{t+2k} = 1$ for every $k > 1$.

	ℓ	u	r
	b	b	b/s
t	$\cdot$	0	$\cdot$
$t+1$	$\cdot$	$\cdot$	$\cdot$
$t+2$	$\cdot$	1	$\cdot$
$t+3$	1	$\cdot$	$\cdot$
$t+4$	$\cdot$	1	$\cdot$

Case 2 : bsb. This case is similar to Case 1. If cell u reaches state 1 at time-step $(t+2)$, then both neighbors will be in state 1 on time-step $t+3$, which turns u into 1 on time-step $t+4$. Inductively, we conclude that $c_u^{t+2k} = 1$ for every $k > 1$.

	ℓ	u	r
	b	s	b
t	$\cdot$	0	$\cdot$
$t+1$	$\cdot$	$\cdot$	$\cdot$
$t+2$	$\cdot$	1	$\cdot$
$t+3$	1	$\cdot$	1
$t+4$	$\cdot$	1	$\cdot$

Case 3 : sbs. First notice that if r and ℓ are in state 0 on t, then the three cells will remain fixed in state 0 on every future time-step, which contradicts the choice of u and t. Then, at least one neighbor of u is in state 1 on t.

Without loss of generality, let us suppose that $c_\ell^t = 1$. This implies that $c_u^{t+1} = 1$, since u has the biased majority rule. Moreover, since $c_u^{t+2} = 1$, necessarily one neighbor of u is in state 1 at time-step $t+1$. This implies that in time-step $t+1$, cell u and one of its neighbors are simultaneously in state 1. This implies that $c_u^{t+k} = 1$ for every $k > 0$.

	ℓ	u	r
	s	b	s
t	1	0	$\cdot$
$t+1$	1	1	$\cdot$
$t+2$	1	1	$\cdot$

	ℓ	u	r
	s	b	s
t	1	0	$\cdot$
$t+1$	$\cdot$	1	1
$t+2$	$\cdot$	1	1

Case 4 : sss. If the three cells are ruled by the stable majority and at least one neighbor is in state 0 on time-step t, then u will get fixed in 0, contradicting the choice of u and t. Then we assume that $c_\ell^t = c_r^t = 1$. This implies that $c_u^{t+1} = 1$. Since $c_u^{t+2} = 1$, then at least one of the neighbors of u is in state 1 at time-step $(t+1)$ (in our figure below, we assume without loss of generality that $c_r^{t+1} = 1$). Then, we have two adjacent cells in state 1, making $c_u^{t+k} = 1$ for every $k > 0$.

	ℓ	u	r
	s	s	s
t	1	0	1
$t+1$	·	1	1
$t+2$	·	1	1

Case 5 : bss **and** ssb. Let us study the case bss (the case ssb is symmetric). Observe that $c_r^t = 1$, beacause otherwise we will have two adjacent stable cells in 0 on the same time-step, fixing them on every future time-step, contradicting the choice of t and u. Since $c_u^{t+2} = 1$, at least one of the neighbors of u has to be in 1 in time-step $t+1$.

	ℓ	u	r
	b	s	s
t	·	0	1
$t+1$	·	·	1
$t+2$	·	1	·

	ℓ	u	r
	b	s	s
t	·	0	1
$t+1$	1	·	·
$t+2$	·	1	·

If $c_u^{t+1} = 1$, then we have that u and one of its neighbors are in state 1 in $t+1$, implying that u and that neighbor get fixed in state 1 on every future time-step.

	ℓ	u	r
	b	s	s
t	1	0	1
$t+1$	·	1	1
$t+2$	1	1	1

	ℓ	u	r
	b	s	s
t	1	0	1
$t+1$	1	1	·
$t+2$	1	1	·

If $c_u^{t+1} = 0$, then necessarily ℓ and r are in state 1 at time-step $(t+1)$. Since r has a stable-majority local rule and $c_u^t = 0$, then necessarily the right neighbor of r, namely w, is in state 1 at time-step t. This means that $c_r^{t+k} = c_w^{t+k} = 1$ for every $k > 0$. This implies that u and r are in state 1 at time-step $t+2$, meaning that $c_u^{t+1+k} = 1$ for every $k > 0$.

	ℓ	u	r	w
	b	s	s	
t	0	0	1	1
$t+1$	1	0	1	1
$t+2$	·	1	1	1
$t+3$	1	1	1	1

On all cases we conclude that $c_u^{t+2k} = 1$ for every $k > 1$. Therefore, the cellular automata given by global function F^2 is 2-change. ■

Theorem 3.1. *The prediction problem is in* NL *for every one-dimensional HMCA.*

Proof. Let (F, c, t, i) be an input of the prediction problem, where F is defined by some assignation of local rules $\mathscr{F}$. Let us denote G the global function F^2. Observe that as a consequence of Proposition 3.1 there exists a nondeterministic logarithmic-space algorithm $\mathscr{A}$ solving the prediction problem. Suppose first that t is even. In this case we simply run $\mathscr{A}$ on input $(G, c, t/2, i)$. Observe that $G^{t/2}(c)_i = F^t(c)_i$. This implies that the output of $\mathscr{A}$ on input $(G, c, t/2, i)$ is the answer of the prediction problem on input (F, c, t, i).

Suppose now that t is odd. In this case we run $\mathscr{A}$ on inputs

$$(G, c, (t-1)/2, i-1), (G, c, (t-1)/2, i) \text{ and } (G, c, (t-1)/2, i+1).$$

From the outputs given by $\mathscr{A}$ we can deduce the values of $G^{(t-1)/2}(c)_{i-1}$, $G^{(t-1)/2}(c)_i$ and $G^{(t-1)/2}(c)_{i+1}$. This values correspond to the states of cells $i-1$, i and $i+1$ in $F^{t-1}(c)$. Finally, using the local function of i we can compute $F_i^t(c)$ and decide the output. We deduce that the prediction problem is in NL. ■

3.1.2 2-dimensional Case

In this section we give evidence that the prediction problem is harder to compute in the two-dimensional case than in the one-dimensional case. More precisely, we show that with the right combination of local functions and cell states, it is possible to simulate any monotone Boolean circuit in the two-dimensional grid. Hence, the prediction problem for the two-dimensional HMCA is P-complete.

Theorem 3.2. *Restricted to two-dimensional HMCA with the von Neumann neighborhood, the prediction problem is* P*-complete.*

Proof. To show that the prediction problem is P-complete we reduce the Monotone Circuit Value problem to it. To do so, we simulate the input and the different parts of a Boolean circuit using a certain combination of biased and stable rules, on a certain initial configuration.

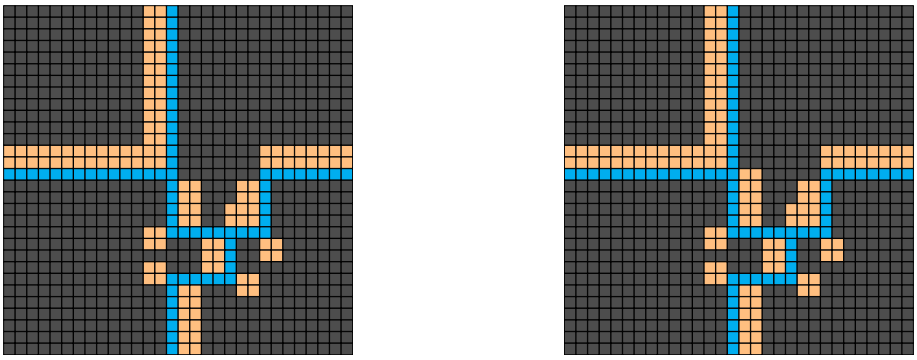
In fact, according to E. GOLES, P. MONTEALEGRE, K. PERROT et G. THEYSSIER 2017, it is enough to construct a series of *gadgets*, namely *wires*, conjunction and disjunction gates, together with a cross-over and signal multipliers. These *gadgets* have to be of constant size, and have to have two inputs and two outputs consistently fulfilling that the inputs are in the west and north side, and the outputs are in the east and south side of the gadget. Also, any rotation of these orientation constraints is valid. Additionally, inputs and outputs have to be placed in such a way that when two gadgets are put together, one of the outputs of one matches one of the inputs of the other. We follow this approach and build gadgets satisfying these conditions.

First, we explain how to simulate *wires*, i.e. gadgets that allow us to propagate signals through the grid. Wires are made by three rows (columns) of cells with the biased majority rule, two of them fully in state $+1$ while the other one remains in state -1. To have a TRUE signal, turn to $+1$ the first two cells of one side of the wire. An example of a wire is depicted in Figure 3.1.

FIGURE 3.1 : Examples of the functioning of wires. In left we have wire initialized as a FALSE signal. In the middle we have a wire initialized as a TRUE signal. In the right hand we have a wire with TRUE signal, after four time steps.

In all our figures, use the colors black (■) and white (□) to represent states -1 and $+1$ of the cells with the stable majority rule, while cyan (■) and orange (■) represents states -1 and $+1$ for the cells with the biased majority rule. Notice that for a wire to work, it needs a background of stable cells in state -1.

Next, we show how to simulate logic gates. Since we only simulate monotone circuits, we need to simulate conjunction and disjunciton gates with fan-in 2 and fan-out 2. Our gadgets, that we call *logic gadgets*, are depicted in Figure 3.2.



(a) Conjunction gadget (b) Disjunction gadget

FIGURE 3.2 : Logic gadgets that simulate conjunction gates (left) and disjunction gates (right).

Conjunction gates output TRUE only when both inputs are TRUE. In Figure 3.3 we show the evolution of a conjunction gadget in different combinations of input values. The disjunction gate construction is similar (see Figure 3.2b), but its outputs are TRUE with at least one TRUE input.

To simulate any circuit, we need to cross signals, but since the HMCA space is a planar graph, it is not evident we can do it. In order to do so, we created a cross-over gadget (Figure 3.4) that can distinguish one signal from the other by using the oscillating behavior of the HMCA as "traffic lights", letting the west signal go through its east output only, and the north signal through the south output only. For an example see Figure 3.5. In the case both signals are TRUE, the cross-over gadget works as a conjunction gate. No prior signal coordination is needed.

Finally, as the space is an undirected graph but circuits are defined over directed graph, we need diodes making the signals going only in directions west-east and north-south (Figure 3.6). For a better understanding of the diode behavior see Figures 3.7 and 3.8.

These diodes have to be appended to every gadgets outputs. In Figure 3.9 we depict the complete conjunction, disjunction and crossing gadgets, including the output diodes. Observe that, in at most 39 time-steps, the gadgets will produce the output values. In fact, the latter upper-bound is uniform for all the gadgets.

■

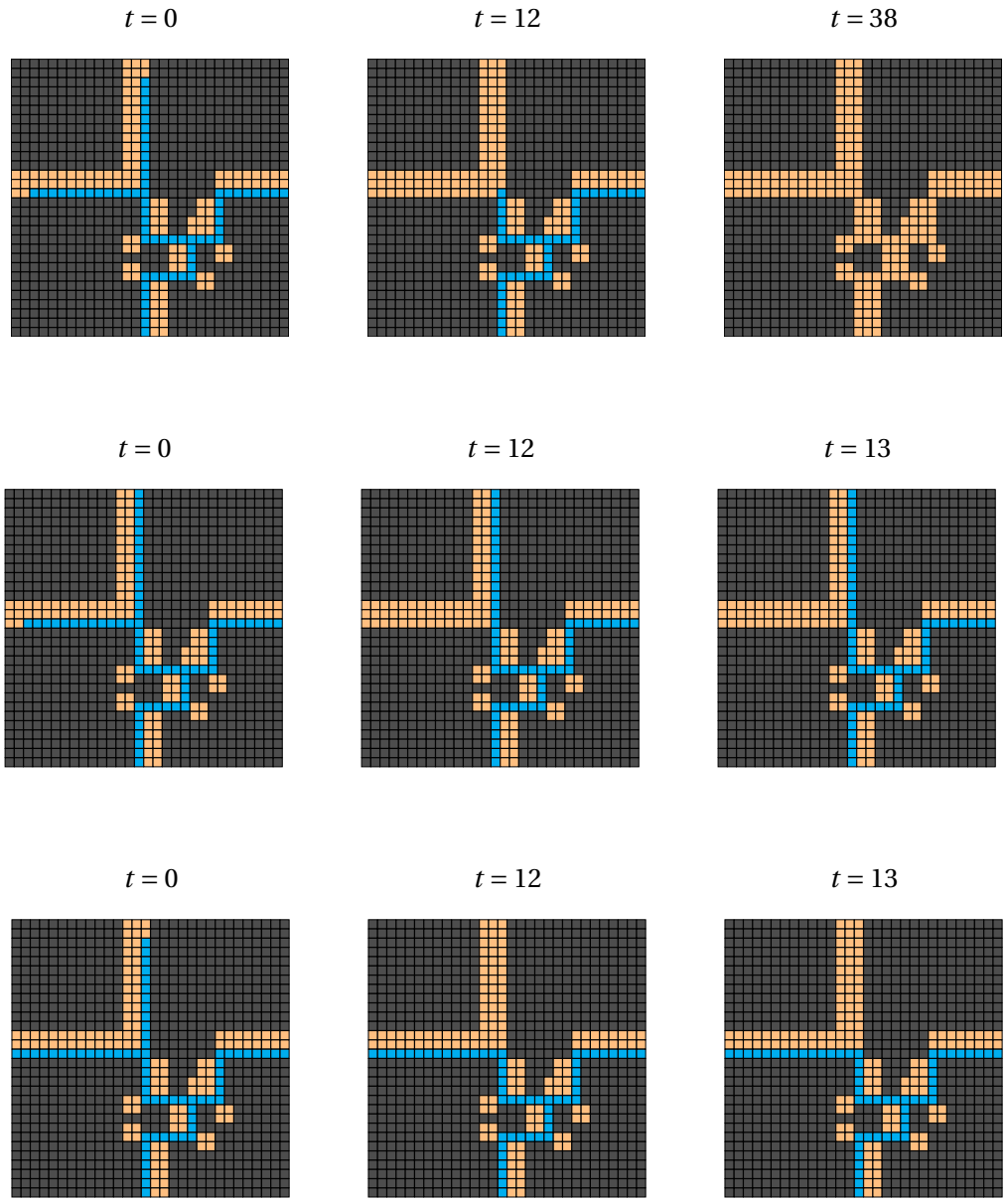


FIGURE 3.3 : Behavior of the conjunction gadget when two inputs are TRUE (top), when the left input is TRUE (middle) and when only the top input is TRUE (bottom).

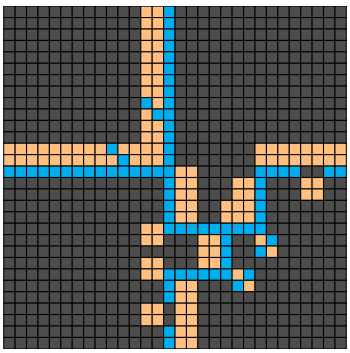


FIGURE 3.4 : Cross-over gadget.

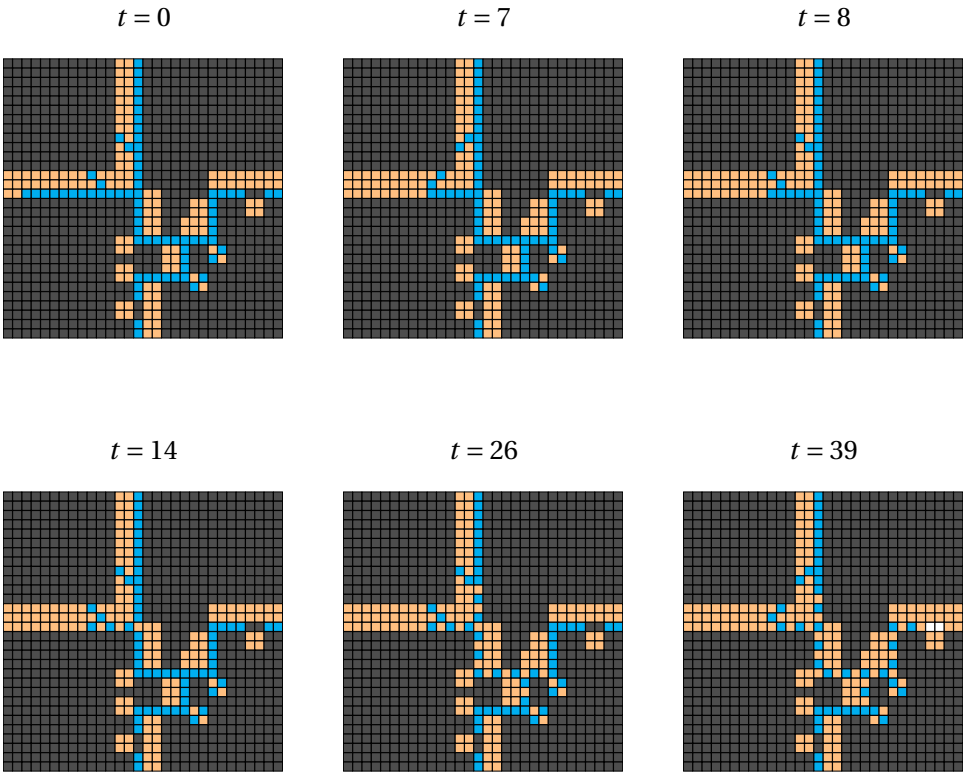


FIGURE 3.5 : Cross-over gadget behavior with one TRUE input.

(a) Vertical diode (b) Horizontal diode

FIGURE 3.6 : Diode gadgets. The vertical diode allows signal to pass only from top to bottom, while the horizontal diode allows the signal to go only from left to right.

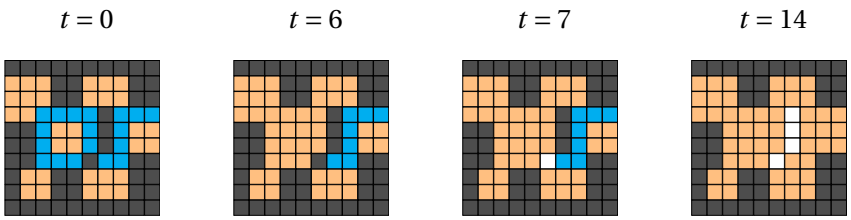


FIGURE 3.7 : Horizontal diode behavior when a signal comes from the west.

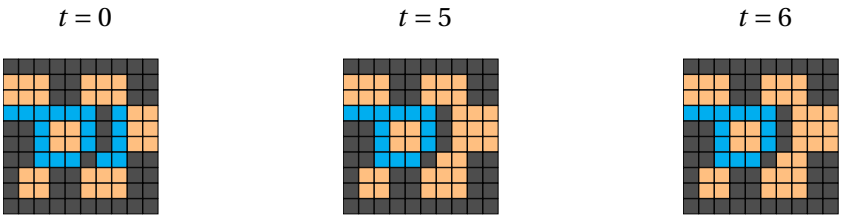


FIGURE 3.8 : Horizontal diode behavior when a signal comes from the east.

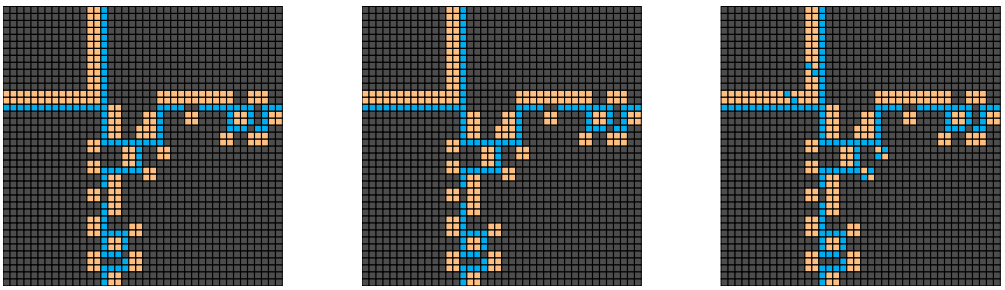


FIGURE 3.9 : Disjunction (left), Conjunction (middle) and crossing gadgets (right) including their output diodes.

3.2 Freezing L-shaped Majority Cellular Automata

In this section, we classify the prediction problem for several families of freezing MCA whose neighborhoods are L-shaped. More precisely, it is proved that for the classic L-neighborhood that consider the center site as well as the upper and the right sites, the decision problem is in NC. While for any other greater size L-neighborhood the prediction problem is P-complete, that is, there are no efficient parallel procedures to determine the state of one or more cells (unless P = NC). It should be noted that the classic L-neighborhood appeared in the literature associated with the problem of characterizing eraser cellular automata, that is, given an initial configuration with a finite number of 1s, characterizing the local rules that "erase" the islands of 1s in finite time, that is, every configuration with a finite number of 1s, converge to the consensus state 0^* TOOM 1980; Peter GÁCS et al. 1978; Péter GÁCS et al. 2022. In fact, also related with erasers, in Peter GÁCS et al. 1978 was conceived a non-connected majority CA whose neighborhood depends on each cell state. Specifically, for a cell i, if its state is 0 then the neighborhood corresponds to cells $(i-3)$, $(i-1)$ and i, while if the site i is in state 1 the majority is calculated over the cells i, $(i+1)$ and $(i+3)$. This rule solves the majority problem in most of the cases except on initial configurations close to equilibrium, meaning that both opinions are equally represented (also called null mangetization in the literature).

More precisely, we define an L-shaped neighborhood with two sets $S_N, S_E \subset_{\texttt{fin}} \mathbb{N} \setminus \{0\}$. In this fashion, we define the L-shaped Freezing Majority Cellular Automata (LFMCA) by the local function :

$$f(c)_{(i,j)} = \begin{cases} 1 & \text{if } c_{(i,j)} = 1, \\ 1 & \text{if } \sum_{k\in S_N} c_{(i,j+k)} + \sum_{k\in S_E} c_{(i+k,j)} > \frac{|S_N|}{2} + \frac{|S_E|}{2}, \\ -1 & \text{otherwise.} \end{cases}$$

This means that a cell in state 1 remains fixed in this state, while if it is in state -1, the majority is computed over the neighborhood defined by S_N and S_E. Here, S_N (resp. S_E) describes the cells to the north (resp. east) of the central cell.

3.2.1 The classic L-shaped neighborhood

In this part of our work, we study the prediction problem for the LFMCA considering the classic L-shaped neighborhood, also known as the Toom neighborhood. This neighborhood is defined by $S_N = S_E = \{1\}$.

In Lemma 3.2, we prove a geometric property of this LFMCA that we exploit in Theorem 3.3 in order to prove that there exist an efficient parallel algorithm (NC) for solving the prediction problem.

Lemma 3.2. *Let us take the LFMCA defined by $S_N = S_E = \{1\}$ over the grid $G = (V, E)$, and an initial configuration $c \in \{-1,+1\}^{n^2}$ and let $\overrightarrow{G} = (V_{-1}, \overrightarrow{E})$ be a graph such that $V_{-1} = \{v \in V : c_v = -1\}$ and $\overrightarrow{E} = \{(v_1, v_2) : v_1 = (i,j) \wedge (v_2 = (i+1, j) \vee v_2 = (i, j+1))\}$,*

then a cell v will remain fixed in state -1 if and only if it is in a cycle or in a path to a cycle of $\overrightarrow{G}$.

Proof. First of all, note that graph $\overrightarrow{G}$ contains all the cells that are initially in state -1.

Let us begin by proving that if v is in a cycle of $\overrightarrow{G}$, then it remains fixed in state -1. First, if v is in a cycle, it means it has at least one neighbor in state -1 at $t = 0$. We can assume that the other neighbor is in state $+1$, since this implies that cell v will change its state if and only if the neighbor in state -1 also changes. We can also extend this assumption for every node in the cycle. Since every node in the cycle is in state -1, and that they depend of each other to change their states, none of them will change. Something similar occurs when v is in a path to a cycle : it depends on the path nodes which in turn depend of the cycle, then it remain fixed in state -1.

In the other hand, let us prove that if there does not exist a time-step t such that $F^t(c)_v = +1$, it implies that v is in a cycle or in a path to a cycle of $\overrightarrow{G}$. If v is always in state -1, it means that at least one if its neighbors is also always in state -1, which in turn always has a neighbor in state -1, and so on. If we continue with this reasoning, at some point we will reach the "end" of the grid (remember we are considering periodical border conditions), and moreover, we will reach an already visited node. This node can be v, which means it is in a cycle. If it is not v, it means v is in a path to the cycle. ■

Theorem 3.3. *The prediction problem is in* NC *for the LFMCA with $S_N = S_E = \{1\}$.*

Proof. To prove it, we need to compute the graph $\overrightarrow{G}$ from Lemma 3.2 and then check whether the given node v is in a cycle, in a path to a cycle or neither. Keep in mind that everything have to be done in poly-logarithmic time with a polynomial number of processors.

For calculating $\overrightarrow{G}$ (Algorithm 1), we assign a processor to every node of the graph G. If the node is in state $+1$, the processor do nothing, otherwise, it will store in memory a pointer to every neighbor in state -1. This is done in constant time using n^2 processors.

For checking whether node v is in a cycle or in a path to a cycle, we first compute the powers of matrix A, i.e., $A, A^2, \ldots, A^n$. It can be done using the prefix sum and matrix multiplication techniques. We consider this algorithm as a black box (Algorithm 2).

To check if v is in a cycle, it is enough to read the entry $(i, j), (i, j)$ of every power of A. This can be done in constant time (Algorithm 3).

Finally, to decide whether v is in a path to a cycle, we use the following technique. Let us take the adjacency matrix A of graph $\overrightarrow{G}$, and add a new node named u, such that every node in a cycle points to it. Let us call this new matrix B. We can compute B by parallel executing the Algorithm 3 on every node. Next, we calculate the first $n+1$ powers of B. Finally, it is enough to check if there exists a $k \leq n+1$ such that $B^k[u, v] = 1$. The latter can be done similarly to Algorithm 3. ■

Algorithm 1: Computing $\overrightarrow{G}$.

Input: An initial configuration c (size $n \times n$).
Output: Adjacency matrix A of graph $\overrightarrow{G}$ (size $n^2 \times n^2$).
for $1 \leq i, j \leq n$ **pardo**
  if $c_{i,j} = -1$ **then**
    if $c_{i+1,j} = -1$ **then**
      $A[(i,j),(i+1,j)] = 1$
    end
    if $c_{i,j+1} = -1$ **then**
      $A[(i,j),(i,j+1)] = 1$
    end
  end
end

Algorithm 2: Computing $A, A^2, \ldots, A^n$.

Input: Adjacency matrix A (size $n^2 \times n^2$)
Output: Powers of A $P = A, A^2, \ldots, A^n$

Algorithm 3: Checking if $v = (i,j)$ is in a cycle of $\overrightarrow{G}$.

Input: Powers of $A : P = A, A^2, \ldots, A^n$
Output: answer = 1 if v is in a cycle of $\overrightarrow{G}$, answer = 0 otherwise
answer = 0
for $1 \leq k \leq n$ **pardo**
  if $P[k][(i,j),(i,j)] = 1$ **then**
    answer = 1
  end
end

3.2.2 Greater size L-shaped neighborhoods

As we mentioned before, we also tackle the prediction problem for greater size L-shaped neighborhoods. In particular, we consider the "connected" L-shaped neighborhood, defined by the sets $S_E = \{1,2,\dots,k_E\}$ and $S_N = \{1,2,\dots,k_N\}$ with $k_E, k_N \in \mathbb{N}$. The LMFCA with a connected L-shaped neighborhood have a P-complete prediction problem (Theorem 3.4). We also consider two types of "non-connected" L-shaped neighborhoods. First, we tackle the non-connected L-shaped neighborhood where both S_E and S_N are of size 2 (Theorem 3.5). Secondly, in Theorem 3.6 we consider the L-shaped neighborhood where all the elements of S_E (resp. S_N) are equally separated by a constant p (resp. p'). These two types of non-connected LMFCA turn out to have P-complete prediction problems.

Theorem 3.4. *When restricted to the LMFCA with sets $S_E = \{1,2,\dots,k_E\}$ and $S_N = \{1,2,\dots,k_N\}$ such that $k_E, k_N > 3$, the prediction problem is* P*-complete.*

Proof. In order to prove that the prediction problem is P-complete, we reduce the monotone circuit value problem to it. To do so, we simulate the input and the different parts of a Boolean circuit in a certain initial configuration that depends on the sets S_E and S_N. Without loss of generality, we assume that $k_E \geq k_N$.

In fact, according to E. GOLES, P. MONTEALEGRE, K. PERROT et G. THEYSSIER 2017, it is enough to construct a series of *gadget*, namely *wires*, conjunction and disjunction gates, together with a cross-over and signal multipliers. These gadgets have to be of constant size and have to have two inputs and two outputs consistently fulfilling that the inputs are in the east and north side and the outputs are in the west and south side of the gadget. Additionally, inputs and outputs have to be placed in such a way that when two gadgets are put together, one of the outputs of one matches one of the inputs of the other. We follow this approach and build gadgets satisfying these conditions.

First, we explain how to simulate *wires*, i.e., gadgets that allow us to propagate signals through the grid. Since k_E and k_N are not necessarily equal, the vertical and horizontal wires are not necessarily the same shape. Hence, we will see both cases separately. Vertical wires are made by $a = \left\lfloor \frac{k_E+k_N}{2} \right\rfloor$ columns of cells in state +1, as shown in Figure 3.10, and the signal have to pass $b = k_E - a$ columns to the left. For creating a TRUE signal, the first cell in the signal column has to be in state +1.

In the other hand, horizontal wires (see Figure 3.11) are made by k_N rows in state +1, and the signal is supposed to be just in the row below. In this case, for initializing a TRUE signal, it is necessary to have the first c cells in state +1, where

$$c = \begin{cases} b+1 & \text{if } k_E + k_N \text{ is even,} \\ b & \text{otherwise.} \end{cases}$$

Note that the horizontal wires cannot use more than a columns, or they would spread horizontally over all the rows they use. However, note that they can be separated by $k_E + k_N - a - 2$ columns and still work as wires without getting row-wise spread.

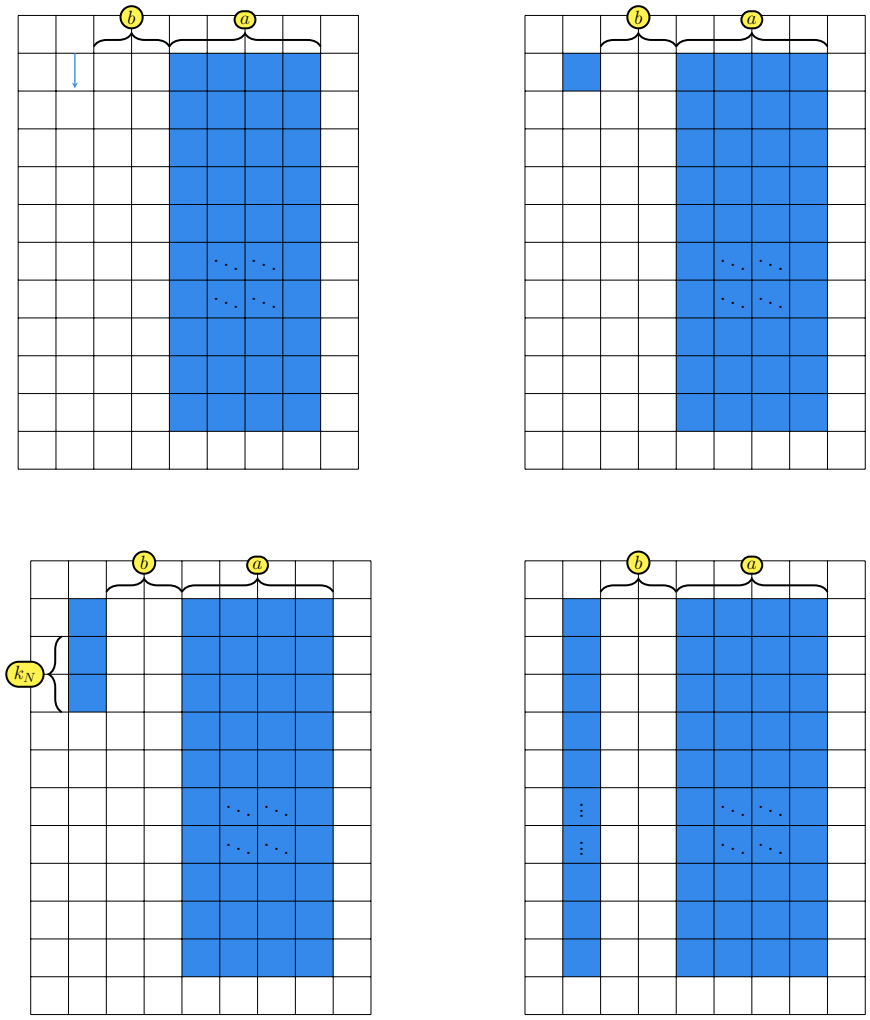


FIGURE 3.10 : Vertical wires. **Top left :** Wire initialized as a FALSE signal, the arrow represents the cell that has to be in state +1 in order to *turn on* the signal. **Top right :** Wire initialized as a TRUE signal. **Bottom left :** Wire initialized as a TRUE signal after 1 step. **Bottom right :** Wire with a TRUE signal after some steps.

Next, we show how to simulate logic gates. Since we only simulate monotone circuits, we need to simulate conjunction and disjunction gates with fan-in 2 and fan-out 2. The conjunction and disjunction gadgets are depicted in Figures 3.12 and 3.13, respectively. Conjunction gates output TRUE only when both inputs are TRUE. The disjunction gate construction is similar, but its outputs are TRUE with at least one TRUE input.

Finally, to simulate any circuit, we need to cross signals. In order to do so, we need a *cross-over* gadget (Figure 3.14) that let the east signal go through its west output only and the north signal through the south output only. In the case both signals are TRUE, the cross-over gadget works as a conjunction gate. No prior signal coordination is needed.

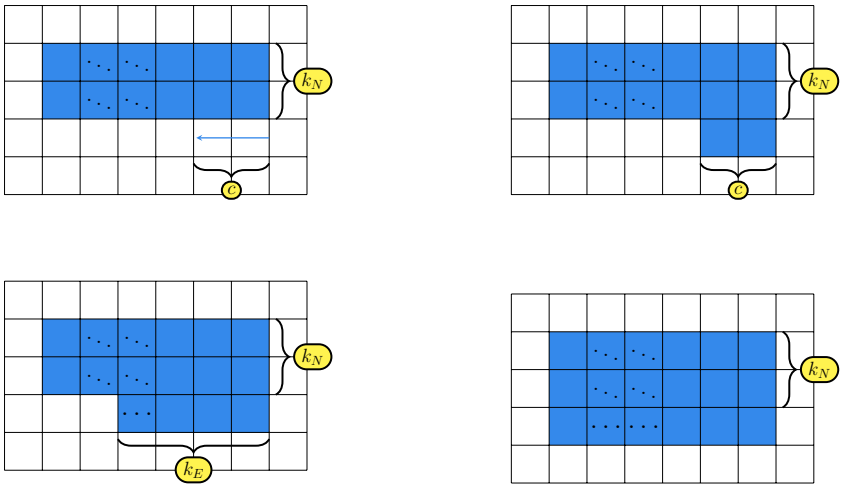


FIGURE 3.11 : Horizontal wires. **Top left :** Wire initialized as a FALSE signal, the arrow represents the cells that have to be in state +1 in order to *turn on* the signal. **Top right :** Wire initialized as a TRUE signal. **Bottom left :** Wire initialized as a TRUE signal after 1 step. **Bottom right :** Wire with a TRUE signal after some steps.

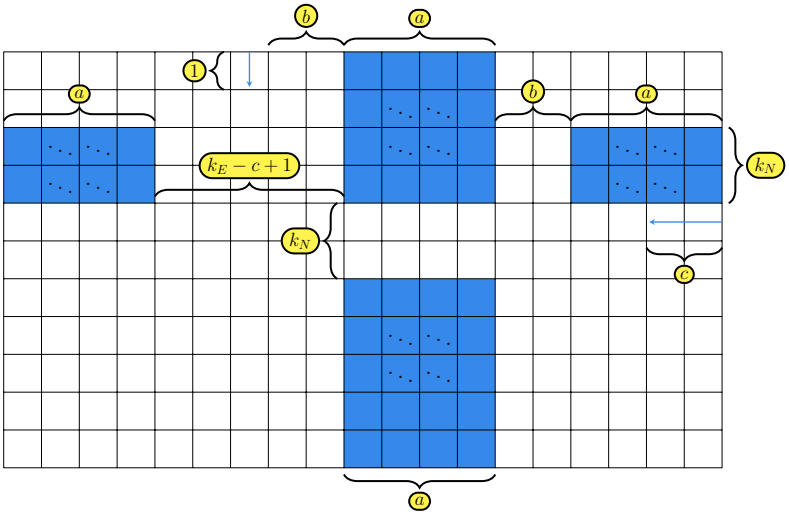


FIGURE 3.12 : Connected neighborhood AND gadget

■

Theorem 3.5. *When restricted to the LMFCA defined with sets* $S_E = \{i_E, j_E\}$ *and* $S_N = \{i_N, j_N\}$ *for some* i_E, j_E, i_N, j_N *such that :*

- i_N *is not a multiple of* i_E,
- i_E *is not a multiple of* i_N,
- j_N *is not a multiple of* i_E,

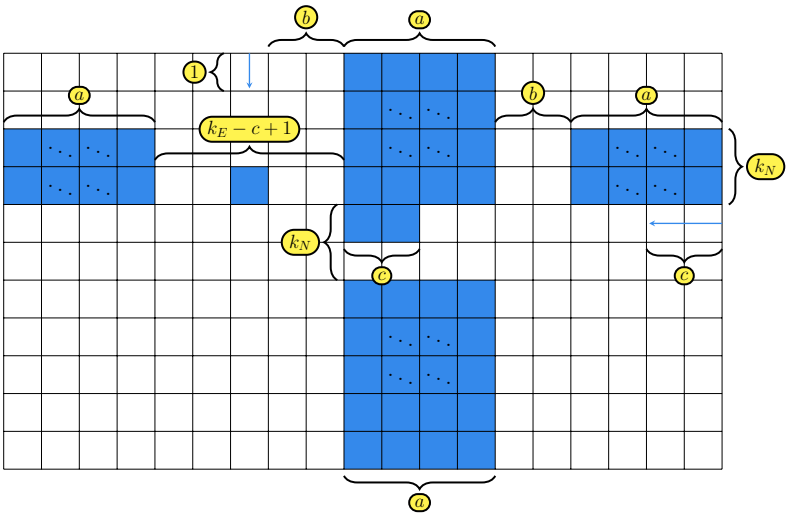


FIGURE 3.13 : Connected neighborhood OR gadget

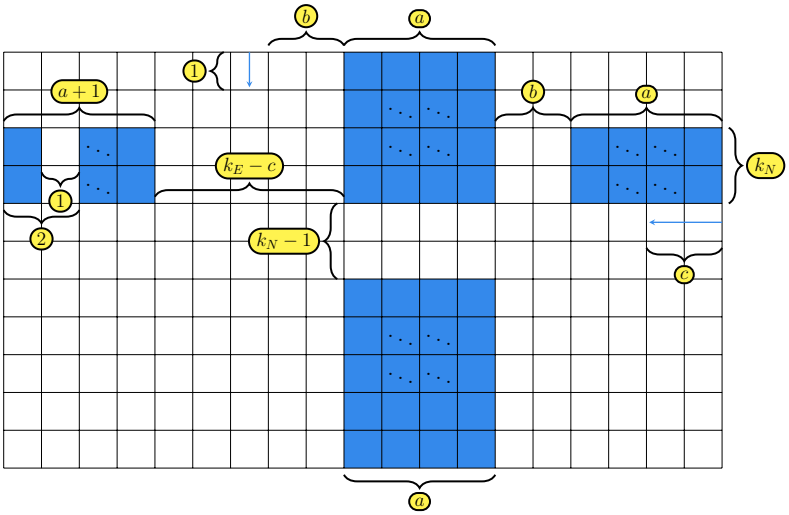


FIGURE 3.14 : Connected neighborhood Crossover gadget

- j_E *is not a multiple of* i_N,
- $0 < i_E < j_E - 1$ *and* $0 < i_N < j_N - 1$,

the prediction problem is P*-complete.*

Proof. As well as Theorem 3.4, we reduce the monotone circuit value problem to the prediction problem. To do so, we construct the conjunction, disjunction and crossover gadget. These gadgets are depicted in Figures 3.15, 3.16 and, 3.17 respectively. Note that, for a signal to be initialized as TRUE, it only needs one cell in state +1. It is of utmost importance that such cell is the cell immediately at the north(east) for the signal column(row), otherwise, it will lead to malfunction of the gadget. The latter fact allows to put the gadgets just one beside/above/below the other. Finally, note that

the gadgets size depends on a fixed value $k > 0$, which has to be the same for every gadget. ■

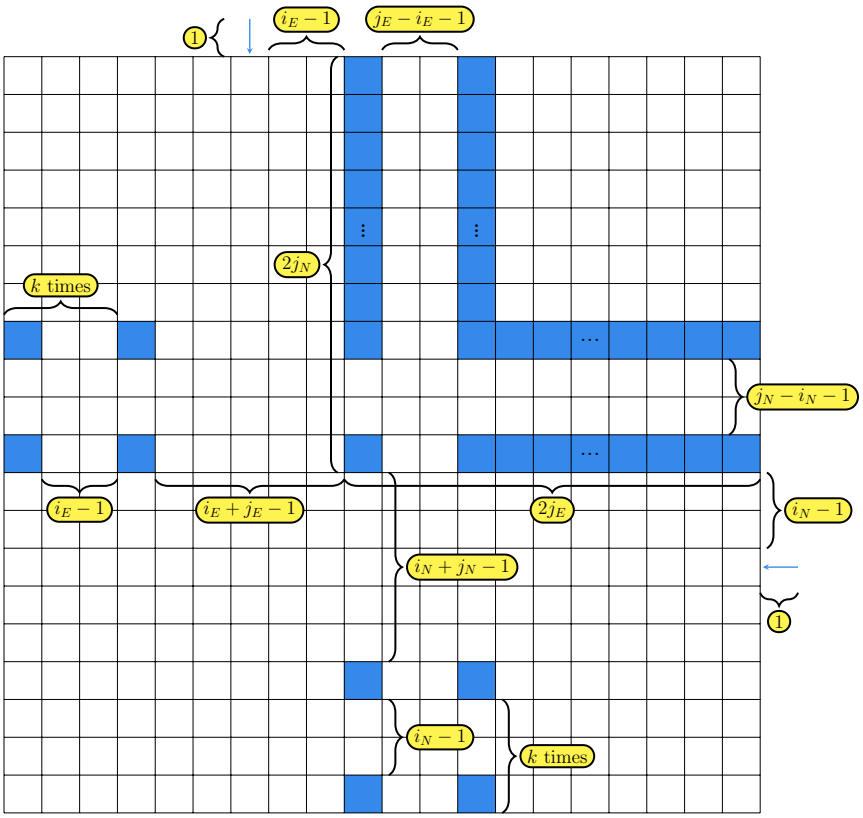


FIGURE 3.15 : Non-connected neighborhood AND gadget

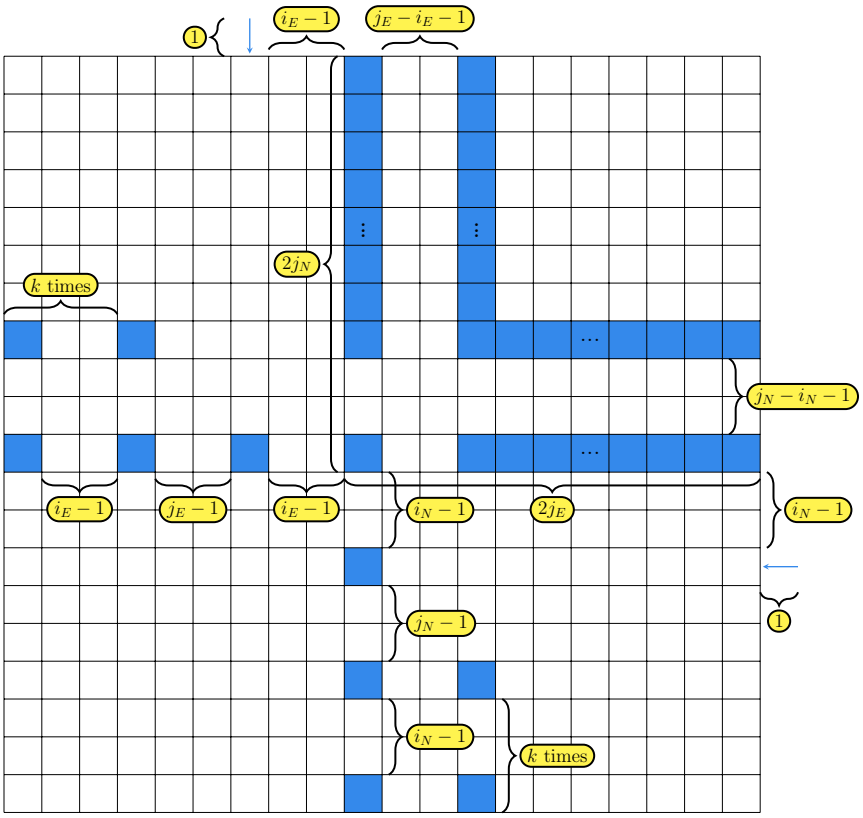


FIGURE 3.16 : Non-connected neighborhood OR gadget

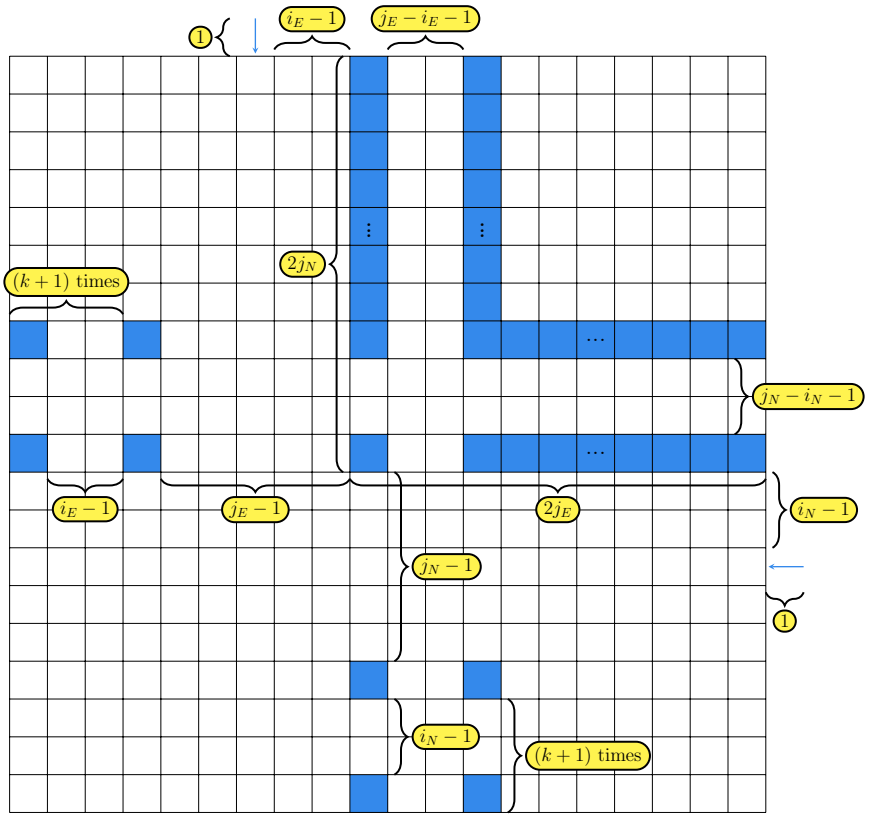


FIGURE 3.17 : Non-connected neighborhood Crossover gadget

Theorem 3.6. *When restricted to the LFMCA defined with sets S_E and S_N satisfying all the following conditions :*

- $|S_E| \geq 2$,
- $|S_N| \geq 2$,
- $(\exists p \in \mathbb{N})(\forall a \in S_E) : a = 0 \mod p$, *and*
- $(\exists p' \in \mathbb{N})(\forall b \in S_N) : b = 0 \mod p'$,

the prediction problem is P*-complete.*

Proof. Since every element of S_E is a multiple of p, it will create partitions over the grid. The same happens with S_N. These partitions do not interact with each other, i.e., cells from one partition do not have neighbors from any other partition. This independence let us see each partition as an independent grid, then one can directly use the gadgets from Theorem 3.4. The latter mean that one can reduce the monotone circuit value problem to this instance of the prediction problem, then it is P-complete. ■

3.3 Chapter Conclusions and Future Work

In Section 3.1, we approximated the prediction problem for the conventional 2-dimensional Majority Cellular Automaton (MCA) by combining both the canonical majority and the biased majority. Our findings indicated that this combination results in a P-complete problem. This raises additional questions beyond the prediction aspect of the canonical majority : What is the complexity of the prediction problem when considering a combination of biased-to-0 and biased-to-1 majority rules? Furthermore, do mixed rules lead to an increase/decrease in the complexity of the associated prediction problem? These questions remain open and present promising avenues for future investigations.

On the other hand, in Section 3.2, we prove the existence of a fast parallel solution for the prediction problem associated with the Freezing MCA when considering the Toom neighborhood. However, the complexity shifts to P-complete as the neighborhood size expands. We propose two possible directions for future work on this topic. First, considering neighborhoods defined over the first quadrant, in other words, for a given radius r, the neighborhood of cell $(0,0)$ could be any subset of the square delimited by coordinates $(0,0)$ and (r,r). This includes a broader category of L-shaped neighborhood, as the ones investigated in this study exhibit certain regular characteristics (connected, equally spaced neighbors, disconnected but with a size equal to 2). Secondly, another direction for future research involves studying the complexity of the prediction problem for the non-freezing MCA, taking into account the L-shaped neighborhoods defined as described above.

4 The Sandpile Cellular Automata

This Chapter is aimed to the study of both prediction and timed-prediction problems over 2-dimensional sandpiles. As we mentioned before, both of these problems are properly classified for one dimension sandpiles as NC problems, while for dimensions greater of equal to three the prediction problem is P-complete. The case of the 2-dimensional sandpile considering either the von Neumann or the Moore neighborhood is still an open problem. The underlying challenge in the context of 2-dimensional sandpile models is the absence of an evident construction of a crossover gate. Furthermore, the work of Goles and Gajardo GAJARDO et al. 2006 has sustained this difficulty, proving that a crossover gate cannot exist for the 2-dimensional sandpile, whether one considers the von Neumann neighborhood or the Moore neighborhood (the eight adjacent cells surrounding a cell). It is crucial to emphasize that the impossibility to construct a crossover gate solely signifies the limitation of the Banks approach, therefore the classification of the prediction problem remains unsolved, as we mentioned before.

This leads us to the focus of this Chapter, where we extend notions in relation to the complexity of 2-dimensional sandpiles by meticulously studying all the sub-neighborhoods within the Moore neighborhood, *i.e.*, all the 256 possible combinations of the eight cells included in the Moore neighborhood. Surprisingly, we found that there are 12 neighborhoods that admit a crossover gate. For them, we construct all the necessary circuitry for proving the P-completeness of their prediction problem. In contrast, for the remaining sub-neighborhoods, we extend the findings of Goles and Gajardo by establishing the impossibility of a crossover gate. While some of these proofs present fairly straightforward conclusions, others require deeper analysis and geometric arguments. In this Chapter, we also prove that the timed prediction problem is P-complete for 52 different neighborhoods : despite the fact that most of them do not permit a regular crossover gate, their inner dynamics is provably inherently sequential.

This Chapter is divided as follows. In Section 4.1 we restate the prediction problem and the timed prediction problem into more adequate although equivalent versions for the sandpile CA. Section 4.2 conducts an exhaustive analysis of the prediction problem associated to every possible sub-neighborhood of the Moore neighborhood. In Section 4.3 we spotlight neighborhoods with a P-complete timed prediction problem. Concluding this Chapter, Section 4.4 presents a comprehensive discussion concerning prediction problems within the context of sandpiles. It considers the analytical tools employed in their study and explores potential model variations, providing perspectives on the subject.

4.1 Sandpiles Prediction Problems and Crossovers

For the sake of simplicity, in this section we define an equivalent version of both, the prediction and timed-prediction problem. Also, since for the odel the rule is always the same and it is the neighborhood that changes, we define the prediction problem for a specyfic neighobrhood $\mathcal{N}$ as $\mathcal{N}$**-PRED**. For any $p \in \mathbb{Z}^2$, let $1p$ be the configuration containing a unique grain at position p, and let + denote the cell-wise addition of two configurations ($\forall p \in \mathbb{Z}^2 : (c+c')(p) = c(p) + c'(p)$). $\mathcal{N}$**-PRED** asks whether cell $q \in \mathbb{Z}^2$ topples during the evolution from $c + 1p$ to $(c+1p)^\circ$.

$\mathcal{N}$-sandpile prediction problem ($\mathcal{N}$**-PRED**)
Input:

- a finite stable configuration $c \in \mathbb{N}^{\mathbb{Z}^2}$
- and two positions $p, q \in \mathbb{Z}^d$.

Question: does $\exists t \in \mathbb{N} : F^t(c+1p)(q) \geq \theta$?

In the case of the $\mathcal{N}$**-TIMED-PRED**, it asks whether cell q topples at time step $t \in \mathbb{N}$ during the evolution from $c + 1p$:

$\mathcal{N}$-sandpile timed prediction problem ($\mathcal{N}$**-TIMED-PRED**)
Input:

- a finite stable configuration $c \in \mathbb{N}^{\mathbb{Z}^2}$
- two positions $p, q \in \mathbb{Z}^d$
- and a time $t \in \mathbb{N}$.

Question: does $F^t(c+1p)(q) \geq \theta$?

Since some of our results prove the inexistance of a crossover gate for a particular neighborhood, we first need to precisely define what a crossover gate is. We denote $[n] = \{1, \dots, n\}$ and $[\![n]\!] = \{0, \dots, n-1\}$.

Definition 4.1 (Crossover gate)**.** A *crossover gate* is a square subset of the lattice on which a stable configuration is set, such that it transports signals from two pairs of opposite sides, independently from each other : two wires cross each other. Given a sandpile CA $\mathcal{N}$, a crossover gate is formally defined as a configuration $g : [\![m]\!]^2 \to \{0, \dots, \theta - 1\}$ together with two pairs $(n,s),(w,e) \in ([\![m]\!]^2 \setminus \{0, m-1\}^2)^2$ of cells not in the corners such that :

- (n,s) are on opposite sides of $[\![m]\!]^2$, *i.e.* $\exists! i_{n,s} \in [\![2]\!]$ such that $|n_{i_{n,s}} - s_{i_{n,s}}| = m-1$;
- (w,e) are on opposite sides of $[\![m]\!]^2$, *i.e.* $\exists! i_{w,e} \in [\![2]\!]$ such that $|w_{i_{w,e}} - e_{i_{w,e}}| = m-1$;

- (n,s) and (w,e) are on different sides of $[\![m]\!]^2$, *i.e.* $i_{n,s} \neq i_{e,w}$;
- adding a grain at n eventually topples s, without toppling any cell on the other sides, *i.e.* $\exists t \in \mathbb{N} : F^t(g+1_n)(s) \geq \theta$ but, denoting $O(i) = \{x \in [\![m]\!]^2 \mid \exists j \in [2] \setminus \{i\} : x_j \in \{0, m-1\}\}$ the sides in dimensions different than i, we have $\forall p \in O(i_{n,s}) : \forall t \in \mathbb{N} : F^t(g+1_n)(p) < \theta$.
- adding a grain at w eventually topples e, without toppling any cell on the other sides, *i.e.* $\exists t \in \mathbb{N} : F^t(g+1_e)(w) \geq \theta$ but, denoting $O(i) = \{x \in [\![m]\!]^2 \mid \exists j \in [2] \setminus \{i\} : x_j \in \{0, m-1\}\}$ the sides in dimensions different than i, we have $\forall p \in O(i_{n,s}) : \forall t \in \mathbb{N} : F^t(g+1_n)(p) < \theta$.

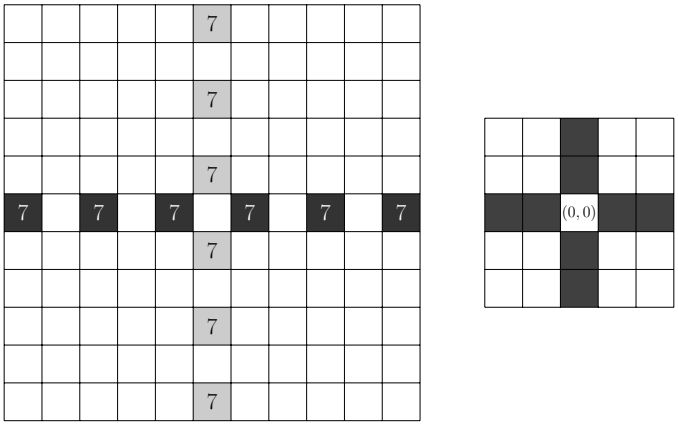


FIGURE 4.1 : Example (from GAJARDO et al. 2006) of crossover gate of size 11×11, for the von Neumann neighborhood of radius two whose eight cells are depicted on the right :

$$\mathcal{N}_{\text{vn2}} = \bigcup_{i \in \{-2,-1,1,2\}} \{(i,0),(0,i)\}).$$

Adding a grain at the grey cell on the north side (resp. black cell on the west side) triggers a chain reaction of topplings eventually toppling the grey cell on the south side (resp. black cell on the east side).

An example of crossover gate is depicted on Figure 4.1, with $n = (5,10), s = (5,0), w = (0,5), e = (10,5)$: adding a grain at n eventually topples s without toppling any cell from the east nor west sides, and adding a grain at w eventually topples e without toppling any cell from the north nor south sides. It is necessary to provide this formal definition (because we will state some impossibility results), although it is difficult to use in practice, and in the following no precise reference to its constituents will be made. The main tool we will employ is the *pair of firing graphs*, presented at the beginning of Section 4.2.

For the timed prediction problem, a *timed crossover gate* is required. It is the same as a crossover gate, except that all quantifications over time t are bounded to a given value $T \in \mathbb{N}^+$ called the *delay* of the gate (which is part of the definition of the gate). That is, when adding a grain to one side (at n or w), the cell on the opposite side (s or

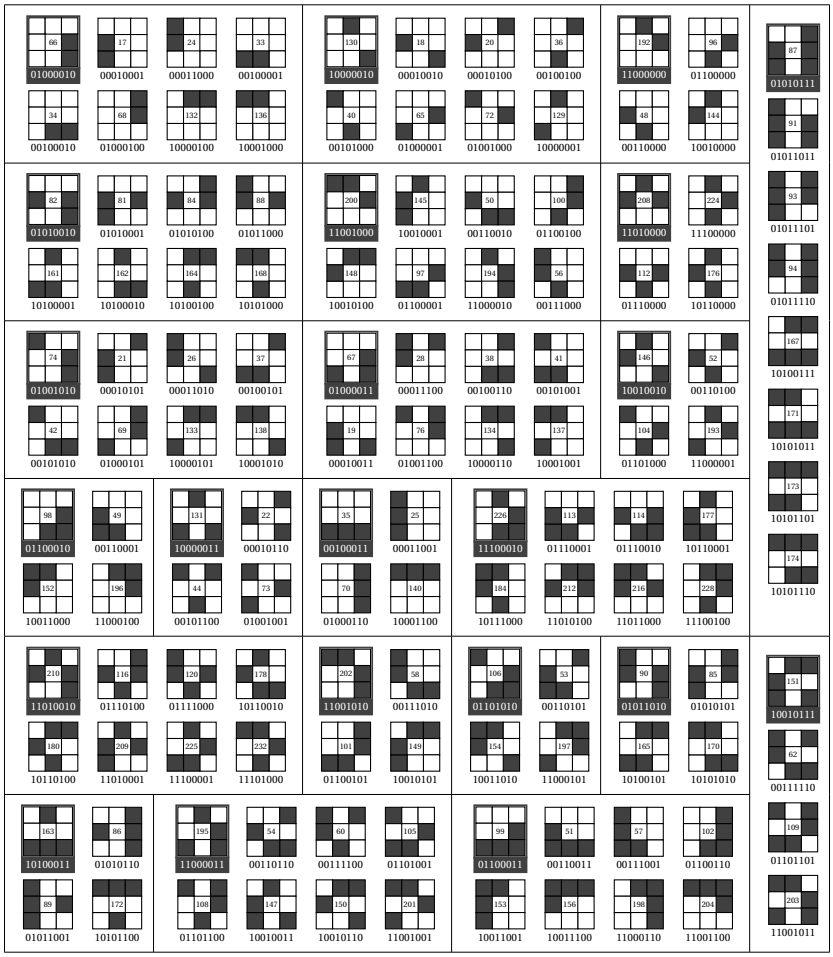


TABLE 4.1 : Equivalence classes of neighborhoods among subsets of $\mathcal{N}_{\mathtt{m}}$. The representative of each class is highlighted. (Part 1/2)

e) must topple at time step T, and no cell on the other sides must topple before time step $T+1$.

4.2 Crossover among subsets of Moore

In this section we present an exhaustive study of whether each subset of the Moore neighborhood $\mathcal{N}_{\mathtt{m}}$ admits a crossover gate or not, in the (uniform) sandpile CA. Observe that some of the 2^8 subsets of $\mathcal{N}_{\mathtt{m}}$ (which has 8 elements) do not span $\mathbb{Z}^2$, and the existence of a crossover gate is invariant by rotation and axial symmetries of the neighborhood. It reduces the number of neighborhoods to study to 43 equivalence classes, presented in Table 4.1. The numbering of neighborhoods is presented below.

Definition 4.2 (Subset of Moore neighborhood)**.** *Let* $\mathtt{n},\mathtt{ne},\mathtt{e},\mathtt{se},\mathtt{s},\mathtt{sw},\mathtt{w},\mathtt{nw}$ *denote the eight cardinal coordinates, starting with* north *and clowkwise. We will denote a subset of Moore neighborhood as an 8-bits string, where the* i*-th bit encodes whether the* i*-th*

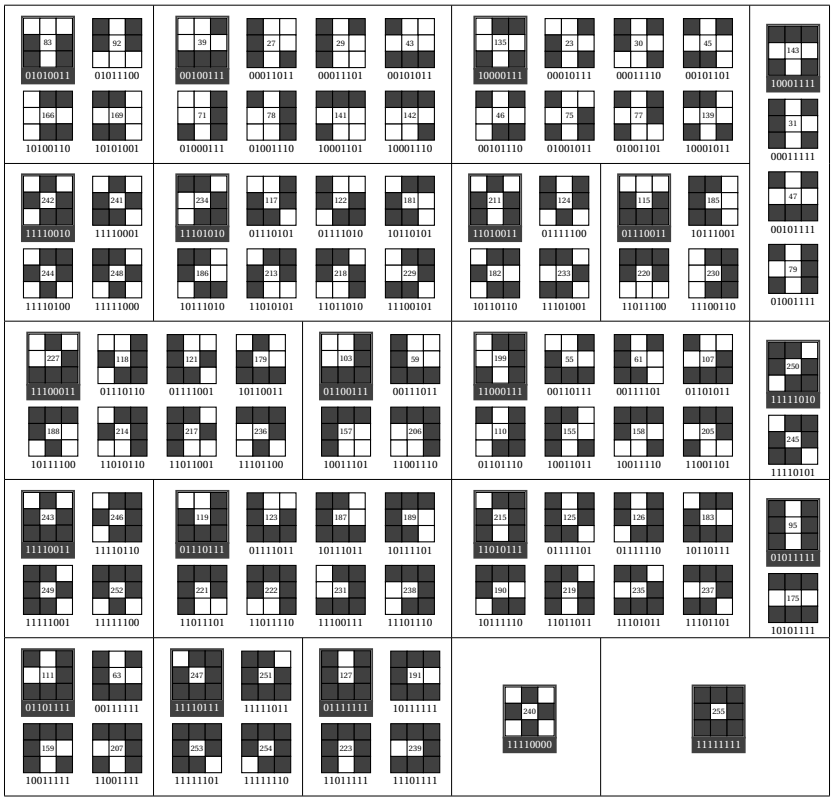


TABLE 4.1 : Equivalence classes of neighborhoods among subsets of $\mathcal{N}_{\mathtt{m}}$. The representative of each class is highlighted. (Part 2/2)

coordinate in the list n,e,s,w,nw,ne,se,sw*, is part of the neighborhood (bit 1), or not (bit 0). We may also interpret 8-bits strings as numbers (converted to decimal) with the least significant bit to the right, see Figure 4.2.*

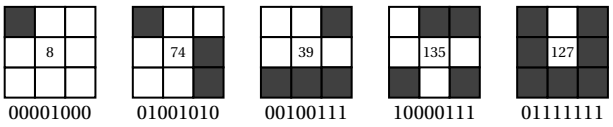


FIGURE 4.2 : From left to right, subsets of Moore neighborhoods corresponding to 00001000 = 8, 01001010 = 74, 00100111 = 39, 10000111 = 135, 01111111 = 127.

In this numbering scheme, the von Neumann neighborhood ($\mathcal{N}_{\mathtt{vn}}$) corresponds to 11110000 = 240, while the Moore neighborhood ($\mathcal{N}_{\mathtt{m}}$) corresponds to 11111111 = 255. It is worth noting that neighborhoods with numbers ≤ 15 consist exclusively of diagonal neighbors. Consequently, these neighborhoods do not span $\mathbb{Z}^2$.

We illustrate with a small figure next to each Theorem statement which neighborhoods we are considering. They are named in the same order they are shown.

Let us start with the early results that motivate the present study.

Theorem 4.1 (GAJARDO et al. 2006; NGUYEN et al. 2018)**.** *Neighborhoods* 240 *(von Neumann* $\mathcal{N}_{\mathtt{vn}}$*) and* 255 *(Moore* $\mathcal{N}_{\mathtt{m}}$*) do not admit a crossover gate.*


11110000 = 240


11111111 = 255

This reduces the equivalence classes from 43 to 41.

The causal structure of topplings is captured by the concept of firing graph, introduced in GAJARDO et al. 2006. It is at the basis of our reasonings in this work. Given a neighborhood $\mathcal{N}$, we say that cell v_1 is an *in-neighbor* of cell v_2 when $(v_2 - v_1) \in \mathcal{N}$. In the symmetric situation $(v_1 - v_2) \in \mathcal{N}$ it is called an *out-neighbor*. Observe that v_1 may be both an in-neighbor and an out-neighbor of v_2. In particular, this is always the case for symmetric neighborhoods, such as von Neumann $\mathcal{N}_{\mathtt{vn}}$ and Moore $\mathcal{N}_{\mathtt{m}}$.

Definition 4.3 (Firing graphs)**.** Given a crossover gate g on cells $n, s, w, e \in [\![m]\!]^2$, we define its two *firing graphs* $G^g_{ns} = (V_{ns}, A_{ns})$ and $G^g_{we} = (V_{we}, A_{we})$ as :

- V_{ns} (resp. V_{we}) is the set of fired cells after adding a grain at cell n (resp. w);
- there is an arc $(v_1, v_2) \in A_{ns}$ (resp. A_{we}) when $v_1, v_2 \in V_{ns}$ (resp. V_{we}), and both :
  - v_1 is an in-neighbor of v_2 for $\mathcal{N}$, and
  - v_1 is fired strictly before v_2 after adding a grain at cell n (resp. w).

The endpoints of a firing graph are called *starting cells* (n and w) and *ending cells* (s and e) respectively. For every arc (v_1, v_2) in a firing graph, we call v_1 a *predecessor* of v_2. Naturally, for every arc (v_1, v_2), we call v_2 a *successor* of v_1. The transitive and reflexive closure of the predecessor (resp. successor) relationship is the *ancestor*

(resp. *descendant*) relationship. By definition, a predecessor is necessarily an in-neighbor, and a successor is necessarily an out-neighbor. The firing graph captures the *effective* causal relationships among cells within a given crossover gate, whereas the in-neighbor and out-neighbor relationships can be seen as *potential*.

Section outline. We present our findings across distinct subsections to facilitate a structured exploration of our results. Initially, our intention was to group numerous neighborhoods into collective proofs. However, we encountered challenges in grouping them together because there is no evident factorization of the arguments, *i.e.*, almost all the proofs are unique for each neighborhood. As a result, we opted to categorize them into separate subsections based on their distinctive characteristics. In Subsection 4.2.1, we show a striking discovery, two sub-neighbors of $\mathcal{N}_{\mathtt{m}}$, identified by the numbers 135 and 143, unexpectedly exhibit the capability for a crossover gate. We prove that both problems 135-**PRED** and 145-**PRED** are P-complete by crafting all the necessary circuitry for reducing from **MCVP**. In Subsection 4.2.2, we introduce the framework used for proving the impossibility of a crossover gate for all the other neighborhoods. This framework hinges on a pivotal concept, the "crossing constraint", which serves as a primary component in the subsequent proofs presented in this section. In Subsection 4.2.3, we prove that all the neighborhoods generating planar graphs on the grid do not admit a crossover gate. Moving to Subsection 4.2.4, our attention turns to neighborhoods 247 and 127. These neighborhoods stand out for their size of 7, which means they are only missing one cell out of the eight found in the Moore neighborhood. Subsection 4.2.5 is dedicated to the study of neighborhoods featuring only two cells on the diagonals, while Subsection 4.2.6 focuses on neighborhoods with three and four diagonal cells. At this stage, we are left with just two remaining equivalence classes, represented by neighborhoods 95 and 39. We have assigned individual subsections to these classes due to their distinct proofs involving the new concept of "temporal minimality of escape cells". In Subsection 4.2.7, we exploit it to prove that a crossover is impossible for neighborhood 95. Finally, Subsection 4.2.8 is dedicated in its entirety to the comprehensive study of neighborhood 39, undoubtedly presenting the most intricate proof among all the Moore sub-neighborhoods. This subsection entails not only a synthesis of prior concepts but also the introduction of new ones, all within a proof structure that relies on nested cases and inductions under narrow hypotheses. The intricacy of the proof can be attributed to the intuitive notion that neighborhood 39 almost permits a crossover gate, which is discussed at the beginning of the Subsection. Table 4.2 summarizes the results of this section.

4.2.1 Crossover possibility

Neighborhoods studied in this subsection :

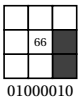
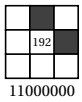
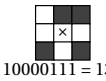
Section	4.2.1	4.2.3	4.2.4	4.2.5	4.2.6	4.2.7	4.2.8
Type	P-complete	Planar	Too many neighbors	2 diagonals	3 and 4 diagonals	Escape cell	Almost crossing
Neigborhoods	 10000111  10001111	 01000010  11000000  10000010  11010000  10010010  11001000  01010010  01100010  01001010  11100010  11010010  11001010  01101010  01011010  11110010  11101010  11111010	 11110111  01111111	 00100011  01000011  10000011  10100011  01100011  11000011  01010011  01110011  11100011  11010011  11110011	 01100111  10010111  01010111  11000111  11010111  01110111  01101111	 01011111	00100111
Have a crossover gate	✓	✗					

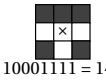
TABLE 4.2 : Summary of results exposed in Section 4.2.

10000111 10001111

Two non-equivalent subsets of Moore neighborhood admit a crossover gate, and

are consequently P-complete to predict, as formalized by $\mathcal{N}$**-PRED** problem. They are neighborhoods 135 and 143, for which we present gadgets for the purpose of implementing arbitrary Boolean circuits. Following one grain addition (at p), the dynamics of the sandpile then simulates the computation of the circuit, until the output bit causes (bit 1) or not (bit 0) the toppling of a given cell (at q).





Theorem 4.2. *$\mathcal{N}$-**PRED** is* P*-complete for neighborhoods* 135 *and* 143.

Proof. We prove P-completeness by reduction from monotone circuit value problem (**MCVP**) with gates of fan-in and fan-out two (problem **AM2CVP** in GREENLAW et al. 1995), We consider an embedding of the monotone circuit on the two dimensional grid (see Figure 4.3), and replace all the elements by the corresponding sandpile gadgets of fixed size.

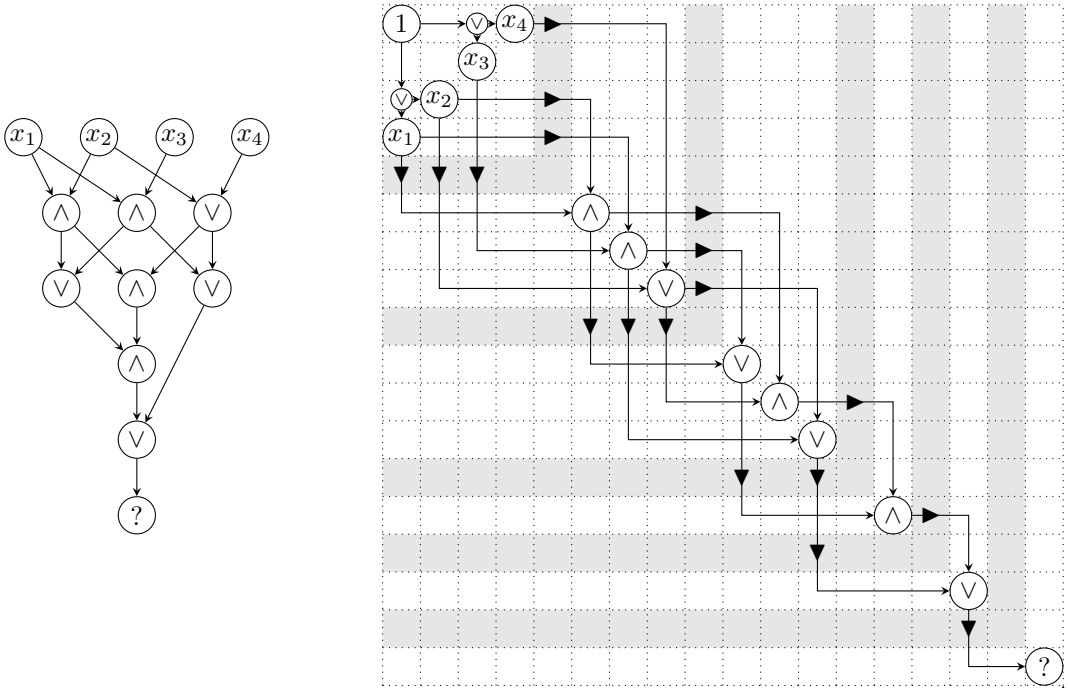


FIGURE 4.3 : Embedding of a **MCVP** instance of fan-in and fan-out two (left) on the grid (right). Diodes (represented as ▶ and ▼) are used systematically between layers in order to enforce the flow of information (and prevent it to go backward).

For neighborhoods 135 and 143 we eplain how to build : *background*, *constants* 0 and 1, west-east and north-south *wires*, west-south and north-east *turns*, *signal-duplications*, *and* gates, *or* gates, *crossover* gates and *diodes*. All these elements are of size 10×10, and are presented for both sandpile CA on Figure 4.4 : signals 1 are chains of topplings, signals 0 are stable, and gates plug neatly along their sides. A single constant 1 gate is used, whose central cell is reset to 3 and is the vertex p on which a sand grain will be added. It triggers the avalanche process (chain reaction

of topplings), whose signal is duplicated (using *or* gates) so as to reach the variable gadgets x_i. These variable gadgets are either *or* gates (for inputs set to 1 in the **MCVP** instance) or constant zero gates (for inputs set to 0). The questioned cell is a wire, with one vertex containing 3 sand grains being the vertex q on which $\mathcal{N}$**-PRED** asks whether it topples. The answer is equivalent to having the circuit outputing 1. The reduction can be performed in logarithmic space, because it amounts to computing positions in $\mathbb{Z}^2$ for each of the elements of the circuit. ■

In the following subsections we prove that none of the remaining 39 equivalence classes, *i.e.*, no other sub-neighborhood within the Moore neighborhood, admit a crossover gate.

4.2.2 Crossover impossibility : crossing constraint

In order to prove impossibility of crossover gates, our proofs will proceed by contradiction, assuming a crossover gate exists. In this subection we give a general setup that will be used in most proofs. We take a result from NGUYEN et al. 2018, proving that if a crossover gate exists, then there also exists a crossover gate where the two firing graphs have disjoint vertex sets (they have no cell in common). In ibid. it is stated for Eulerian graphs, a property verified by the sandpile CA as introduced in the present work.

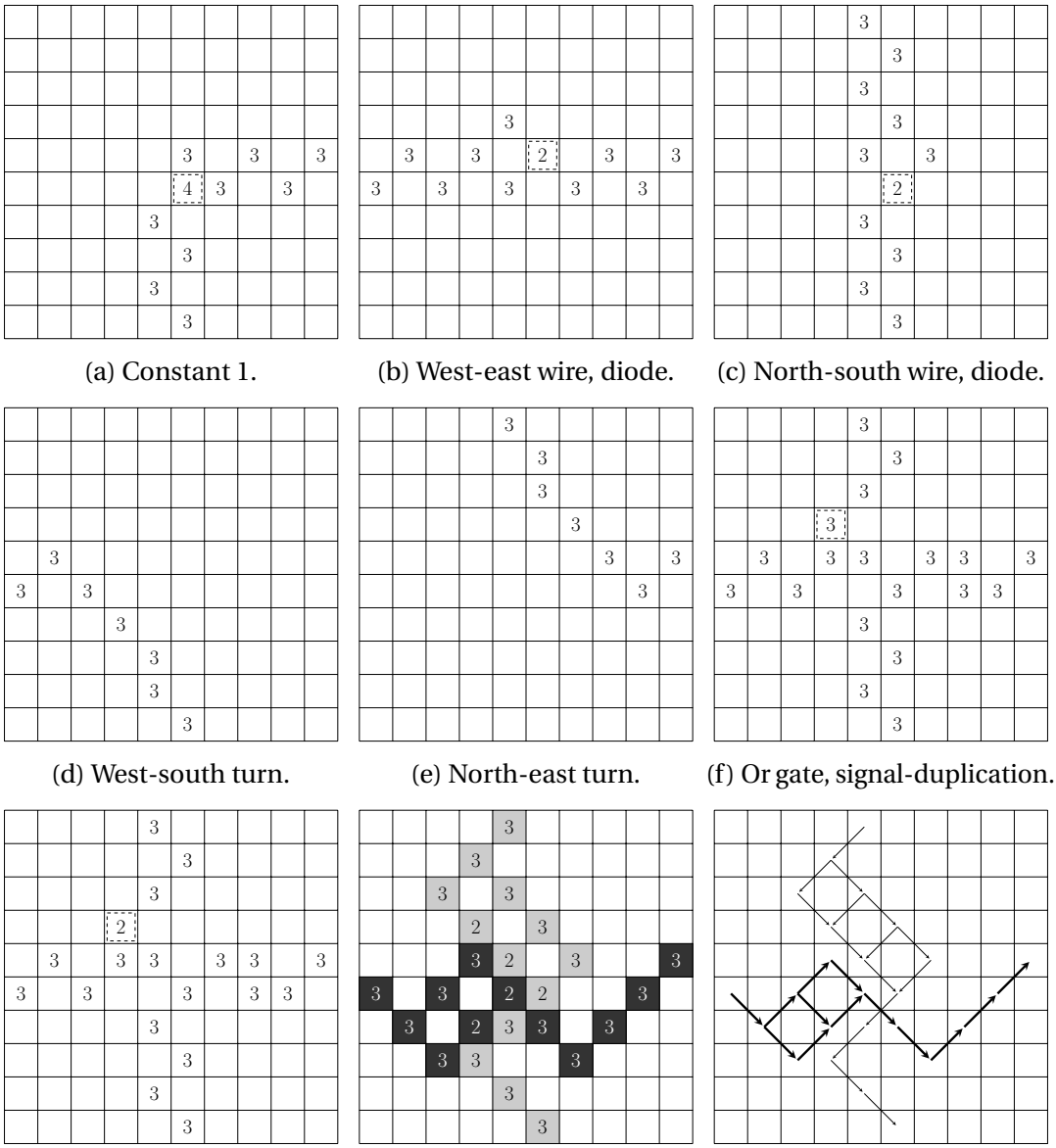
Lemma 4.1 (ibid.)**.** *If a sandpile CA $\mathcal{N}$ admits a crossover gate, then it also admits a crossover gate g with firing graphs $G^g_{ns} = (V_{ns}, A_{ns})$ and $G^g_{we} = (V_{we}, A_{we})$, such that $V_{ns} \cap V_{we} = \emptyset$.*

Because of Lemma 4.1, from now on, everytime time we refer to firing graphs of a crossover gate, we assume that they are disjoint. The following Lemma is a very basic consideration used intensively in the coming proofs, describing the initial setup for further reasonings.

Lemma 4.2 (Crossing constraint)**.** *If $\mathcal{N} \subset_{fin} \mathcal{N}_m$ admits a crossover gate g, then there exists $(i, j) \in \mathbb{Z}^2$ such that the two firing graphs G^g_{ns}, G^g_{we} verify, up to rotation :*

- $(i, j), (i+1, j+1) \in V(G^g_{ns})$ *and* $((i+1, j+1), (i, j)) \in E(G^g_{ns})$,
- $(i+1, j), (i, j+1) \in V(G^g_{we})$ *and* $((i, j+1), (i+1, j)) \in E(G^g_{we})$.

Proof. Since at some point at least one edge of G^g_{ns} has to go over at least one edge of G^g_{we}, and considering the fact that $\mathcal{N} \subseteq \mathcal{N}_{\mathbb{m}}$, the only way of crossing is in a diagonal fashion. Otherwise, $\mathcal{N}$ is not a sub-neighborhood of $\mathcal{N}_{\mathbb{m}}$. ■



(a) Constant 1. (b) West-east wire, diode. (c) North-south wire, diode.

(d) West-south turn. (e) North-east turn. (f) Or gate, signal-duplication.

(g) And gate. (h) Crossover gate. (i) Crossover firing graphs.

FIGURE 4.4 : Wiring toolkit and gates for 135. For 143, all non-empty cells must have one additionnal grain. No number means 0 grains (empty). Important cells are highlighted. The background and contant 0 gates are simply full of 0s. Diodes and wires coincide. Note that the constant 1 gate sends its signal to both east and south. Gates take their inputs from west and north then send their output to both east and south. The crossover gate is bicolored relative the the *west to east* and *north to south* signals, whose firing graphs are also presented (last figure).

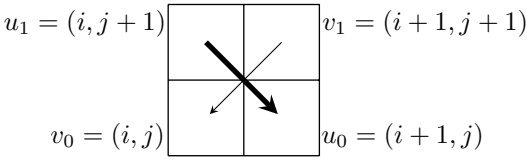


FIGURE 4.5 : Illustration of the crossing constraint (Lemma 4.2), up to rotation. One firing graph with plain arcs and vertices v_i, the other one with bold arcs and vertices u_i (effective causalities). All subsequent figures will use the same convention, with dashed arrows usually representing in-neighbors (potential causality).

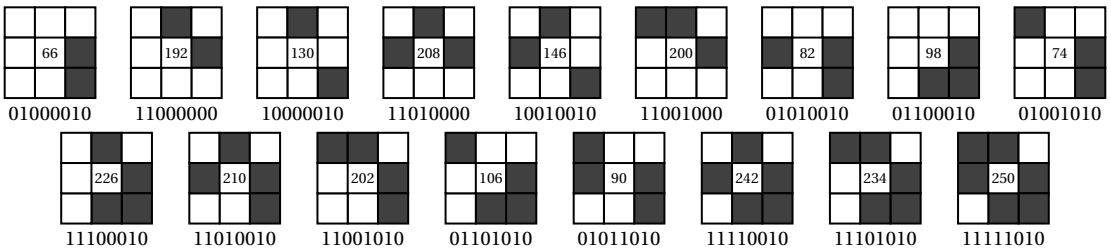
In the proofs, we will denote the four cells given by Lemma 4.2 as $v_0 = (i, j)$, $u_0 = (i+1, j)$, $u_1 = (i, j+1)$, $v_1 = (i+1, j+1)$, as illustrated on Figure 4.5. Besides, depending of the orientation of the crossing contraint, we say that a crossover gate g with firing graphs G^g_{ns} and G^g_{we}, can cross :

- downwards, if $(u_1, u_0) \in E(G^g_{ns})$ and $(v_1, v_0) \in E(G^g_{we})$;
- rightwards, if $(u_1, u_0) \in E(G^g_{ns})$ and $(v_0, v_1) \in E(G^g_{we})$;
- upwards, if $(u_0, u_1) \in E(G^g_{ns})$ and $(v_0, v_1) \in E(G^g_{we})$; or
- leftwards if $(u_0, u_1) \in E(G^g_{ns})$ and $(v_1, v_0) \in E(G^g_{we})$.

Most of our reasonings will be based on the fact that, if some vertex v from one firing graph is an in-neighbor of vertex u from the other firing graph, then vertex u must have at least two predessors in its firing graph (otherwise it is also fired in the other firing graph, contradicting our assumption that the two firing graphs are disjoint). The arguments applies to any number of in-neighbors from the other firing graph : a vertex must have strictly more predecessors from its own firing graph than in-neighbors from the other firing graph.

4.2.3 Crossover impossibility : planar neighborhoods

Neighborhoods studied in this subsection :



Among the 39 remaining non-equivalent neighborhoods, a first pruning can be done looking at those neighborhoods that describe a planar graph. Indeed, the crossover impossibility is a corollary of Lemma 4.1 in this case. We first introduce formally the sandpile graph corresponding to a given neighborhood, which is an infinite graph representing the potential chain of reactions that can occur in the sandpile dynamics of a given neighborhood.

Definition 4.4 (Sandpile graph)**.** Given a neighborhood $\mathcal{N} \subset_{\texttt{fin}} \mathbb{Z}^2$, its *sandpile graph* has vertex set $\mathbb{Z}^2$, and arcs (v_1, v_2) when $(v_2 - v_1) \in \mathcal{N}$.

Corollary 4.1. *If neighborhood $\mathcal{N}$ has a planar sandpile graph, then it does not admit a crossover gate.*

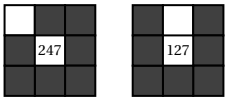
Proof. It follows directly from Lemma 4.1, which states that the two firing graphs (which are finite subgraphs of the sandpile graph) of a crossover gate can have disjoint sets of vertices (cells). If the sandpile graph is planar, then it is impossible to have a path from n to s and a path from w to e without any cell in common. ■

There are 17 non equivalent planar neighborhoods, for which Corollary 4.1 can be applied to deduce the impossibility of crossover.

Theorem 4.3. *Neighborhoods* 66, 192, 130, 208, 146, 200, 82, 98, 74, 226, 210, 202, 106, 90, 242, 234, *and* 250 *are planar, and therefore do not admit a crossover gate.*

4.2.4 Crossover impossibility : too many neighbors

Neighborhoods studied in this subsection :



11110111 01111111

In this subsection we study the two neighborhoods which are missing only one cell among the 8 cells of the Moore neighborhood. Neigborhood 247 misses one cell on the diagonal, and neighborhood 127 misses one cell on the side. Starting from the crossing constraint of Lemma 4.2, the argument in this case is that the predessors of u_1 and v_1 cannot be chosen without interacting too much with each other (via the in-neighbor relationship). For the former we quickly reach an impossibility of crossover, whereas the latter requires substantial case analysis.


11110111 = 247

Theorem 4.4. *Neighborhood* 247 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 can be oriented as a downward or rightward crossing.

Downward crossing. This case is depicted on Figure 4.6. Both cells u_0 and u_1 are in-neighbors of v_1, consequently it needs three predecessors which can be selected among cells $v_2 = (i+2, j+1)$, $v_3 = (i+2, j+2)$, $a = (i+1, j+2)$ and $b = (i, j+2)$. However, if both cells a and b are selected as the predecessors of v_1, then u_1 would have four in-neighbors from the opposite firing graph, then it would need five predecessors which is not possible. Hence, only one of a and b can be a predecessor of v_1, while the other two have to be the cells v_2 and v_3. Now considering that cells v_0, v_1 and a (resp. b) are in-neighbors of u_1, it needs four predecessors in order not to topple in the other firing graph, which can only be cells b (resp. a), $u_2 = (i-1, j+2)$, $u_3 = (i-1, j+1)$, and $u_4 = (i-1, j)$. Since b (resp. a) is an in-neighbor of v_1, this latter needs an extra predecessor, which is not possible by our previous considerations.

Rightward crossing. It is, up to rotation, the same proof. ■

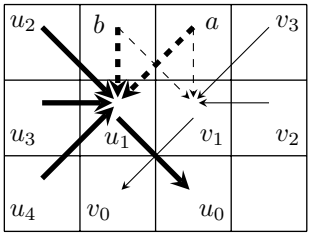


FIGURE 4.6 : Proof of Theorem 4.4 for neighborhood 247, downward crossing. Only one of the cells a and b can be a predecessor of v_1.

Theorem 4.5. *Neighborhood* 127 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 can be oriented as an upward, downward, rightward or leftward crossing.

Upward crossing. For a better understanding of this proof, see Figure 4.7a. Since v_0 and v_1 are in-neighbors of u_0, it needs three predecessors which can be taken among cells $u_2 = (i, j-1)$, $u_3 = (i+2, j-1)$, $u_4 = (i+2, j)$ and $u_5 = (i+2, j+1)$. Let us prove that cells u_4 and u_5 cannot both be predecessors of u_0. If we pick both u_4 and u_5, since they are in-neighbors of v_1, then v_1 would need four predecessors (because u_1 is also an in-neighbor of v_1). One of them is v_0 by the crossing constraint (Lemma 4.2), while the three other ones must be cells $v_2 = (i+2, j+2)$, $v_3 = (i+1, j+2)$ and $v_4 = (i, j+2)$. In turn, since v_1, v_3 and v_4 are in-neighbors of u_1, this latter would need four predecessors. One of them is u_0 by Lemma 4.2, while the other ones can only be cells $u_6 = (i-1, j+2)$, $u_7 = (i-1, j+1)$ and $u_8 = (i-1, j)$. Considering that u_0, u_1, u_7 and u_8 are in-neighbors of v_0, then it would need five predecessors, which is not possible. In simple words, cells u_4 and u_5 cannot be prededecessors of u_0 simultaneously, because it causes a ripple effect which lets v_0 without enough predecessors.

As a consequence, both u_2 and u_3 must to be predecessors of u_0, while the third one has to be selected between u_4 and u_5. For the porpose of this proof, it does not matter if we choose u_4 or u_5, since both are in-neighbors of v_1. Then, without loss of generality, let us pick cell u_4 as the third predecessor of u_0. In this way, v_1 needs only three predecessors. The predecessors of v_1 can be chosen among cells u_5, v_2, v_3 and v_4. However, if we choose u_5, then u_0 would need an extra predecessor, which is impossible. On the other hand, if we choose v_3 and v_4, it will again cause a ripple effect which lets cell v_0 without enough predecessors. Hence, similarly as before, v_2 must be one of the predecessors of v_1, while the other one has to be chosen between v_3 and v_4. Without loss of generality (since they are both in-neighbors of u_1), we choose v_3. Let us take a look at cell u_1. Since v_1 and v_3 are its in-neighbors, it needs three predecessors. We already know that one of them is u_0, while the other two have to be selected among cells v_4, u_6, u_7 and u_8. Nevertheless, if v_4 is one of them, cell v_1 would need an extra predecessor, which is not possible. Furthermore, if both u_7 and u_8 are predecessors of u_1, then v_0 would need five predecessors, which we already know is not possible. In other words, one of the predecessors of u_1 must be cell u_6, while the second one has to be selected between u_7 and u_8. Again, for porpouse of this proof, it does not matter which one we choose, so let us consider that it is u_7. The latter makes

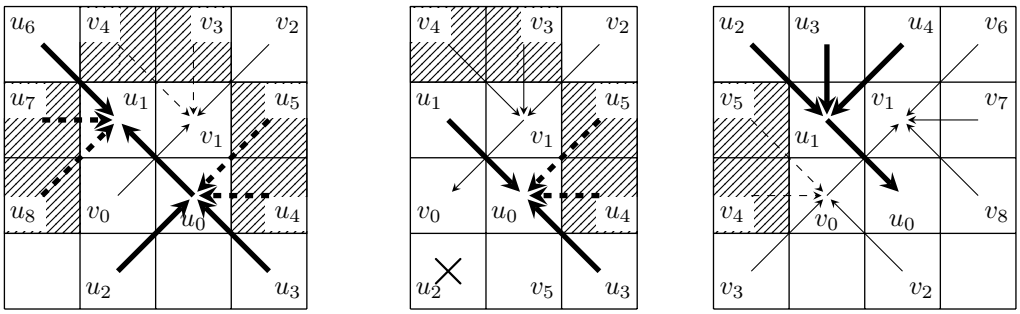
v_0 to have three in-neighbors from the opposite firing graph (cells u_0, u_1 and u_7), thus it needs four predecessors which is not possible.

Downward crossing. We will use the same cell names as in the previous case. First, let us take a look at cell u_0. Since v_0 and v_1 are its in-neighbors, it needs three predecessors. One of them is u_1 by Lemma 4.2, while the other two have to be selected among cells u_2, u_3, u_4 and u_5. Let us assume u_2 is not a predecessor of u_0, *i.e.*, it does not belong to the firing graph of u (see Figure 4.7b). Note that if both cells u_4 and u_5 are in-neighbors of u_0, then the cell v_1 would have three in-neighbors from the opposite firing graph, *i.e.*, it would need four predecessors, which is not possible. Therefore, u_3 must be one of the predecessors of u_0, while the other one has to be chosen between u_4 and u_5. For the purpose of this proof, it does not matter which one we choose, so, without loss of generality, let us consider that it is u_4. Now, considering that cells u_1 and u_4 are in-neighbors of v_1, it needs three predecessors, which have to be cells v_2, v_3 and v_4. Since v_1, v_3 and v_4 are also in-neighbors of u_1, this latter needs four predecessors, which is impossible.

As a consequence, cell u_2 must be a predecessor of u_0. By symmetry, it also implies that cell $v_5 = (i, j-1)$ must be a predecessor of v_0. However, it would generate an upward crossing with arcs (u_2, u_0) and (v_5, v_0), which we already proved is not allowed in a crossover gate for neighborhood 127.

Rightward crossing. In this case, we will not use the same cell numbering as in the previous case. An illustration of this proof is depicted on Figure 4.7c. Let us focus on cell v_0. Since u_0 and u_1 are its in-neighbors, it needs three predecessors, which can be chosen among cells $v_2 = (i+1, j-1)$, $v_3 = (i-1, j-1)$, $v_4 = (i-1, j)$ and $v_5 = (i-1, j+1)$. However, if both cells v_4 and v_5 are its predecessors, since they are also in-neighbors of u_1, then u_1 would need four predecessors (v_1 is also a predecessor of u_1), which is not possible. It means that both cells v_2 and v_3 must be predecessors of v_0, while the third one has to be chosen between cells v_4 and v_5. For the purpose of this proof, it does not matter which one we pick. Without loss of generality, let us consider v_4 as a predecessor of v_0. Since v_1 and v_4 are in-neighbors of u_1, it needs three predecessors which can be selected among cells v_5, $u_2 = (i-1, j+2)$, $u_3 = (i, j+2)$ and $u_4 = (i+1, j+2)$. Nevertheless, selecting v_5 would make v_0 to need an extra predecessor, which is impossible. In other words, the predecessors of u_1 must be cells u_2, u_3 and u_4. Note that u_1, u_3 and u_4 are also in-neighbors of v_1, *i.e.*, cell v_1 needs four predecessors. One of them is v_0 by Lemma 4.2, while the other three have to be cells $v_6 = (i+2, j+2)$, $v_7 = (i+2, j+1)$ and $v_8 = (i+2, j)$. Considering that v_0, v_1, v_7 and v_8 are all in-nighbors of u_8, then this latter needs five predecessors, which is not possible.

Leftward crossing. Considering the symmetry of neighborhood 127, the proof for this case is analogous to the rightward crossing. ■



(a) Upward crossing. Pairs of cells cannot belong to the same firing graph : u_4 with u_5, v_3 with v_4, and u_7 with u_8.

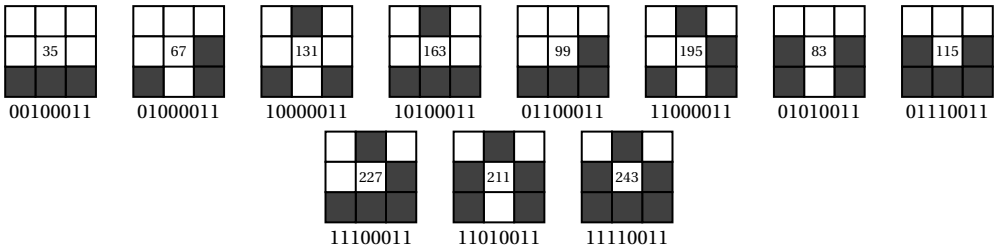
(b) Downward crossing. Assuming u_2 does not belong to the firing graph of u.

(c) Rightward crossing. Only one of v_4 and v_5 can be a predecessor of v_0.

FIGURE 4.7 : Proof of Theorem 4.5 for neighborhood 127.

4.2.5 Crossover impossibility : two diagonals

Neighborhoods studied in this subsection :



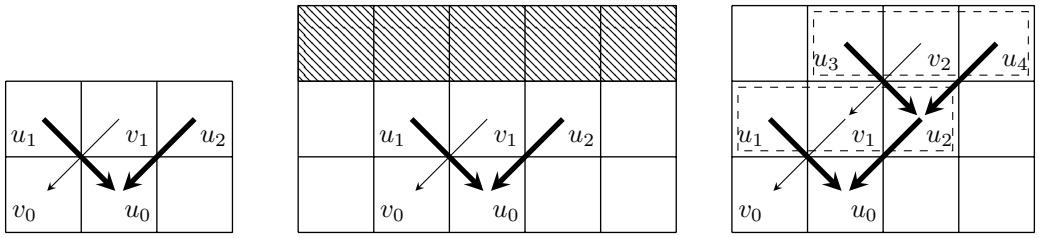
Let us now study the neighborhoods having only two cells in the diagonals, numbered 35, 67, 131, 163, 99, 195, 83, 115, 227, 211 and 243. In all these cases, it is enough to check whether a downward crossing is possible, because the two cells in the diagonals are on positions south-east and south-west (up to the equivalences presented in Table 4.1). The arguments are adapted to each neighborhood, they rely on short case analysis leading either to a direct contradiction, or by induction to the fact that a crossover gate would be of infinite size.

Theorem 4.6. *Neighborhoods* 35 *and* 67 *do not admit a crossover gate.*

Proof. The proofs for the neighborhoods 35 and 67 are very similar. According to the definitions of these neighborhoods, the four cells given by Lemma 4.2 must be oriented downward, as depicted on Figure 4.8a.

One of the cells v_0 or v_1 is an in-neighbor of u_0, hence u_0 requires another predecessor (otherwise it also belongs to the other firing graph), which must be at position $u_2 = (i+2, j+1)$. There is a horizontal pattern u_1, v_1, u_2 of three adjacent cells, alternating between the two firing graphs. We will now show that in both neighborhoods 35 and 67, it implies that there is another such pattern on the row

just above. Since the crossover gate must eventually connect the in-neghbors of these two extremities (*e.g.* u_1 and u_2) to a unique cell on the border, we conclude that the crossover gate cannot have a finite size, which is impossible.



(a) 35, 67. From the crossing constraint, u_2 is a predecessor of u_0.

(b) 35. Two in-neighbors for each of u_1 and u_2 must be choosen in the hatched area.

(c) 35, 67. The contradictory pattern repetition is dashed.

FIGURE 4.8 : Proof of Theorem 4.6 for neighborhoods 35 and 67.

Consider neighborhood 35, and the three potential predecessors v_2 of cell v_1. If $v_2 = (i+1, j+2)$ is just above v_1, then it is an in-neghbor of both u_1 and u_2, as a consequence each of them must have two predecessors. Now observe (see Figure 4.8b) that any choice of these two predecessors for each lead to the fact that v_1 belongs to both firing graphs, hence contradicting Lemma 4.1. Therefore $v_2 = (i, j+2)$ or $v_2 = (i+2, j+2)$. The two cases are symmetric and we consider $v_2 = (i+2, j+2)$ (see Figure 4.8c). Since v_2 is an in-neighbor of u_2, cell u_2 must have two predecessors, which can only be cells $u_3 = (i+1, j+2)$ and $u_4 = (i+3, j+2)$. Cells u_3, v_2, u_4 form a pattern leading to the announced contradiction.

Consider neighborhood 65. Cell u_1 is an in-neighbor of v_1, therefore v_1 must have two predecessors, including $v_2 = (i+2, j+2)$ (see Figure 4.8c). Since v_3 is an in-neighbor of u_2, cell u_2 must have two predecessors, which can only be cells $u_3 = (i+1, j+2)$ and $u_4 = (i+4, j+2)$. Cells u_3, v_3, u_4 form a pattern leading to the announced contradiction. ■

Theorem 4.7. *Neighborhood* 131 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 must be oriented downward, as depicted on Figure 4.9. Cell v_0 is an in-neighbor of u_1, therefore u_1 requires two predecessors which can only be cells $u_2 = (i-1, j+2)$ and $u_3 = (i+1, j+2)$. Similarly, in the other firing graph, cells $v_2 = (i, j+2)$ and $v_3 = (i+2, j+2)$ must be predecessors of v_1. This generates another downward crossing just above the original one, and by induction the crossover gate must be of infinite size, a contradiction. ■

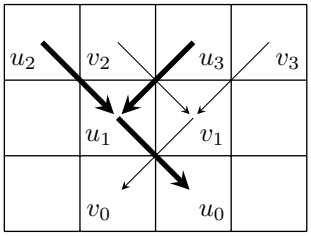


FIGURE 4.9 : Proof of Theorem 4.7 for neighborhood 131.


10100011 = 163

Theorem 4.8. *Neighborhood* 163 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 must be oriented downward, as depicted on Figure 4.10. Let us focus on node u_1. Since v_0 is an in-neighbor of it, u_1 needs at least two predecessors. The available cells are $a = (i-1, j+1)$, $b = (i, j+1)$ and $c = (i+1, j+1)$. This means that at least one of the predecessors is cell b or c. However, it would imply that node v_1 has three or four predecessors (because u_0 is also an in-neighbor of v_1), which is impossible : not enough cells are available in the in-neighborhood of v_1. ■

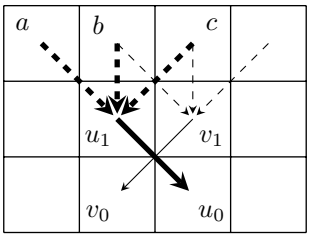


FIGURE 4.10 : Proof of Theorem 4.8 for neighborhood 163. The dashed arrows represent the possible predecessors of cells u_1 and v_1.


01100011 = 99


11000011 = 195

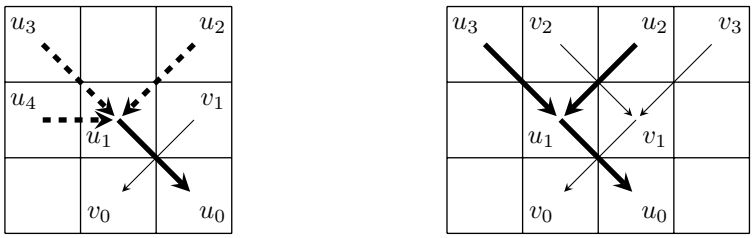
Theorem 4.9. *Neighborhoods* 99 *and* 195 *do not admit a crossover gate.*

Proof. For neighborhood 99, cell u_0 needs three predecessors, but there are only two available. The same thing happens with cell v_1 in the case of the neighborhood 195. ■


01010011 = 83

Theorem 4.10. *Neighborhood* 83 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 must be oriented downward. Cell v_1 is an in-neighbor of u_1, therefore u_1 needs two predecessors in its firing graph. There are three available cells : $u_2 = (i+1, j+2)$, $u_3 = (i-1, j+2)$ and $u_4 = (i-1, j+1)$ (see Figure 4.11a). However, if u_4 is a predecessor of u_1, then cell v_0 will have two in-neighbors in the opossite firing graph (u_0 and u_4), and it would need three predecessors, which is not possible. As a consequence, the only possible predecessors for u_1 are the cells u_2 and u_3. By the symmetry of neighborhood 83, the same reasoning applies to obtain the predecessors of v_1, which can only be the cells $v_2 = (i, j+2)$ and $v_3 = (i+2, j+2)$ (see Figure 4.11b). This creates another downward crossing above the initial one, meaning by induction that the crossover gate is of infinite size, a contradiction. ■



(a) Available predecessors of cell u_1.

(b) Cells u_2 and u_3 (resp. v_2 and v_3) must be predecessors of u_1 (resp. v_1).

FIGURE 4.11 : Proof of Theorem 4.10 for neighborhood 83.

Theorem 4.11. *Neighborhood* 115 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 must be oriented downward, as depicted on Figure 4.12. Cells v_0 and v_1 are in-neighbors of u_0, hence cell u_0 needs at least three predecessors in the firing graph, which can only be cells u_1, $u_2 = (i+2, j+1)$ and $u_3 = (i+2, j)$. Since u_1 and u_2 are in-neighbors of v_1, cell v_1 needs three predecessors, and the only cells availables for this are the three cells above it : $v_2 = (i, j+2)$, $v_3 = (i+1, j+2)$, $v_4 = (i+2, j+2)$. In turn, this causes cell u_2 to need three predecessors too, but it does not have enough available in-neighbors. ■

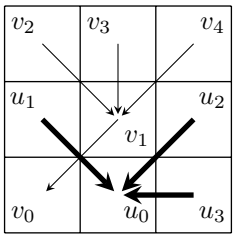


FIGURE 4.12 : Proof of Theorem 4.11 for neighborhood 115.

Theorem 4.12. *Neighborhood* 227 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 must be oriented downward, as depicted on Figure 4.13. Both cells u_0 and u_1 are in-neighbors of cell v_1, consequently cell v_1 needs at least three predecessors, which can only be cells $v_2 = (i, j+2)$, $v_3 = (i+1, j+2)$ and $v_4 = (i+2, j+2)$. Since v_2 and v_3 are in-neighbors of u_1, the latter must have three predecessors, which is not possible. ■

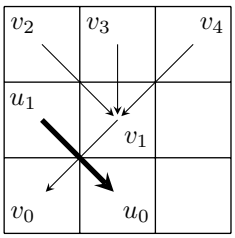


FIGURE 4.13 : Proof of Theorem 4.12 for neighborhood 227.


11010011 = 211

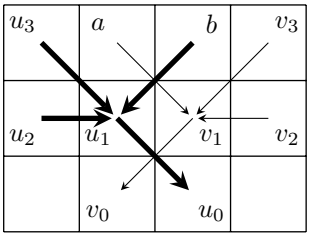

11110011 = 243

Theorem 4.13. *Neighborhoods* 211 *and* 243 *do not admit a crossover gate.*

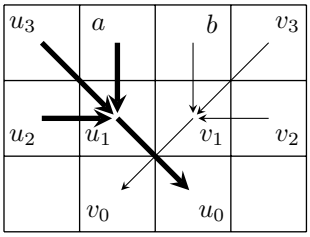
Proof. The four cells given by Lemma 4.2 must be oriented downward, as depicted on the two cases of Figure 4.14. Let us focus on cell u_1. Since v_0 and v_1 are both in-neighboors of u_1, it needs at least three predecessors.

For neighborhood 211 these three predecessors of u_1 can only be cells $u_2 = (i-1, j+1)$, $u_3 = (i-1, j+2)$ and $a = (i+1, j+2)$ (see Figure 4.14a). By the symmetry of neighborhood 211, the same reasonong applies to v_1, and its only three available predecessors are $v_2 = (i+2, j+1)$, $v_3 = (i+2, j+2)$ and $b = (i, j+2)$. Now, observe that cells v_1, v_2 and b are all in-neighbors of a. Cell a therefore requires four predecessors, which is not possible.

In the case of the neighborhood 243 (see Figure 4.14b), the three predecessors of cell u_1 can be selected among cells u_2, u_3, a and b. Hence at least one of a and b must be chosen. Since a and b are both in-neighbors of v_1, the latter needs at least four predecessors (because u_0 and u_1 are also in-neighbors of v_1), which is again not possible. ■



(a) 211, 243 : Cell a (resp. b) is a predecessor of v_1 (resp.u_1).

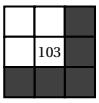


(b) 243 : Cell a (resp. b) is a predecessor of u_1 (resp. v_1).

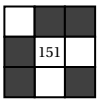
FIGURE 4.14 : Proof of Theorem 4.13 for neighborhoods 211 and 243.

4.2.6 Crossover impossibility : three or four diagonals

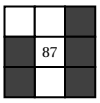
Neighborhoods studied in this subsection :



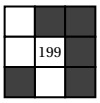
01100111



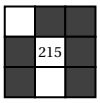
10010111



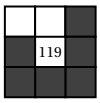
01010111



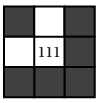
11000111



11010111

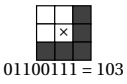


01110111



01101111

In this subsection we study neighborhoods having three or four diagonal cells, hence for which the crossing constraint (Lemma 4.2) has two to four possible orientations. We continue to present proofs by contradiction, following case analysis. Neighborhood 111 requires a somewhat long development. For neighborhoods 95 and 39, we present a different approach in the next subsection.



Theorem 4.14. *Neighborhood* 103 *does not admit a crossover gate.*

Proof. The four cells given by Lemma 4.2 can have two orientations : downward or rightward crossing.

Downward crossing. See Figure 4.15. First notice that v_0 and v_1 are in-neighbors of cell u_0, hence it needs two extra predecessors, which can only be cells $u_2 = (i, j-1)$ and $u_3 = (i+2, j+1)$. At the same time, u_1 is an in-neighbor of cell v_1, hence it needs two predecessors, that have to be selected among cells $v_2 = (i, j+2)$, $v_3 = (i+1, j+2)$ and $v_4 = (i+2, j+2)$. Consequently, at least one of v_2 or v_3 must be a predecessor of v_1. The latter means that cell u_1 also needs two predecessors, which can in turn be chosen among cells v_2, v_3, $u_4 = (i-1, j)$, $u_5 = (i-1, j+1)$ and $u_6 = (i-1, j+2)$. However, chosing v_2 or v_3 makes v_1 need another predecessor, but it already ran out of possibilities. Then the remaining cells are u_4, u_5 and u_6, and at least one of u_4 and u_5 must be a predecessor of u_1. If we chose u_4 (resp. u_5), then the cell v_0 has two in-neighbors from the opposite firing graph, and it needs three predecessors, which can only be cells v_1, v_5 and u_5 (resp. u_4). This implies that u_1 needs an additional predecessor, but the only remaing available cells are v_2 or v_3, which we already showed cannot be selected as predecessors of u_1.

Rightward crossing. Up to rotation, the rightward crossing has the same proof of impossibility as the downward crossing. ■

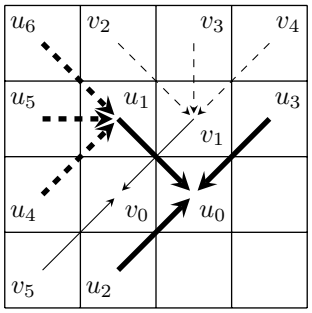
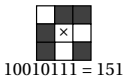


FIGURE 4.15 : Proof of Theorem 4.14 for neighborhood 103.

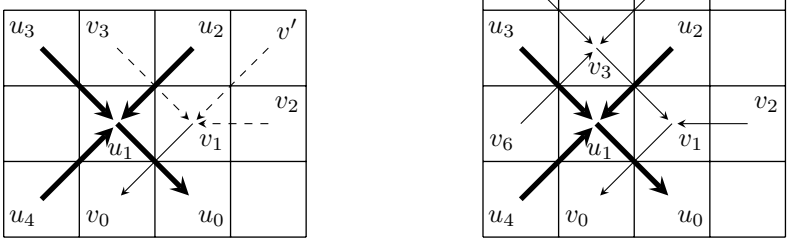


Theorem 4.15. *Neighborhood* 151 *does not admit a crossover gate.*

Proof. **Downward crossing.** Let us place the four cells given by Lemma 4.2 as depicted in Figure 4.16a. First of all, notice that cell u_1 has v_0 and v_1 as in-neighbors, hence it needs three predecessors which can be only cells $u_2 = (i+1, j+1)$, $u_3 = (i-1, j+1)$ and $u_4 = (i-1, j-1)$. On the other hand, cell u_0 is an in-neighbor of v_1, consequently v_1 needs at least two predecessors. These predecessors can be chosen among cells $v_2 =$

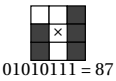
$(i+2, j+1)$, $v_3 = (i, j+1)$ and $v' = (i+2, j+1)$. However, if we take v' as a predecessor of v_1, then cell u_2 would need three predecessors, which is not possible. Hence, the predecessors of v_1 must be cells v_2 and v_3. Now let us focus on cell v_3. Both cells u_2 and u_3 are in-neighbors of v_3, consequently it needs three predecessors, that have to be cells $v_4 = (i+1, j+3)$, $v_5 = (i-1, j+3)$ and $v_6 = (i-1, j+1)$. Finally, cell v_6 needs three predecesors because u_1 and u_4 are in-neighbors of it. Nevertheless, v_6 does not have enough available in-neighbors (see Figure 4.16b).

Rightward crossing. Same proof up to rotation. ■



(a) Available predecessors of v_1. (b) Predecessors of cells v_1 and v_3.

FIGURE 4.16 : Proof of Theorem 4.15 for neighborhood 151.



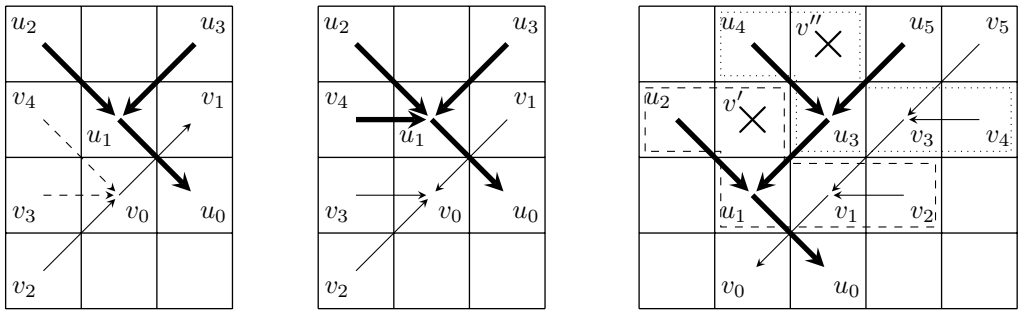
Theorem 4.16. *Neighborhood* 87 *does not admit a crossover gate.*

Proof. **Rightward crossing.** This case is depicted on Figure 4.17a Let us first focus on cell v_0. Since u_0 is its in-neighbor, it needs two predecessors which can be cells $v_2 = (i-1, j-1)$, $v_3 = (i-1, j)$, and $v_2 = (i-1, j+1)$. If cells v_3 and v_4 are both predecessors of v_0, considering that they are also in-neighbors of u_1, then u_1 would need four predecessors (v_1 is also an in-neighbor), which is not possible. In other words, v_2 has to be one of the predecessors of v_0, while the second one has to be chosen between v_3 and v_4. For the purpose of this proof, there is no difference if we choose v_3 or v_4, then, without loss of generality, let us pick v_3. Now, u_1 has two in-neighbors from the opposite firing graph : v_1 and v_3, and consequently it needs three predecessors that can only be cells v_4, $u_2 = (i-1, j+2)$ and $u_3 = (i+1, j+2)$. However, since v_4 is an in-neighbor of v_0, it follows that v_0 needs an extra predecessor, which is not possible.

Downward crossing. For this case, we use the same cell numbering as before. This proof is depicted in Figure 4.17b. First of all, let us prove that cells u_2 and u_3 have to be predecessors of u_1. Initially, u_1 has only one in-neighbor from the oppossite firing graph, which is cell v_1, then it needs two predecessors. As we want to prove that both u_2 and u_3 have to be predecessors of u_1, let us show what happens if we pick one of v_3 or v_4. Considering that v_3 (resp. v_4) and u_0 are in-neighbors of v_0, the latter therefore needs three predecessors. One of them is v_1 by Lemma 4.2, while the other two can only be cells v_2 and v_4 (resp. v_3). Note that v_4 (resp. v_3) is also an in-neighbor of u_1, hence now it needs an extra predecessor. In other words, selecting one of v_3 and v_4 as a predecessors of u_1, just makes u_1 need an extra predecessor. Subsequently, we

can ensure that u_2 and u_3 are predecessors of u_1 since there are the only remaining in-neighbors.

Now, we prove that no matter what predecessors are chosen, the following happens (see Figure 4.17c) : the initial cross implies either another cross, or a given pattern P; then in turn the pattern P implies either another cross, or another pattern P. Inasmuch as the crossover gate has to be of finite size, this leads to contradiction. Since u_1 is an in-neighbor of v_1, cell v_1 needs two predecessors. Its available in-neighbors are cells $v' = (i, j+2)$, $v_2 = (i+2, j+1)$ and $v_3 = (i+2, j+2)$. However, if v' is a predecessor, then it would be generating another cross above the original one. Therefore, the only option left is to consider cells v_2 and v_3 as the predecessors of v_1. This is the first appearance of the pattern, which consists in cells u_2, v', u_1, v_1 and v_2. Considering that v_3 is an in-neighbor of u_3, then u_3 needs two predecessors, which can only be cells $u_4 = (i, j+3)$ and $u_5 = (i+2, j+3)$ (cell v' cannot be chosen since it makes v_1 to exhaust its available predecessors). Since in turn u_3 is an in-neighbor of v_3, cell v_3 needs two predecessors which can be chosen among cells $v'' = (i+1, j+3)$, $v_4 = (i+3, j+2)$ and $v_5 = (i+3, j+3)$. If v'' is chosen, then it generates another cross, therefore it is necessary to select v_4 and v_5. Here the pattern appears again, formed by cells u_4, v'', u_3, v_3 and v_4. ■



(a) Rightward crossing. Cell v_0 exhausts its in-neighbors.

(b) Downward crossing. Selecting v_4 as a predecessor of u_1.

(c) Downward crossing. An infinite pattern appears. The first occurrence is dashed, the second one is dotted.

FIGURE 4.17 : Proof of Theorem 4.16 for neighborhod 87.

Theorem 4.17. *Neighborhood* 199 *does not admit a crossover gate.*

Proof. **Downward crossing.** Cells u_0 and u_1 are in-neighbors of v_1, i.e., v_1 needs three predecessors, but it does not have enough available in-neighbors. Therefore a downward crossing is impossible.

Rightward crossing. The cells and arcs used for this proof are depicted in Figure 4.18. Cells u_0 and u_1 are in-neighbors of v_1, therefore it needs three predecessors which can only be cells v_0, $v_2 = (i+2, j+2)$ and $v_3 = (i, j+2)$. On the other hand, cell v_0 is an in-neighbor of u_1, hence u_1 needs two predecessors. The in-neighbors available are cells $u' = (i+1, j+2)$, $u_2 = (i-1, j+2)$, $u_3 = (i-1, j+1)$, $u_4 = (i-1, j)$. If we choose u' as one of the predecessors, we would be generating a downwawrds crossing with arcs

(u', u_1) and (v_3, v_1), but this is impossible for neighborhood 199 by the previous case. Additionaly, if we choose both u_3 and u_4, then cell v_0 would need three predecessors which is not possible. In other words, one of the predecessors of u_1 has to be u_2, while the other one has to be either u_3 or u_4. Since u_3 and u_4 are in-neighbors of v_0, it needs two predecessors and $v_4 = (i-1, j-1)$ together with $v_5 = (i, j-1)$ are the only in-neighbors available. Cells u_3 and u_4 cannot be predecessors of v_0, because is would follow that u_1 needs an extra predecessor, which can only be contradictory by the preceding case analysis.

Now let us focus on cell u_0. It has two in-neighbors from the opposite firing graph, which are cells v_0 and v_5, and consequently it needs three predecessors, which can only be cells u_1, $u_5 = (i+1, j-1)$ and $u_6 = (i+2, j+1)$. Remark that we can consider without loss of generality that v_1 has a successor. We deduce that the successor of v_1 can only be $v' = (i+2, j)$. However this creates a downward crossing with arcs (v_1, v') and (u_6, u_0), which is impossible by the previous case. ■

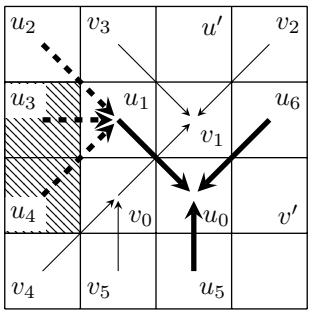


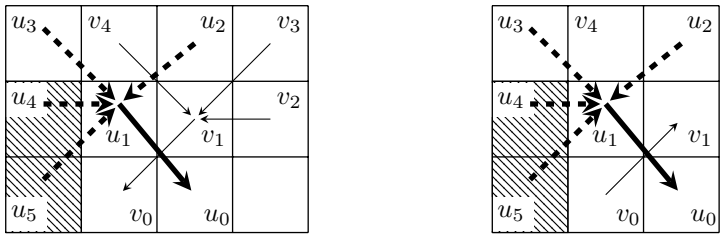
FIGURE 4.18 : Proof of Theorem 4.17 for neighborhood 199. Dashed arrows show the possible predecessors of u_1, however only one from the hatched area can be selected.

Theorem 4.18. *Neighborhood* 215 *does not admit a crossover gate.*

Proof. **Downward crossing.** The cells for this proof are depicted in Figure 4.19a. Note that cells u_0 and u_1 are both in-neighbors of cell v_1, hence it needs three predecessors. The only available cells for being predecessors of v_1 are cells $v_2 = (i+2, j+1)$, $v_3 = (i+2, j+2)$ and $v_4 = (i, j+2)$. Now, let us consider cell u_1. It has cells v_0 and v_1 as in-neighbors, hence it also needs three predecessors which can be selected among cells $u_2 = (i+1, j+2)$, $u_3 = (i-1, j+2)$, $u_4 = (i-1, j+1)$, and $u_5 = (i-1, j)$. However, if both cells u_4 and u_5 are predecessors of u_1, then cell v_0 would need four predecessors, which is not possible. The latter means that u_2 and u_3 have to be predecessors of u_1, while the third predecessor has to be selected between cells u_4 and u_5. Nevertheless, observe that v_1, v_3 and v_4 are all in-neighbors of cell u_2, therefore u_2 needs four predecessors, which is also impossible.

Rightward crossing. The cells for this proof are depicted in Figure 4.19b. Analogously to the rightward crossing, u_2 and u_3 have to be two of the three predecessors of u_1, and the third one has to be selected between u_4 and u_5. Now

let us take a look at cell v_0. It has two in-neighbors from the opossite firing graph : the first one is either u_4 or u_5, and the other is u_0. Consequently, cell v_0 needs three predecessors, which is not possible. ■



(a) Downward crossing. (b) Rightward crossing.

FIGURE 4.19 : Proof of Theorem 4.18 for neighborhood 215.

Theorem 4.19. *Neighborhood* 119 *does not admit a crossover gate.*

Proof. **Downward crossing.** Note that cells v_0 and v_1 are in-neighbors of u_0, therefore it needs three predecessors. One of them is u_1 by the crossing constraint (Lemma 4.2). The other two can be selected among $u_2 = (i, j-1)$, $u_3 = (i+2, j)$ and $u' = (i+2, j+1)$. However, if u' is a predecessor of u_0 (see Figure 4.20a), then v_1 would have two in-neighbors from the opposite firing graph (cells u_1 and u'). It means that v_1 would need three predecessors, which can only be cells $v_2 = (i+2, j+2)$, $v_3 = (i+1, j+2)$ and $v_4 = (i, j+2)$. In this situation, cell u_1 has three in-neighbors from the opposite firing graph : cells v_1, v_3 and v_4. It follows that it needs four predecessors, which is not possible. In other words, u' cannot be a predecessor of u_0.

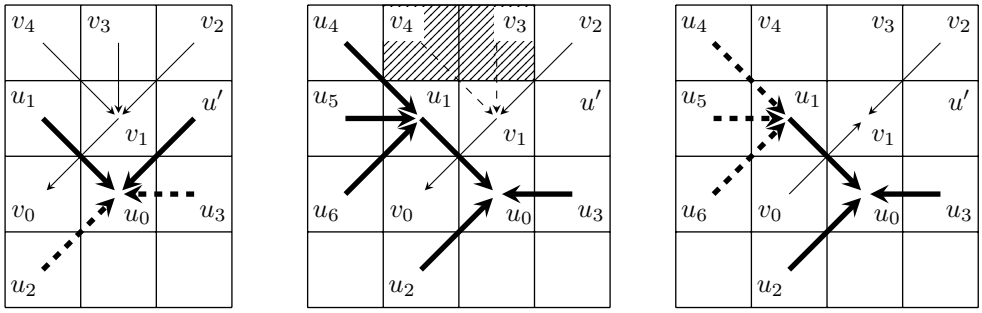
We deduce that we have to pick cells u_2 and u_3 as predecessors of u_0 (see Figure 4.20b). In that case, v_1 needs only two predecessors. Since v_3 and v_4 cannot simultaneously be predecessors of v_1 (otherwise u_1 would need four predecessors again), only one of them has to be selected, while the second one must be cell v_2. Note that cell u' cannot be selected as a predecessor of v_1, since it would make u_0 to need an extra predecessor, which is not posible. Now cell u_1 needs three predecessors, which can only be cells $u_4 = (i-1, j+2)$, $u_5 = (i-1, j+1)$ and $u_6 = (i-1, j)$. Note that v_3 and v_4 cannot be selected as predecessors of u_1, because it would make cell v_1 need an extra predecessor, nevertheless there are no more available in-neighbors. Since u_0, u_1, u_5 and u_6 are all in-neighbors of v_0, cell c_0 needs five predecessors, which is not possible.

Rightward crossing. The proof for this case is depicted in Figure 4.20c. Analogously to the downward crossing, cell u_0 needs two more predecessors and u' cannot be chosen as one of them by the exact same reasoning. As a consequence, we have to take cells u_1, u_2 and u_3 as predecessors of u_0. Now cell v_1 needs two predecessors, one of them is v_0 while the other can be chosen among cells v_2, v_3 and v_4. Let us study these cases separately.

If we choose v_3 (resp. v_4), then u_1 needs three predecessors as both v_1 and v_3 (resp. v_4) are its in-neighbors. These three predecessors have to be chosen among the

cells v_4 (resp. v_3), u_4, u_5 and u_6. The latter implies that at least one of u_5 and u_6 is a predecessor of u_1. Since u_0, u_5 and u_6 are in-neighbors of v_0, then it needs at least three predecessors which is not possible.

On the other hand, if we choose v_2 then u_1 needs only two predecessors (as v_1 is its in-neighbor). These two predecessors have to be taken from cells v_3, v_4, u_4, u_5 and u_6. Note that if we take v_3 (resp. v_4), then cell v_1 would need an extra predecessor, which can only be cell v_4 (resp. v_3), what in turn would make u_1 to need an extra predecessor. In other terms, selecting v_3 or v_4 is optional in this case and, without loss of generality, we will consider only cells u_4, u_5 and u_6, as possible predecessors for u_1. The latter implies that at least one of u_5 and u_6 is a predecessor of u_1. Since u_0, u_5 and u_6 are in-neighbors of v_0, then it needs at least three predecessors which is not possible. ■



(a) Cell u' cannot be chosen as a predecessor of u_0.

(b) Downward crossing. Cell v_0 does not have enough in-neighbors.

(c) Rightward crossing. Cell v_0 does not have enough in-neighbors.

FIGURE 4.20 : Proof of Theorem 4.19 for neighborhood 119.

Theorem 4.20. *Neighborhood* 111 *does not admit a crossover gate.*

Proof. **Upward crossing.** The cells for this proof are depicted in Figure 4.21a. First of all, notice that cell u_0 needs three predecessors. They can only be cells $u_2 = (i, j-1)$, $u_3 = (i+2, j-1)$ and $u_4 = (i+2, j+1)$. On the other hand, v_0 needs two predecessors, which can be selected among cells $v_2 = (i-1, j+1)$, $v_3 = (i-1, j)$, $v_4 = (i-1, j-1)$ and $v' = (i+1, j-1)$. We will prove that v' has to be a predecessor of v_0, which creates another upward crossing just below the original one, with arcs (v', v_0) and (u_2, u_0). By induction, this new upward crossing creates another one, etc, and the crossover gate must have an infinite size, which is absurd. We prove the latter by contradiction, assuming v' is not a predecessor of v_0. In such a case, both predecessors of v_0 have to be chosen among v_2, v_3 and v_4. Note that if we choose v_3 and v_4, they make u_2 to need four predecessors, which is not possible. In other words, one predecessor of v_0 must be v_2, while the second predecessor is v_3 or v_4. If we choose v_3 (resp. v_4), then cell u_2 needs three predecessors. The predecessors of u_2 can only be cells v_4 (resp. v_3), $u_5 = (i-1, j-2)$ and $u_6 = (i+1, j-2)$. Considering that v_4 (resp. v_3) is an in-neighbor of v_0, then v_0 needs an extra predecessor and the only remaining in-neighbor of v_0 is v'.

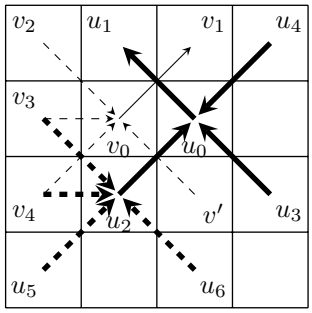
Leftward crossing. This case is symmetric to the upward crossing.

Downward crossing. We rename the cells for this case, starting with u_0, u_1, v_0, v_1 given by Lemma 4.2 as depicted in Figure 4.21b. Both v_0 and v_1 are in-neighbors of u_0, hence it needs three predecessors. One of them is u_1, while the other two have to be chosen among cells $u_2 = (i, j-1)$, $u_3 = (i+2, j-1)$ and $u' = (i+2, j+1)$. We first proceed by contradiction, assuming u' is a prodecessor of u_0, in order to prove that this is impossible (see Figure 4.21b). Let us consider cell v_1. It has one in-neighbor from the opposite firing graph (cell u_1), *i.e.*, it needs two predecessors which can be selected among cells $v' = (i+2, j)$, $v_2 = (i+2, j+2)$, $v_3 = (i+1, j+2)$ and $v_4 = (i, j+2)$. We can immediatly discard cell v' since it creates a leftward crossing with arcs (v', v_1) and (u', u_0), which we already proved impossible. Moreover, cells v_2 and v_3 cannot both be predecessors of v_1. This is because they are, together with v_1, in-neighbors of u', making it to need four predecessors, which is not possible. It follows that one of the in-neighbors of v_1 must be v_4, while the second one can be chosen between v_2 and v_3. If v_2 (resp. v_3) is chosen, then cell u' needs three predecessors that can only be cells $u'' = (i+3, j)$, $u''' = (i+3, j+2)$ and v_3 (resp. v_2). Since v_3 (resp. v_2) is also an in-neighbor of v_1, cell v_1 needs an extra predecessor and the only available in-neighbor is v' which, as we already mentioned, generates a leftward crossing. With this we proved that u' cannot be a predecessor of u_0.

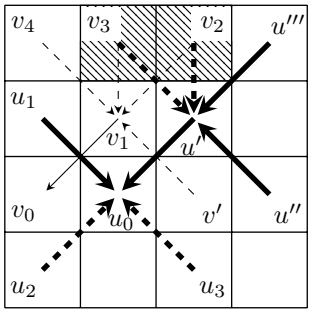
The latter implies that the predecessors of u_0 are u_2 and u_3, so let us continue the reasoning with this hypothesis (see Figure 4.21c). Apart from u_2 and u_3, we rename again the cells for a better understanding of the proof. Note that u_1 is an in-neighbor of v_0, hence v_0 needs an extra predecessor which can be selected among cells $v_2 = (i+1, j-1)$, $v_3 = (i-1, j+1)$, $v_4 = (i-1, j)$ and $v_5 = (i-1, j)$. We now prove that none of them can be the extra predecessor of v_0, thus the downward crossing is not possible. Let us proceed by cases.

Case v_2. It generates an upward crossing with arcs (v_2, v_0) and (u_2, u_0), which has already been proven impossible.

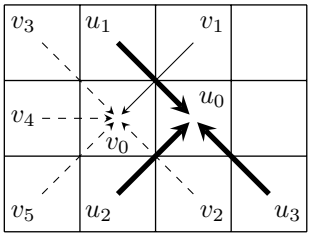
Case v_3. Since v_3 is an in-neighbor of u_1, cell u_1 needs two predecessors which can be selected among cells v_4, $u_4 = (i-1, j+2)$, $u_5 = (i, j+2)$ and $u_6 = (i+1, j+2)$. We again proceed by exhaustion.



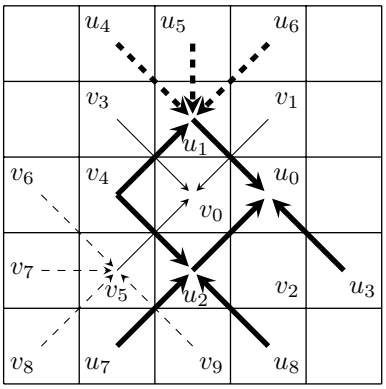
(a) Upward crossing.



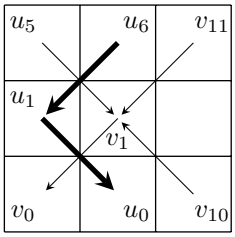
(b) Downward crossing. If u' is a predecessor of u_0.



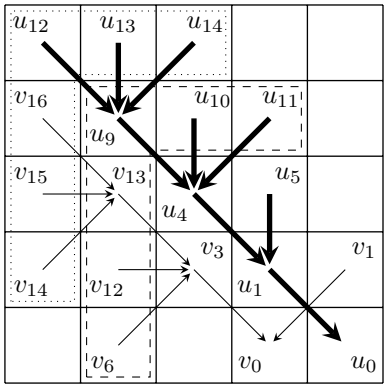
(c) Downward crossing. All possible predecessors of v_0.



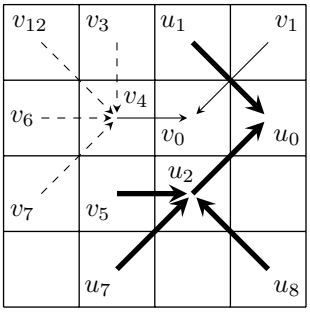
(d) Downward crossing (case v_3). If v_4 is a predecessor of u_1.



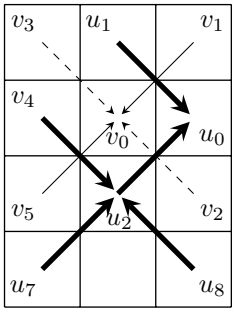
(e) Downward crossing (case v_3). If u_6 is a predecessor of u_1.



(f) Downward crossing (case v_3). If u_4 and u_5 are predecessors of u_1.



(g) Downward crossing (case v_4).



(h) Downward crossing (case v_5).

FIGURE 4.21 : Proof of Theorem 4.20 for neighborhood 111.

First, let us prove that v_4 cannot be a predecessor of u_1 (see Figure 4.21d). In such a case, cell v_0 needs an extra predecessor which can only be cell v_5 (selecting v_2 generates an upward crossing). Since v_0 and v_5 are in-neighbors of u_2, it needs three predecessors and the only remaining in-neighbors are the cells u_4, $u_7 = (i-1, j-2)$ and $u_8 = (i+1, j-2)$. At the same time, since v_4 is an in-neighbor of v_5, cell v_5 requires two predecessors which can be selected among cells $v_6 = (i-2, j)$, $v_7 = (i-2, j-1)$, $v_8 = (i-2, j-2)$ and $v_9 = (i, j-2)$. Selecting v_9 creates an upward crossing with arcs (v_9, v_5) and (u_7, u_2), thus we can discard it. On the other hand, selecting (at least) one of v_6 and v_7 (which is mandatory) makes v_4 to exhaust its possible predecessors (note that v_3 is also an in-neighbor v_4). We conclude that v_4 cannot be a predecessor of u_1.

Now, let us prove that choosing u_6 as predecessor of u_1 also leads to a contradiction (see Figure 4.21e). Since both u_1 and u_6 are in-neighbors of v_1, it needs three predecessors that can only be the cells u_5, $v_{10} = (i+2, j)$ and $v_{11} = (i+2, j+2)$. This creates a new downwards crossing with arcs (u_5, v_1) and (u_6, u_1). Observe that the arc (v_{11}, v_1) also leads to a contradiction since v_{11} cannot be a predecessor of v_1, for the same reason that cell $u' = (i+2, j+1)$ cannot be a predecessor of u_0 (Figure 4.21b). Therefore, cell u_6 cannot be a predecessor of u_1.

Given that none of v_4 and u_6 can be predecessor of u_1, we are forced to assume that u_4 and u_5 are the only two predecessors of u_1. We now prove that a pattern is repeated infinitely within the crossing gate, which is impossible (see Figure 4.21f). Since both u_4 and u_5 are in-neighbors of v_3, it needs three predecessors which can only be cells v_6, $v_{12} = (i-2, j+1)$ and $v_{13} = (i-2, j+2)$. Consequently, since v_{12} and v_{13} are in-neighbors of u_4, it needs three predecessors which can only be cells $u_9 = (i-2, j+3)$, $u_{10} = (i-1, j+3)$ and $u_{11} = (i, j+3)$. This is the first occurrence of the pattern, which is composed of the cells v_6, v_{12}, v_{13}, u_9, u_{10} and u_{11} (dashed on Figure 4.21f). Following with the reasoning, note that cells u_9 and u_{10} are both in-neighbors of v_{13}, which means that it needs three predecessors. These predecessors can only be cells $v_{14} = (i-3, j+1)$, $v_{15} = (i-3, j+2)$ and $v_{16} = (i-3, j+3)$. Both cells v_{15} and v_{16} are in-neighbors of u_9, *i.e.*, it needs three predecessors which can only be cells $u_{12} = (i-3, j+4)$, $u_{13} = (i-2, j+4)$ and $u_{14} = (i-1, j+4)$. This creates the second occurrence of the pattern which consists in the cells v_{14}, v_{15}, v_{16}, u_{12}, u_{13} and u_{14}. Inductively the pattern would repeat infinitely, which is absurd. We conclude that u_4 and u_5 cannot be the predecessors of u_1, which in turn implies that v_3 cannot be the predecessor of v_0.

Case v_4. This case is depicted in Figure 4.21g. Note that the cells v_0 and v_4 are in-neighbors of u_2, *i.e.*, it needs three predecessors, which can only be cells v_5, u_7 and u_8. On the other hand, cells u_1 and u_2 are in-neighbors of v_4, hence it needs three predecessors which can be selected among cells v_3, v_6, v_7 and v_{12}. There are two important facts to note at this point : first that cells v_0, v_4, v_6 and v_7 are all in-neighbors of v_5, and second that since v_4 needs three predecessors and there are four options, at least one of v_6 and v_7 has to be chosen. It follows that cell v_5 needs at least four predecessors, which is not possible.

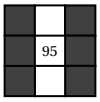
Case v_5. Cells v_0 and v_5 are in-neighbors of u_2, thus it needs three predecessors, which can only be cells v_4, u_7 and u_8. Since v_4 is an in-neighbor of v_0, cell v_0 needs an

extra predecessor, however we have already exhausted all the other possibilities in the previous cases.

Righward crossing. This case is symmetric to the downward crossing. ■

4.2.7 Crossover impossibility : escape cells and timestamps

Neighborhood studied in this subsection :



01011111

For the proofs that crossover gates do not exist in neighborhoods 95 and 39, we will strengthen the hypothesis on the crossing constraint given by Lemma 4.2 by considering the *first* crossing constraint occuring in one of the two firing graphs. This temporal minimality is formalized as follows.

Definition 4.5 (Escape cell)**.** Given a crossover gate g with firing graphs G^g_{ns} and G^g_{we}, let us call *crossing edge* of G^g_{ns} (resp. G^g_{we}) an arc $(u', u) \in A_{ns}$ (resp. $\in A_{we}$) that intersects an arc $(v', v) \in A_{we}$ (resp. A_{ns}). In this case u and v are called *escape cells*.

Definition 4.6 (Timestamp)**.** Given a crossover gate g with firing graphs G^g_{ns} and G^g_{we}, we associate to each cell of V_{ns} and V_{we} a timestamp $t \in \mathbb{N}$ defined as the one plus the maximum timestamp among its predessors, starting with timestamp 1 for cells n and w.

Timestamps exactly correspond to the temporality of topplings within a crossover gate. Note that a firing graph G^g_{ns} (resp. G^g_{we}) can have several escape cells with the same timestamp, and we will consider any of the minimum ones. Intuitively, this extra hypothesis of minimality will be helpful in reaching contradictions : in some cases a predecessor of the minimum escape cell under consideration will be surrounded by cells from the other firing graph, leaving no choice but to have another crossing edge on the path from n (resp. w) to it, hence another escape cell with a strictly smaller timestamp, contradicting the assumption on its minimality.

We formalize this intuition into the following lemma, but first define a notion of path partitioning the crossover gate (joining the two extremities of a firing graph).

Definition 4.7 (Dividing path)**.** *In a firing graph, a* dividing *path is a directed path from n to s (in G_{ns}), or a directed path from w to e (in G_{we}). Such a path divides the crossover gate into two parts.*

Without loss of generality, we can assume that each cell (appart from the starting and ending cells of the firing graph) from both firing graphs has at least one predecessor and at least one successor, otherwise it can be removed from the firing graph (by setting its sand content to 0) whilst the configuration is still a crossover gate. Observe that with this assumption, any directed path can be extending into a dividing path.

Lemma 4.3. *Given the pair of firing graphs of a crossover gate, if cell u has two ancestors u', u'', such that in the other firing graph there is a dividing path with u' and u'' lying in different parts, then u has an ancestor distinct from itself which is an escape cell.*

Proof. Cells u' and u'' must have an ancestor in common (the starting cell of their firing graph), which implies that the corresponding path (linking u' and u'') intersects the dividing path from the other firing graph. Since we consider only ancestors of u' and u'', cell u must have a strict ancestor (not itself) which is an escape cell. ■

We also derive the following corollary, because the intersection of the (directed) dividing path with the ancestors of u (more precisely with some path from its starting cell to u) must happen ahead of the predecessors of u.

Corollary 4.2. *In Lemma 4.3, if the dividing path intersects an arc whose head is cell u, then it has another intersection with the other firing graph, which happens strictly earlier on the dividing path.*


01011111 = 95

Theorem 4.21. *Neighborhood* 95 *does not admit a crossover gate.*

Proof. Let $u_1 = (i, j)$ be a minimum escape cell of one of the two firing graphs. It is important to mention that (u_0, u_1) is a crossing edge, *i.e.* it intersects an arc of the opposite firing graph. The two endpoints of this latter, denoted v_0 and v_1, are fixed by (u_0, u_1), but we will ignore its precise direction (which cell is the head or tail does not matter).

According to Lemma 4.2, there are a four possible orientations for the arc (u_0, u_1). However, they are all equivalent by symmetry. Therefore, without loss of generality, we consider only the case with $u_0 = (i+1, j-1)$, $v_0 = (i, j-1)$ and $v_1 = (i+1, j)$ (see Figure 4.22).

Since v_1 is an in-neighbor of u_1, then u_1 requires an extra predecessor. This predecessor can be one of the following cells : $w_2 = (i-1, j-1)$, $w_4 = (i-1, j)$, $w_6 = (i-1, j+1)$ or $w_8 = (i+1, j+1)$. We will prove that these four cases lead to contradictions.

Case w_4. If w_4 is a predecessor of u_1, then cells w_4 and u_0 are both predecessors of u_1. Furthermore, the edge between v_0 and v_1 can be extended into a dividing path, with w_4 and u_0 on different sides. Lemma 4.3 applies, contradicting the temporal minimality of escape cell u_1.

Case w_6. If w_6 is a predessor of u_1, then the situation is analogous to the previous case with w_6 playing the role of w_4 (on the opposite side of u_0).

Case w_2. If the extra predecessor of u_1 is w_2, then v_0 needs three predecessors, since both u_0 and w_2 are its in-neighbors. Considering that v_0 also needs a successor, then the following cells are in its firing graph : v_1, w_4, $w_0 = (i-1, j-2)$ and $w_1 = (i+1, j-2)$. For any choice of the successor among them, a directed path can be extended into a dividing path with w_2 and u_0 on different sides. It follows again by Lemma 4.3 that there is an escape cell with a strictly smaller timestamp than u_1, a contradiction.

Case w_8. Finally, if w_8 is the extra predecessor of u_1, then cell v_1 has an in-neighbor (u_1) from the other firing graph hence it needs at least two predecessors, and still at

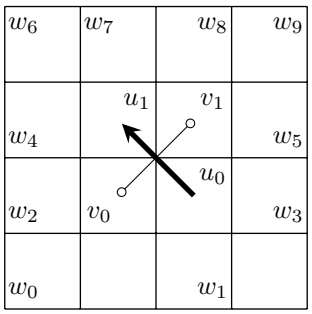
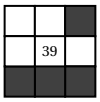


FIGURE 4.22 : Proof of Theorem 4.21 for neighborhood 95.

least one successor. For any choice of these three relatives of v_1 (cell v_0 and two more, chosen among cells $w_7 = (i, j+1)$, $w_3 = (i+2, j-1)$, $w_5 = (i+2, j)$ or $w_9 = (i+2, j+1)$), it creates a directed path that can be extended into a dividing path with u_0 and w_8 on different sides. By Lemma 4.3 we reach again a contradiction. ■

4.2.8 Crossover impossibility : almost crossing signals

Neighborhood studied in this subsection :



00100111

The study of neighborhood 39 requires much more developments, and the impossibility of crossover consists in a much longer proof split in multiple intermediate results. The intuitive reason for this is that neighborhood 39 almost has a crossover gate. This behavior is illustrated on Figure 4.23, which presents a situation where two disjoint firing graphs (none of the vertices from one firing graph is toppled by the other firing graph) almost cross each other. They are intricated, and this intrication can be continued up to an arbitrary length. We eventually prove that, despite of this construction where the two firing graphs interact in a non trivial way, a crossover gate is impossible in neighborhood 39 (Theorem 4.22).

For the proof, we first identify two forbidden patterns (Lemma 4.4; even though the forbidden pattern 2 will not be necessary, we still present it). Then the proof proceeds again by contradiction (assuming a crossover gate exists), with a large case analysis split into multiple lemmas. Each case is discard either *(1)* by encountering a forbidden pattern, or *(2)* because the crossover gate would be of infinite size, or *(3)* when it contradicts the temporal minimality of the escape cell under consideration.

Lemma 4.4 (Forbidden patterns)**.** *The firing graphs G^g_{ns} and G^g_{we} of a crossover gate g for neighborhood* 39 *cannot have any of the two induced pairs of subgraphs $G_{F_i} = (V_{F_i}, A_{F_i})$ and $G'_{F_i} = (V'_{F_i}, A'_{F_i})$ at any position $p, q \in \mathbb{Z}$ for $i \in \{1,2\}$, where (see Figure 4.24) :*

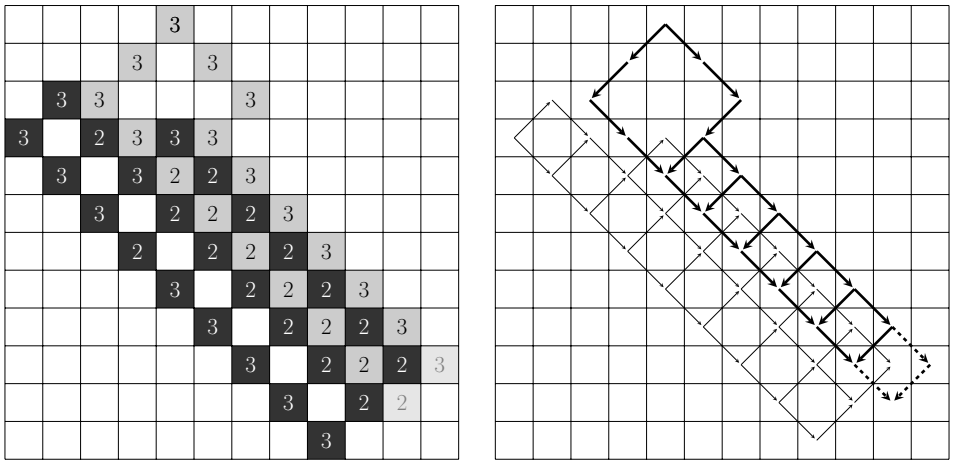
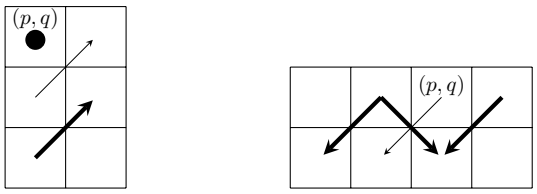


FIGURE 4.23 : Almost crossover gate for 39. The pattern can be repeated up to an arbitrary length, while preserving the fact that both firing graphs are disjoint. However, the two graphs do not eventually cross each other, as this is proven impossible in Theorem 4.22.



(a) Forbidden pattern 1.

(b) Forbidden pattern 2.

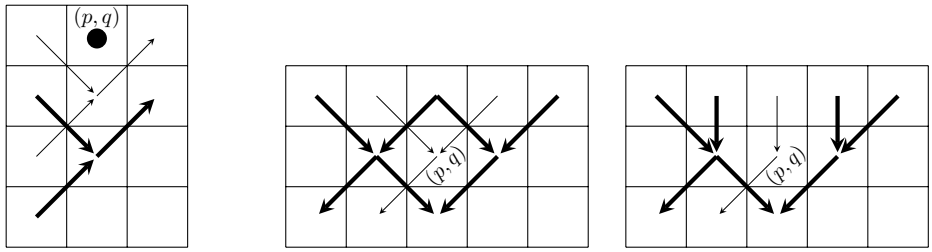
FIGURE 4.24 : Forbidden patterns for neighborhood 39 (Lemma 4.4).

Forbidden pattern 1.
$V_{F_1} = \{(p,q),(p,q-2),(p+1,q-1)\}$ *and* $A_{F_1} = \{((p,q-2),(p+1,q-1))\}$,
$V'_{F_1} = \{(p,q-1),(p+1,q)\}$ *and* $A'_{F_1} = \{((p,q-1),(p+1,q))\}$.
Forbidden pattern 2.
$V_{F_2} = \{(p,q),(p-1,q-1),(p-2,q),(p-3,q-1)\}$ *and* $A_{F_2} = \{((p,q),(p-1,q-1)),((p-2,q),(p-1,q-1)),((p-2,q),(p-3,q-1))\}$,
$V'_{F_2} = \{(p-1,q),(p-2,q-1)\}$ *and* $A'_{F_2} = \{((p-1,q),(p-2,q-1))\}$.

Proof. We show that these subgraphs repeat, hence inductively require that g is infinite.

Forbidden pattern 1. See Figure 4.25a. In such a case, cell $(p,q-2)$ needs two predecessors which can only be the cells $(p-1,q-1)$ and $(p-1,q-3)$. Something similar occurs to $(p,q-1)$, it needs two predecessors and the only options are cells $(p-1,q)$ and $(p-1,q-2)$. This repeats the original pattern shifted by $(-1,-1)$.

Forbidden pattern 2. Cell (p,q) needs to "come" from somewhere, *i.e.*, it needs a predecessor. If cell $(p-1,q+1)$ is its predecessor (Figure 4.25b), then cell $(p-1,q)$



(a) Forbidden pattern 1. Repetition.

(b) Forbidden pattern 2. Repetition.

(c) Forbidden pattern 2. Impossibility at (p, q).

FIGURE 4.25 : Proof of Lemma 4.4 for neighborhood 39.

needs two predecessors which can only be cells $(p-2, q+1)$ and $(p, q+1)$. Since $(p, q+1)$ is an in-neighbor of (p, q), it needs an extra predecessor that can only be the cell $(p+1, q+1)$, which causes $(p+1, q)$ to need two predecessors and the only remaining options are $(p, q+1)$ and $(p+2, q+1)$. The latter generates a repetition of the original pattern shifted by $(1, 1)$. The exact same pattern is obtained if $(p+1, q+1)$ is initially considered as the predecessor of (p, q).

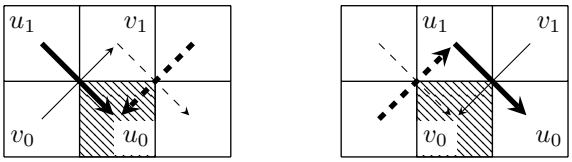
On the other hand, if $(p, q+1)$ is considered as the predecessor of (p, q) (Figure 4.25c), then both cells $(p-1, q)$ and $(p+1, q)$ need two predecessors each. The only options for cell $(p-1, q)$ are cells $(p-2, q+1)$ and $(p-1, q+1)$, while for $(p+1, q)$ the only options are $(p+1, q+1)$ and $(p+2, q+1)$. Since both cells $(p-1, q+1)$ and $(p+1, q+1)$ are in-neighbors of (p, q), cell (p, q) needs two extra predecessors, which is not possible. ■

We will consider a minimum escape cell, and number again the cells as given by Lemma 4.2, that is with $v_0 = (i, j)$, $v_1 = (i+1, j+1)$, $u_0 = (i+1, j)$ and $u_1 = (i, j+1)$. For neighborhood 39, up to symmetries there are four ways that a cell can be a minimum escape cell, which correspond to the following scenarios :

1. rightward crossing where u_0 is a minimum escape cell,
2. rightward crossing where v_1 is a minimum escape cell,
3. downward crossing where u_0 is a minimum escape cell,
4. downward crossing where v_0 is a minimum escape cell.

Scenarios 1 and 4 are the most difficult to handle, and in Lemma 4.5 we prove that they are actually equivalent, in the sense that the existence of one implies the existence of the other. Then a succession of tedious studies will lead to Lemma 4.8, stating that scenario 1 is impossible (and so is scenario 4). Lemma 4.9 proves that scenario 3 is impossible, which is fairly straightforward. At this point, given that only scenario 2 is possible in *both* firing graphs, a case analysis on the orientation of the crossover gate (regarding the cardinal directions north, east, south, west) will allow to conclude that a crossover gate is impossible for neighborhood 39. This last step of the proof (Theorem 4.22) exploits new arguments, using the fact that neighborhood 39 cannot transmit grains in the north-west direction.

Lemma 4.5. *The existence of a crossover gate g for neighborhood* 39 *with a rightward (resp. downward) crossing such that u_0 (resp. v_0) is a minimum escape cell, implies that g also contains a downward (resp. rightward) crossing with arcs $(v_1,(i+2,j))$ and $((i+2,j+1),u_0)$ (resp. $(u_1,(i-1,j))$ and $((i-1,j+1),v_0)$) (see Figure 4.26).*



(a) Rightward crossing ⇒ downward crossing.

(b) Downward crossing ⇒ rightward crossing.

FIGURE 4.26 : Statement of Lemma 4.5 for neighborhood 39. Minimum escape cell is dashed.

Proof. We prove the two respective implications of the statement one after the other.

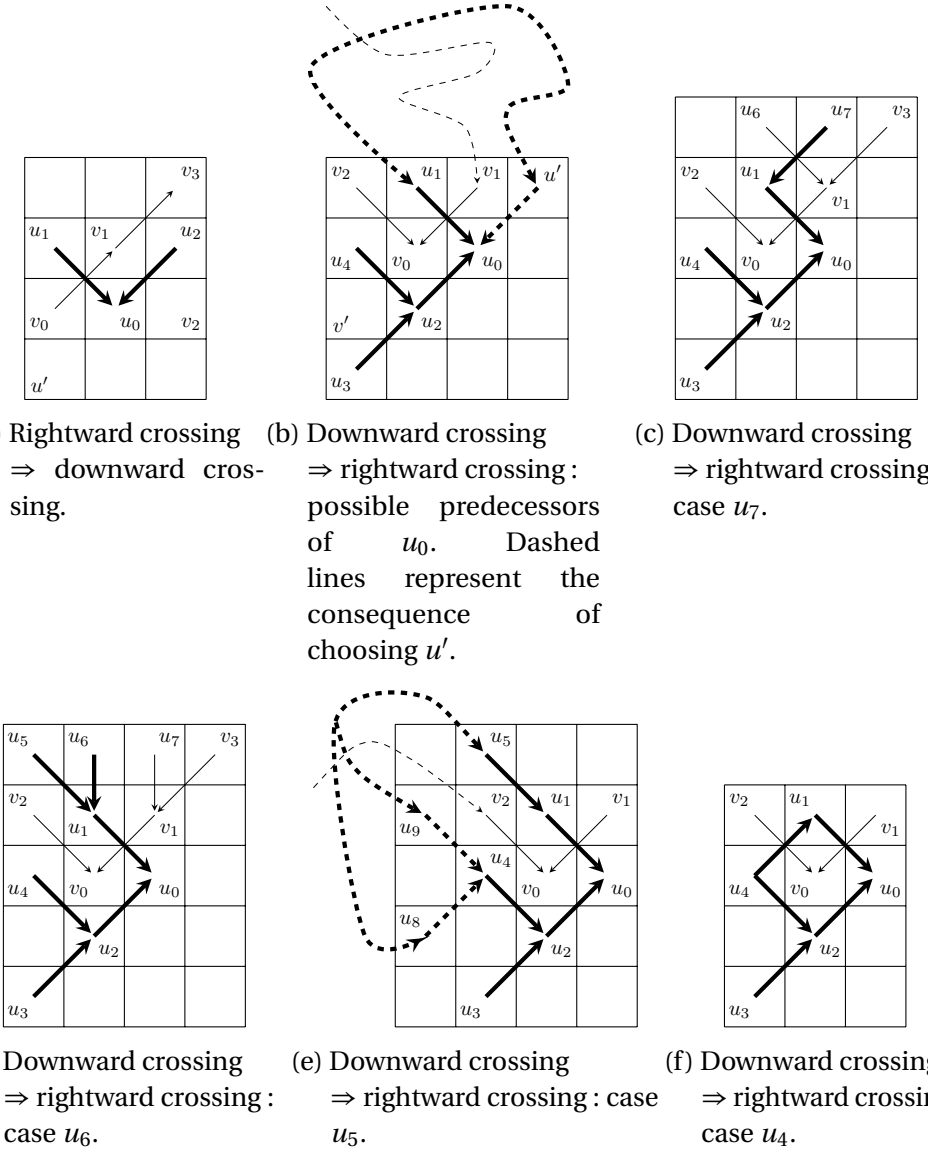
Righward crossing $\implies$ downward crossing. For the rightward crossing (Figure 4.27a), cell u_0 needs two predecessors, one of them is u_1 by our hypothesis, while the second one can be chose among $u' = (i, j-1)$ and $u_2 = (i+2, j+1)$. However, choosing u' creates a forbidden pattern 1 (Lemma 4.4), hence the second predecessor of u_0 must be cell u_2. Now we want to prove that v_1 can only have $v_2 = (i+2, j)$ as its successor. The other possibily is $v_3 = (i+2, j+2)$, but it would create a dividing path with u_1 and u_2 on different parts (recall that any directed path can be extended to a dividing path, and firing graphs are acyclic), contradicting by Lemma 4.3 the minimality of u_0.

Downward crossing $\implies$ rightward crossing. For a downward crossing where v_0 is a minimum escape cell, cell u_0 also needs an extra predecessor, which can be either be cell $u' = (i+2, j+1)$ or cell $u_2 = (i, j-1)$ (Figure 4.27b). However, note that choosing u' would let it be on the other side (compared to u_1) of any dividing path going through arc (v_1, v_0), hence contradicting the minimality of v_0 (by Corollary 4.2 on the dividing path). The latter means that u_2 must be the extra predecessor of u_0. Since v_0 is an in-neighbor of u_2, u_2 needs two predecessors that can only be the cells $u_3 = (i-1, j-2)$ and $u_4 = (i-1, j)$. The next step consists in choosing a predecessor for cell v_0. The possibilities are cells $v' = (i-1, j-1)$ and $v_2 = (i-1, j+1)$. Nevertheless, choosing v' creates a forbidden pattern 1 (Lemma 4.4), hence v_2 is the only option.

Now we prove that the arc (u_4, u_1) is mandatory. To do so, we analyse all cases over the possible predecessors of cell u_1. There are four possibilities : u_4, $u_5 = (i-1, j+2)$, $u_6 = (i, j+2)$ and $u_7 = (i+1, j+2)$.

Case u_7 (Figure 4.27c)**.** Since u_7 is an in-neighbor of v_1, cell v_1 needs two predecessors that can only be the cells $v_3 = (i+1, j+2)$ and u_6. However, this turns v_1 into an escape cell that topples strictly before v_0.

Case u_6 (Figure 4.27d)**.** Considering that u_6 is an in-neighbor of v_1, then v_1 needs two predecessors that can only be the cells v_3 and u_7. Since u_7 is also an in-neighbor



(a) Rightward crossing ⇒ downward crossing.

(b) Downward crossing ⇒ rightward crossing : possible predecessors of u_0. Dashed lines represent the consequence of choosing u'.

(c) Downward crossing ⇒ rightward crossing : case u_7.

(d) Downward crossing ⇒ rightward crossing : case u_6.

(e) Downward crossing ⇒ rightward crossing : case u_5.

(f) Downward crossing ⇒ rightward crossing : case u_4.

FIGURE 4.27 : Proof of Lemma 4.5 for neighborhood 39.

of u_1, then u_1 needs an extra predecessor that can be selected among u_4 and u_5. If it is u_4, then we are done. Hence, let us consider u_5. Since both u_5 and u_6 are in-neighbors of v_2, cell v_2 needs three predecessors, which is not possible.

Case u_5 (Figure 4.27e)**.** Cell v_2 is an in-neighbor of u_4, therefore u_4 needs two predecessors that have to be chosen among u_1, $u_8 = (i-2, j-1)$ and $u_9 = (i-2, j+1)$. Since it needs two predecessors, we are forced to pick at least one of u_8 and u_9. However, they both trap cell v_2 into the hypothesis of Corollary 4.2, which means there is another escape cell that topples before v_0.

Case u_4 (Figure 4.27f)**.** It is the only possible option. This generates a rightward crossing with arcs (u_4, u_1) and (v_2, v_0), as announced. ■

Next is a technical result (Lemma 4.6) for proving the existence of one arc in a precise situation. It will be an ingredient in the subsequent reasoning (Lemma 4.7), which aims at giving a large setup for the firing graphs in the case of scenario 1. Then Lemma 4.8 will conclude that scenario 1 is impossible.

Lemma 4.6. *If there exist a crossover gate g for neighborhood* 39 *with firing graphs G^g_{ns} and G^g_{we}, such that the following arcs belong to G^g_{ns} (resp. G^g_{we}) (see Figure 4.28a) :*
$((0,0),(1,-1))$, $((2,0),(1,-1))$,
and the following arcs belong to G^g_{we} (resp. G^g_{ns}) :
$((0,-1),(1,-2))$, $((0,-3),(1,-2))$, $((1,-2),(2,-1))$, $((1,0),(2,-1))$,
then the arc $((1,-1),(2,-2))$ belongs to G^g_{ns} (resp. G^g_{we}).
Any translation of the statement holds as well.

Proof. If the first part of the statement holds, then it is obvious that it holds up to translation by any vector $(p,q) \in \mathbb{Z}^2$, since we have no assumption peculiar to any absolute position within the $\mathbb{Z}^2$ grid.

The hypothesis are depicted on Figure 4.28a, and our goal is to prove that the dashed arc $(1,-1),(2,-2)$ is enforced. Note that the cell $(1,-1)$ needs a successor that can be either $(2,-2)$ or $(0,-2)$. If it is $(2,-2)$ we are done, so let us study the other case. If $(0,-2)$ is the successor of $(1,-1)$, then the cell $(0,-3)$ needs two predecessors that can only be the cells $(-1,-2)$ and $(-1,-4)$. Also note that $(0,-1)$ is an in-neighbor of $(0,-2)$, then cell $(0,-2)$ needs an extra predecessor that has to be chosen among cells $(-1,-1)$ and $(-1,-3)$. However, it cannot be the cell $(-1,-3)$ because it generates the forbidden pattern 1 (Lemma 4.4), then the only option is the cell $(-1,-1)$. Observe that the latter reasoning generates a repetition of the initial pattern, shifting by vector $(-1,-1)$ (Figure 4.28b).

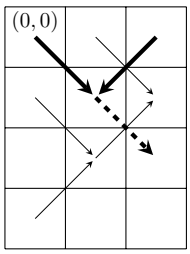
There is now a choice on the successor of cell $(0,-2)$, that can be either $(1,-3)$ or $(-1,-3)$. If $(1,-3)$ is chosen, then the reasoning generates another repetition of the initial pattern, shifted by vector $(-2,-2)$. This process can be repeated an arbitrary number of times (but a finite number, because a crossover gate is finite), until the successor of cell $(-k+1,-k-1)$ for some $k \in \mathbb{N}$ is chosen to be cell $(-k+2,-k-2)$ (see Figure 4.28c).

In that case, $(-k+2,-k-2)$ needs an extra predecessor since the cell $(-k+2,-k-1)$ is its in-neighbor. There are two options : cell $(1-k,-k-3)$, which generates the forbidden patern 1 (Lemma 4.4) hence we cannot chose it, and cell $(-k+3,-k-1)$. Now observe that cell $(-k+3,-k-1)$ has cell $(3-k,-k)$ as an in-neighbor, therefore it needs two predecessors which can only be cells $(-k+2,-k)$ and $(-k+4,-k)$. Moreover, the same reasoning applies inductively to all the cells :

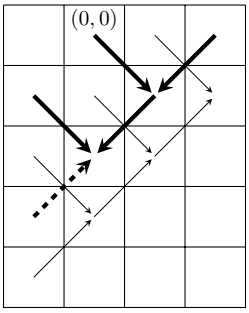
$$(-k+3,-k-1),(-k+4,-k),(-k+5,-k+1),\ldots,(1,-3),(2,-2).$$

This is depicted in Figure 4.28d. Eventually, cell $(2,-2)$ needs two predecessors which can only be cells $(1,-1)$ and $(3,-1)$, *i.e.*, the arc $((1,-1),(2,-2))$ is enforced. ■

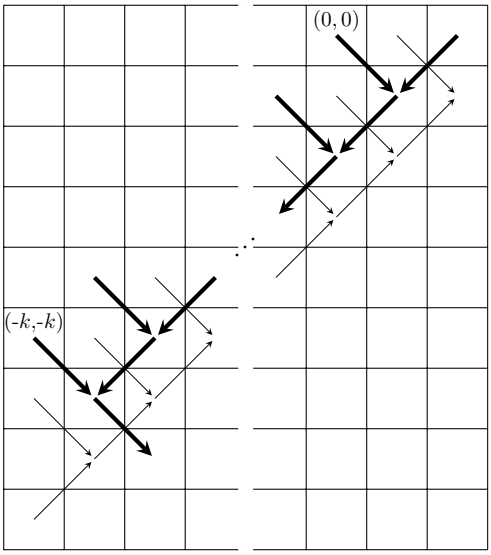
The next lemma settles a large part of the architecture of firing graphs in scenario 1.



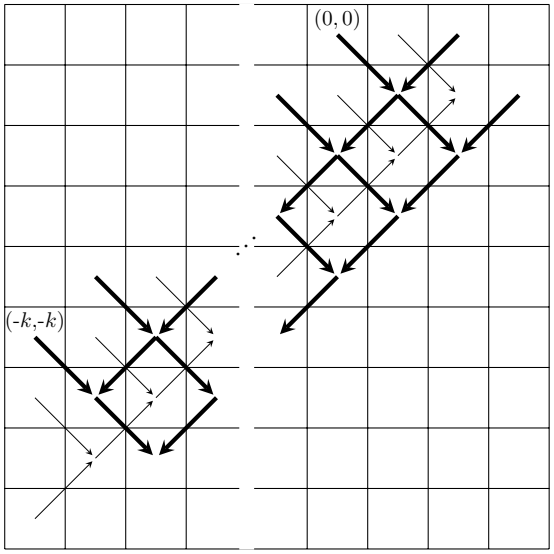
(a) Initial pattern with plain arcs, and the enforced arc dashed.



(b) Choosing $(0,-2)$ as a successor of $(1,-1)$. The dashed arc generates a forbidden pattern.



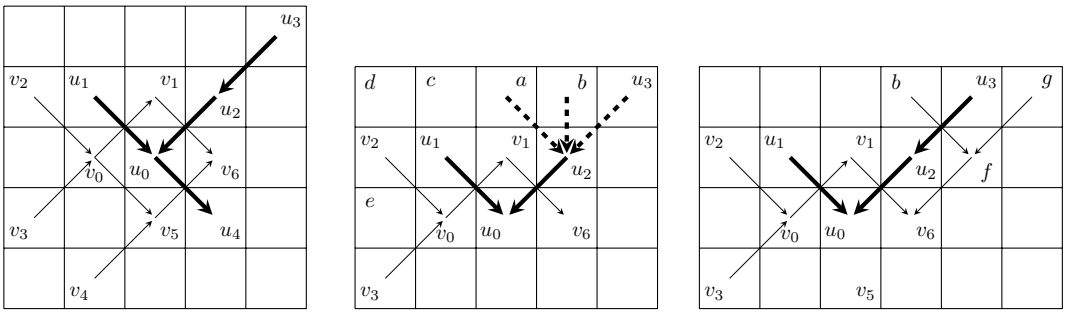
(c) The pattern repeated k times.



(d) Cell $(2,-2)$ requires $(1,-1)$ as a predecessor.

FIGURE 4.28 : Proof of Lemma 4.6 for neighborhood 39.

Lemma 4.7. *If there exists a crossover gate for neighborhood* 39 *with a rightwards crossing such that u_0 is a minimum escape cell (scenario 1), then the firing graphs for this crossover gate has the following nodes (see Figure 4.29a) : u_0, u_1, $u_2 = (i+2, j+1)$, $u_3 = (i+3, j+2)$, $u_4 = (i+2, j-1)$, v_0, v_1, $v_2 = (i-1, j+1)$, $v_3 = (i-1, j-1)$, $v_4 = (i, j-2)$,*



(a) Arcs obtained by Lemma 4.7.

(b) Predecessors of u_2.

(c) f cannot be a predecessor of v_6.

FIGURE 4.29 : Proof of Lemma 4.7 for neighborhood 39.

$v_5 = (i+1, j-1)$, $v_6 = (i+2, j)$, *and arcs :* (u_1, u_0), (u_0, u_4), (u_2, u_0), (u_3, u_2), (v_2, v_0), (v_3, v_0), (v_0, v_1), (v_0, v_5), (v_4, v_5), (v_1, v_6), (v_5, v_6).


00100111 = 39

Proof. The goal of this proof are the firings graphs depicted in Figure 4.29a. Starting from the hypothesis that u_0 is a minimum escape cell within a rightward crossing, we proced by proving that all the arcs stated in this Lemma are mandatory.

Arcs (u_2, u_0) **and** (v_1, v_6)**.** They follow by Lemma 4.5.

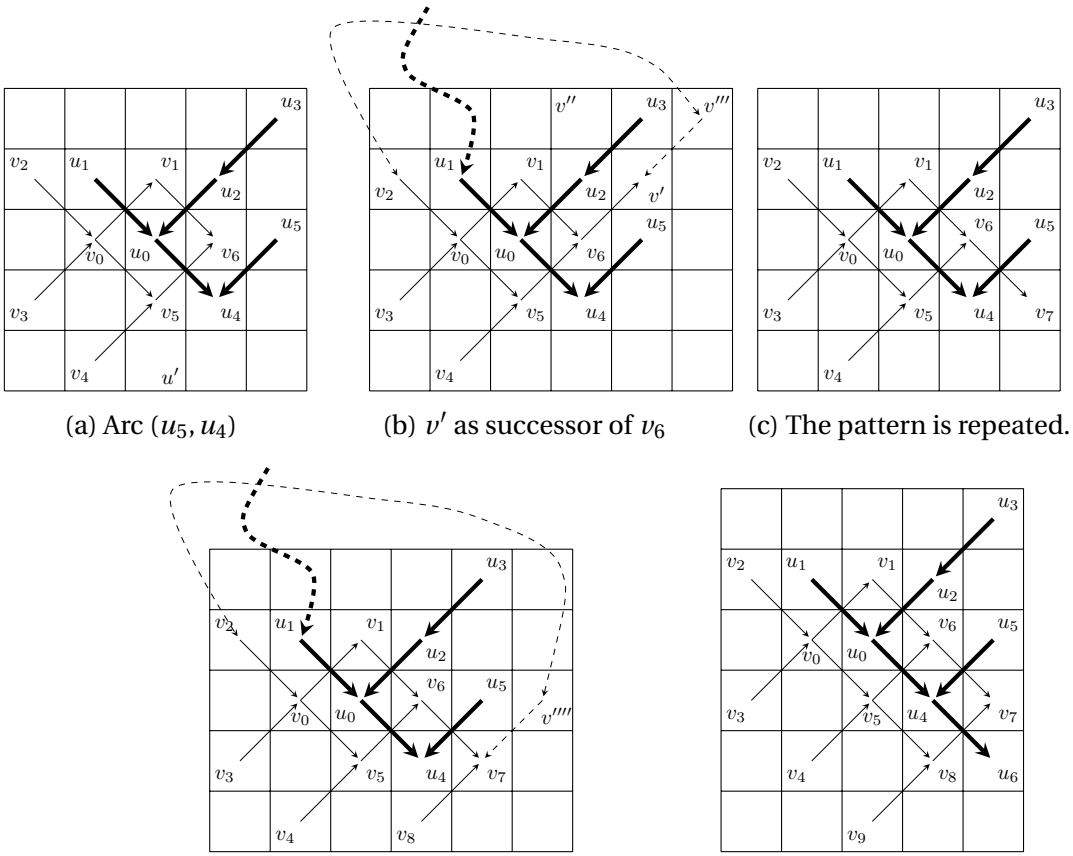
Arcs (v_2, v_0) **and** (v_3, v_0)**.** Cell v_0 has u_1 as an in-neighbor, hence it needs two predecessors which can only be cells v_2 and v_3.

Arc (u_3, u_2)**.** Let us check the possible predecessors of cell u_2 (see Figure 4.29b), which can be $a = (i+1, j+2)$, $b = (i+2, j+2)$ or u_3. If a is chosen, considering that it is an in-neighbor of v_1, then v_1 needs an extra predecessor which can be either $c = (i, j+2)$ or b. If b is chosen, then u_2 would turn into an escape cell that topples before u_0, contradicting its temporal minimality as an escape cell. On the other han, if c is chosen as the extra predecessor of v_1, then u_1 would need two predecessors that have to be chosen among the cells a, $d = (i-1, j+2)$ and $e = (i-1, j)$. However, chosing e generates the forbidden pattern 1 (Lemma 4.4), consequently a and d have to be the predecessors of u_1. Nevertheless, since a is an in-neighbor of v_1, cell v_1 needs an extra predecessor that can only be the cell b, but we already proved this to be contradictory.

In the case that b is chosen as the predecessor of u_2, then v_1 needs an extra predecessor that can be chosen between a and c. I we chose a, since it is an in-neighbor of u_2, then u_2 needs an extra predecessors, which can only be the cell u_3, and we are done. The remaining option is chosing c as predecessor of v_1. In such a case, u_1 would need two predecessors among cells a, d and e. Choosing cell e generates the forbidden pattern 1 (Lemma 4.4), then a and d have to be the predecessors of u_1. Nevertheless, since a is an in-neighbor of v_1, cell v_1 needs an extra predecessor, but there are no more available in-neighbors.

Finally, the only remaining possible predecessor of u_2 is u_3.

Arc (v_5, v_6)**.** Since u_2 is an in-neighbor v_6, then v_6 needs an extra predecessor, that can be either $f = (i+3, j+1)$ or v_5 (Figure 4.29c). If f is chosen then it would need two



(a) Arc (u_5, u_4) (b) v' as successor of v_6 (c) The pattern is repeated.

(d) Chosing v'''' as a predecessor of v_7 "traps" u_1. (e) Predecessors of v_8.

FIGURE 4.30 : Base case for Lemma 4.8 for neighborhood 39.

predecessors, this is because u_3 is an in-neighbor of f. The only possible predecessors for f are the cells b and $g = (i+4, j+2)$. However, this would turn u_2 into an escape cell that topples strictly before u_0. Therefore, the extra predecessor of v_6 has to be v_5.

Arcs (v_0, v_5) **and** (v_4, v_5)**.** Since u_0 is an in-neighbor of v_5, cell v_5 needs two predecessors which can only be the cells v_0 and v_4.

Arc (u_0, u_4)**.** It is obtained by Lemma 4.6. ■

Eventually, scenario 1 is proven to be impossible.

Lemma 4.8. *There does not exist a crossover gate g for neigborhood* 39 *with a rightward crossing such that* u_0 *is a minimum escape cell (scenario 1).*

Proof. We proceed by contradiction, assuming that there exists a crossover gate g such that u_0 is a minimum escape cell within a rightward crossing. We use the firing graphs obtained in Lemma 4.7 (Figure 4.29a) as a starting point. We prove by induction that this pattern requires a crossover gate of infinite size, which is absurd. The induction relies on shifting the pattern by vector $(1,-1)$ (except the arc from u_3 to u_2). We prove the first pattern repetition as a base case, and then prove that it would repeat

indefinitely, because the reasoning used after this first repetition does not require the presence of u_3 anymore.

Base case. Cell v_6 is an in-neighbor of u_4, then u_4 needs an extra predecessor. There are two options : $u' = (i+1, j-2)$ which generates the forbidden pattern 1 (Lemma 4.4), and $u_5 = (i+3, j)$ which must therefore be chosen (see Figure 4.30a).

Now let us study the successor of v_6. There are two possible out-neighbors to select. One of them is $v' = (i+3, j+1)$ and the other one is $v_7 = (i+3, j-1)$. We prove that chosing v' is not possible (see Figure 4.30b). If v' is the successor of v_6, since u_3 is its in-neighbor then it needs an extra predecessor than can be either $v'' = (i+2, j+2)$ or $v''' = (i+4, j+2)$. However, v'' cannot be chosen because it would turn u_2 into an escape cell that topples before u_0. Cell v''' cannot be selected either, since it traps the cell u_1, meaning there is another escape cell that topples before u_0 (Corollary 4.2 applies, considering the ancestors v_2 and v''' of v', and the dividing path going through u_1, u_0, u_4). The latter means that the successor of v_6 has to be v_7 (Figure 4.30c).

Note that u_5 is an in-neighbor of v_7, then v_7 needs an extra predecessor. This predecessor can be either $v'''' = (i+4, j)$ or $v_8 = (i+2, j-1)$. Analogously to the previous reasoning, choosing v'''' traps the cell u_1 (Figure 4.30d), therefore v_8 has to be selected.

The next step is to consider cell v_8. It needs two predecessors, since u_4 is its in-neighbor. The only options are v_5 and $v_9 = (i+1, j-3)$.

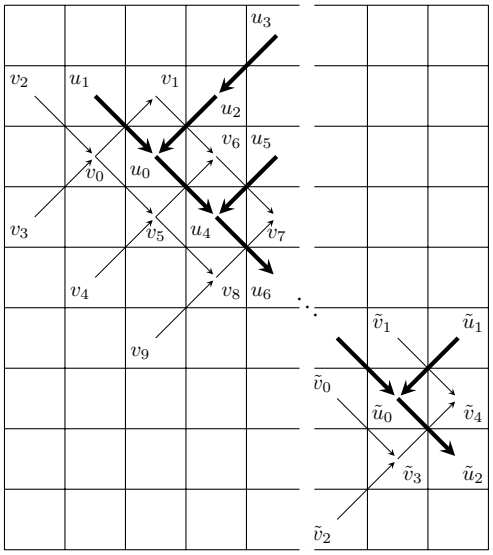
Finally, consider the cell $u_6 = (i+3, j-2)$, it can be added as the successor of u_4 by Lemma 4.6. The obtained situation is depicted on Figure 4.30e.

Induction step. Assume the pattern has already been repeated $k \in \mathbb{N}$ times, we will prove that it must repeated once more. We got the cells (see Figure 4.31a) : $\tilde{u}_0 = (i+k+1, j-k)$, $\tilde{u}_1 = (i+k+2, j-k+1)$, $\tilde{u}_2 = (i+k+2, j-k-1)$, $\tilde{v}_0 = (i+k, j-k)$, $\tilde{v}_1 = (i+k+1, j-k+1)$, $\tilde{v}_2 = (i+k, j-k-2)$, $\tilde{v}_3 = (i+k+1, j-k-1)$, $\tilde{v}_4 = (i+k+2, j-k)$, and the arcs : $(\tilde{u}_0, \tilde{u}_2)$, $(\tilde{u}_1, \tilde{u}_0)$, $(\tilde{v}_0, \tilde{v}_3)$, $(\tilde{v}_2, \tilde{v}_3)$, $(\tilde{v}_1, \tilde{v}_4)$, $(\tilde{v}_3, \tilde{v}_4)$.

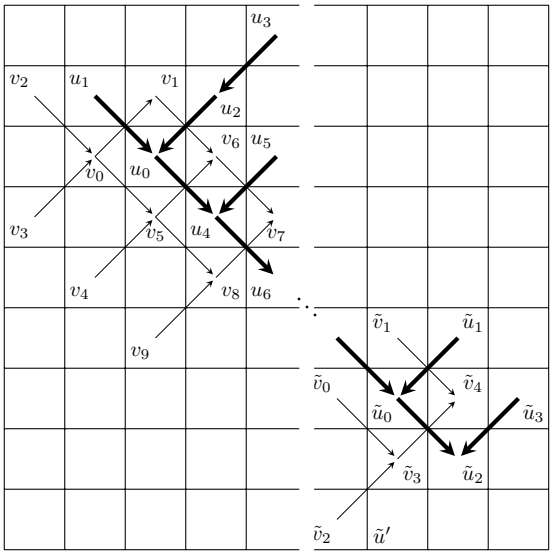
Note that cell $\tilde{v}_4$ is an in-neighbor of $\tilde{u}_2$, then $\tilde{u}_2$ needs an extra predecessor. There are two options, however, one of them, the cell $\tilde{u}' = (i+k+1, j-k-2)$, generates the forbidden pattern 1 (Lemma 4.4), consequently the only valid option is cell $\tilde{u}_3 = (i+k+3, j-k)$ (Figure 4.31b).

Now let us analyse the possible successors of $\tilde{v}_4$. There are two possibilities : cells $\tilde{v}_5 = (i+k+3, j-k-1)$ and $\tilde{v}_6 = (i+k+3, j-k+1)$. We prove that choosing $\tilde{v}_6$ implies that the arc $(\tilde{v}_4, \tilde{v}_5)$ also exists (Figure 4.31c). If $\tilde{v}_6$ is the successor of $\tilde{v}_4$ we can assume that $\tilde{v}_4$ is the only cell in the path $p = (v_1, v_6, v_7, \ldots, \tilde{v}_1, \tilde{v}_4)$ that has its north-east out-neighbor as a succesor. Also, if $\tilde{v}_6$ is the successor of $\tilde{v}_4$, then $\tilde{u}_3$ needs two predecessors (since $\tilde{v}_6$ is its in-neighbor). There are only two options : cells $\tilde{u}_1$ and $\tilde{u}_4 = (i+k+4, j-k+1)$. Now, the same reasoning that was applied over $\tilde{v}_4$ can be applied over $\tilde{v}_6$. Moreover, it can be applied inductively. Let assume it has been applied $\tilde{k} \in \mathbb{N}$ times. The latter means that we have the following cells (see Figure 4.31c) : $\hat{v}_0 = (k+\tilde{k}+2, -k+\tilde{k})$, $\hat{v}_1 = (k+\tilde{k}+1, -k+\tilde{k}-1)$, $\hat{u}_0 = (k+\tilde{k}+2, -k+\tilde{k}-1)$, $\hat{u}_1 = (k+\tilde{k}+1, -k+\tilde{k})$, $\hat{u}_2 = (k+\tilde{k}+3, -k+\tilde{k})$, and arcs : $(\hat{v}_1, \hat{v}_0)$, $(\hat{u}_1, \hat{u}_0)$, $(\hat{u}_2, \hat{u}_0)$.

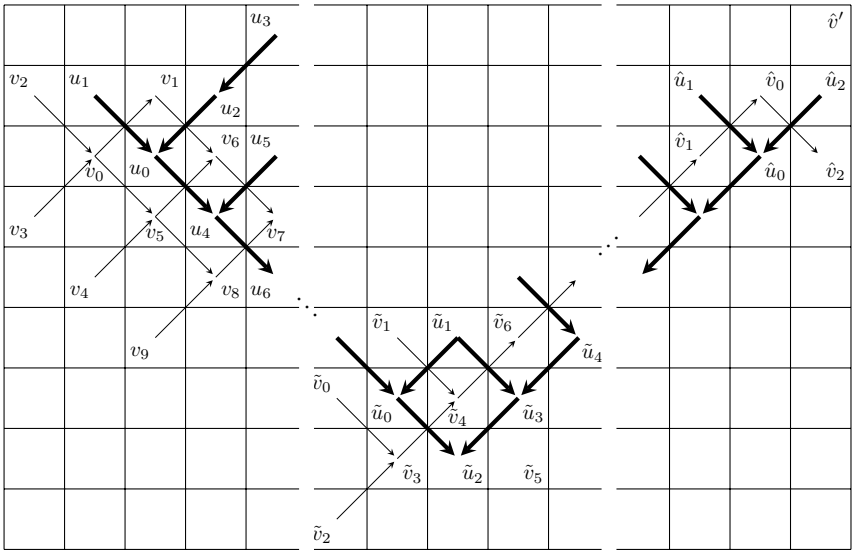
Now $\hat{v}_0$ needs a successor that can be either $\hat{v}' = (k+\tilde{k}+3, -k+\tilde{k}+1)$, repeating the pattern once again, or $\hat{v}_2 = (k+\tilde{k}+3, -k+\tilde{k}-1)$. As choosing $\hat{v}'$ just repeats the pattern,



(a) Induction hypothesis



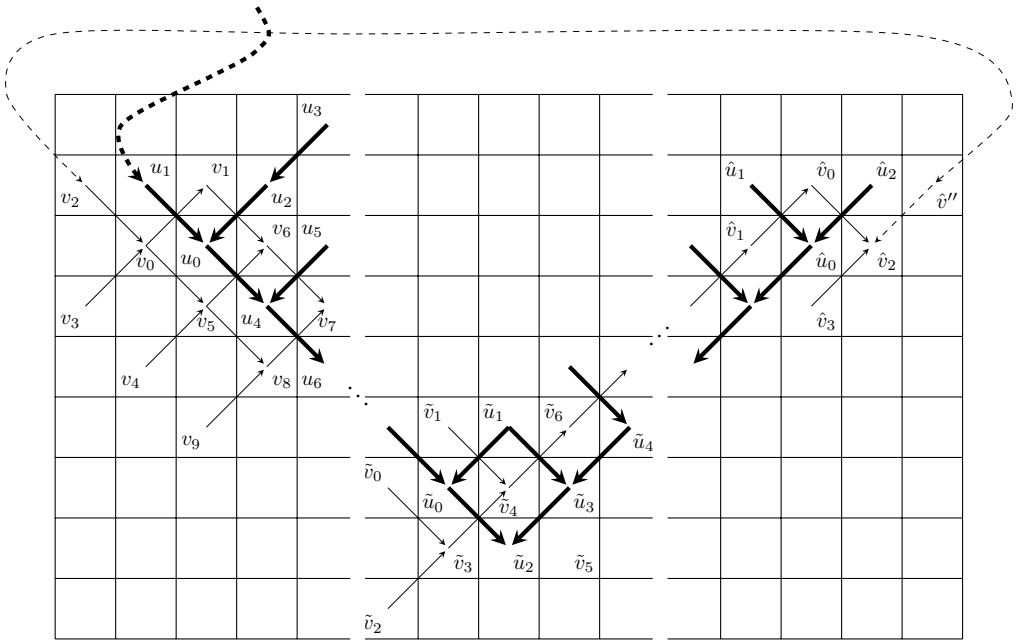
(b) $\tilde{u}_3$ has to be the extra predecessor of $\tilde{u}_2$.



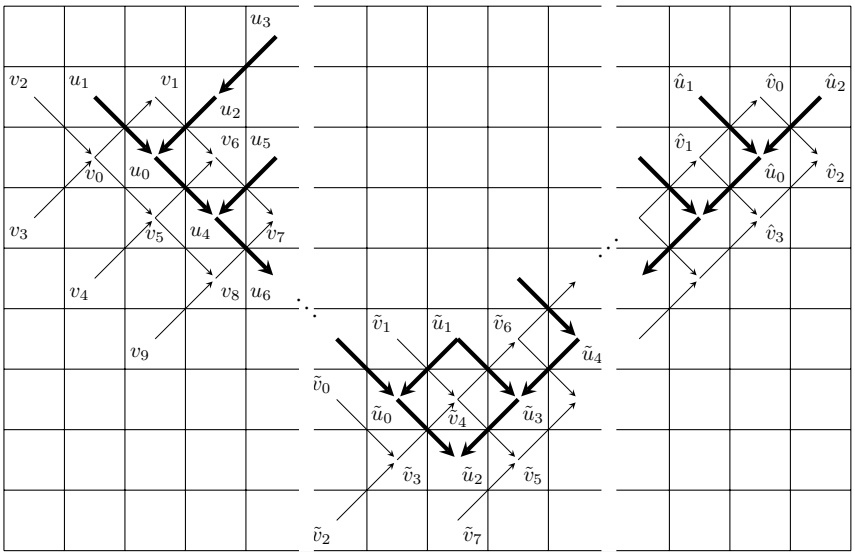
(c) $\tilde{v}_6$ as successor of $\tilde{v}_4$.

FIGURE 4.31 : Induction step for Lemma 4.8 for neighborhood 39.

let us focus on $\hat{v}_2$. Since $\hat{u}_2$ is an in-neighbor of $\hat{v}_2$, it needs an extra predecessor. There are two options, the cells $\hat{v}'' = (k+\tilde{k}+4, -k+\tilde{k})$ and $\hat{v}_3 = (k+\tilde{k}+2, -k+\tilde{k}-2)$ (Figure 4.31d). However, chosing cell $\hat{v}''$ traps cell u_1 (as depicted in Figure 4.31d), which would contradict the minimality of escape cell u_0. One could think, in order to avoid contradicting the minimality of u_0, that one of the cell of the path p can be an ancestor of $\hat{v}''$, however, as we mentioned before, we assume that v_4 is the only cell in p that has its north-east out-neighbor as a successor, and v_4 is clearly not an ancestor of $\hat{v}''$. In other words, v_2 and $\hat{v}''$ have an ancestor in common that is above



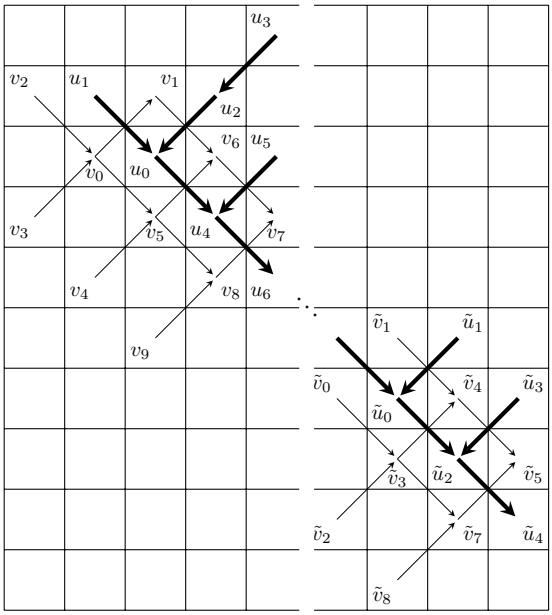
(d) Possible predecessors of $\hat{v}_2$.



(e) The arc $(\tilde{v}_4, \tilde{v}_5)$ is mandatory.

FIGURE 4.31 : Induction step for Lemma 4.8 for neighborhood 39.

both of them, then, we can extend the arc (u_1, u_0) into a dividing path with v_2 and $\hat{v}''$ on different sides, contradicting the minimality of escape cell u_0. Therefore, chosing $\hat{v}_3$ is the only remaining option. Since $\hat{u}_0$ is an in-neighbor of $\hat{v}_3$, cell $\hat{v}_3$ needs two predecessors, that can only be the cells $\hat{v}_1$ and $\hat{v}_4 = (k + \tilde{k} + 1, -k + \tilde{k} - 3)$. Moreover, all the following cells need two predecessors and they only have the north-east and



(f) Repetition of the pattern.

FIGURE 4.31 : Induction step for Lemma 4.8 for neighborhood 39.

south-east in-neighbors available (Figure 4.31e) :

$$\hat{v}_3 = (k+\tilde{k}+2, -k+\tilde{k}-2), (k+\tilde{k}+1, -k+\tilde{k}-3), (k+\tilde{k}, -k+\tilde{k}-4),$$
$$(k+\tilde{k}-1, -k+\tilde{k}-5), \ldots, (i+k+4, j-k), \tilde{v}_5 = (i+k+3, j-k-1).$$

In particular, the two predecessors of $\tilde{v}_5$ must be the cells $\tilde{v}_4$ and $\tilde{v}_7 = (i+k+2, j-k-2)$, *i.e.*, the arc $(\tilde{v}_4, \tilde{v}_5)$ is mandatory.

Now, note that $\tilde{u}_2$ is an in-neighbor of $\tilde{v}_7$, then $\tilde{v}_7$ needs two predecessors that can only be the cells $\tilde{v}_3$ and $\tilde{v}_8 = (i+k+1, j-k-3)$ (Figure 4.31f). Finally, by Lemma 4.6, cell $\tilde{u}_4 = (i+k+3, j-k-2)$ is a successor of cell $\tilde{u}_2$, which completes the $(k+1)$-th repetition of the pattern for the induction. ■

We deduce that scenario 4 is also impossible.

Corollary 4.3. *There does not exist a crossover gate g for neigborhood* 39 *with a downward crossing such that v_0 is a minimum escape cell (scenario 4).*

Proof. Direct consequence of Lemmas 4.5 (equivalence between scenarios 1 and 4) and 4.8 (impossibility of scenario 1). ■

Let us now prove that scenario 3 is impossible, which is much simpler to establish.

Lemma 4.9. *There does not exist a crossover gate g for neigborhood* 39 *with a downward crossing such that u_0 is a minimum escape cell (scenario 3).*

Proof. Note that u_1 is an in-neighbor of v_0, then v_0 needs an extra predecessor. The only two options are cells $v_2 = (i-1, j+1)$ and $v_3 = (i-1, j-1)$. However, both of them trap the arc (u_1, u_0) into a dividing path where v_1 and v_2 (or v_3) are on opposite sides (see Figure 4.32). By Corollary 4.2, there is an escape cell that topples strictly before u_0 and contradicts its minimality. ■

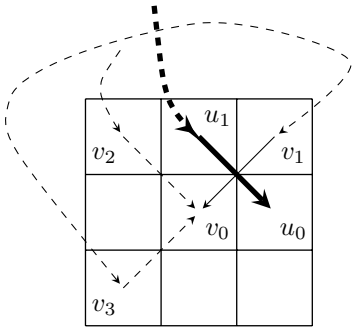


FIGURE 4.32 : Proof of Lemma 4.9.

We now turn to the final reasoning for the impossibility of a crossover gate in neighborhood 39. According to the previous results, minimum escape cells from *both* firing graphs must be in scenario 2 (rightward crossing where v_1 is a minimum escape cell. The following lemma expresses the fact that the descendant relationship in neighborhood 39 cannot go in the north-west direction.

Lemma 4.10. *For neighborhood* 39*, if cell* (i', j') *is a descendant of cell* (i, j)*, then :*

$$j - i \geq j' - i'.$$

Proof. Considering that the neighborhood 39 is defined as :

$$\mathcal{N}_{39} = \{(1,1), (1,-1), (0,-1), (-1,-1)\}$$

the out-neighbors (potential descendants) of a cell (i, j) are $(i+1, j+1)$, $(i+1, j-1)$, $(i, j-1)$ and $(i-1, j-1)$. All of them fulfil the condition, and the lemma follows by transitivity of the order $\geq$ on integers (recall that the descendent relationship is the transitive closure of the successor relationship). ■


00100111 = 39

Theorem 4.22. *Neighborhood* 39 *does not admit a crossover gate.*

Proof. Considering Corollary 4.3, Lemma 4.8 and Lemma 4.9, if there exists a crossover gate g with firing graphs $G_a^g = (V_a, A_a)$ and $G_b^g = (V_b, A_b)$ for the neighborhood 39, then any minimum escape cell (i_a, j_a) of G_a^g (resp. (i_b, j_b) of G_b^g) must appear in a rightward crossing and be toppled by the predecessor $(i_a - 1, j_a - 1)$ (resp. $(i_b - 1, j_b - 1)$). This corresponds to scenario 2, which is the situation in *both* firing graphs, since scenarios 1, 3 and 4 in *any* of the two firing graphs has been proven to be impossible.

Notice that the rightward crossing having minimum escape cell (i_a, j_a) consists in the arcs :

$$((i_a-1, j_a-1),(i_a, j_a)) \in A_a \text{ and } ((i_a-1, j_a),(i_a, j_a-1)) \in A_b,$$

while the rightward crossing having minimum escape cell (i_b, j_b) consists in the arcs :

$$((i_b-1, j_b-1),(i_b, j_b)) \in A_b \text{ and } ((i_b-1, j_b),(i_b, j_b-1)) \in A_a.$$

The orientation of the crossing with a minimum escape cell has been fixed (rightward in scenario 2), therefore it is necessary to consider all global orientations of the crossover gate (in terms of its cardinal endpoints). We study the following four scenarios, which exhaust all possibilities (up to exchange of the two firing graphs).

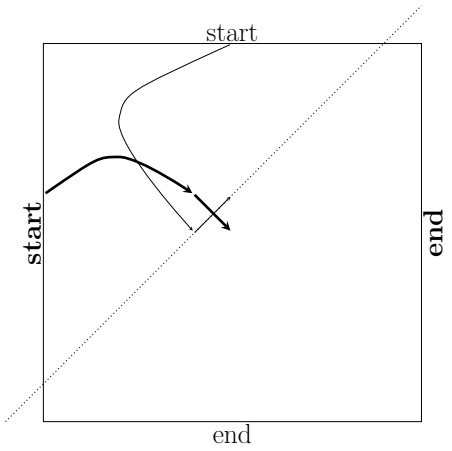
2.1. G_a^g goes from north to south and G_b^g goes from west to east,

2.2. G_a^g goes from north to south and G_b^g goes from east to west,

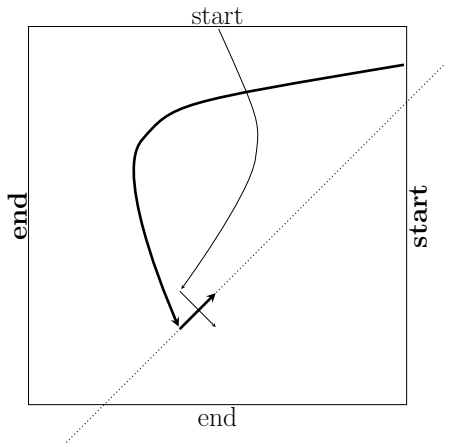
2.3. G_a^g goes from south to north and G_b^g goes from west to east,

2.4. G_a^g goes from south to north and G_b^g goes from east to west.

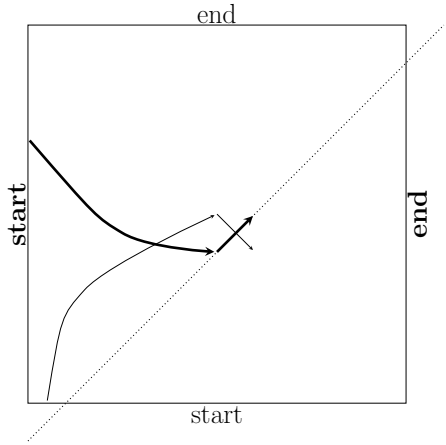
Scenario 2.1 (Figure 4.33a)**.** Consider the rightward crossing with minimum escape cell (i_a, j_a) in G_a^g. Firing graph G_a^g must have a path p_a from the north border to the cell (i_a-1, j_a), and firing graph G_b^g must have a path p_b from the west border to the cell (i_a-1, j_a-1). In this case, either the path p_b intersects the path p_a (contradicting the minimality of escape cell (i_a, j_a) in G_a^g), or the path p_b contains a cell (i, j) such that $j-i < j_a-i_a$ (under the dotted diagonal), which cannot reach cell (i_a-1, j_a-1) by Lemma 4.10 (another contradiction). Consequently this scenario is impossible.
Scenario 2.2 (Figure 4.33b)**.** Consider the rightward crossing with minimum escape cell (i_b, j_b) in G_b^g. Firing graph G_a^g must have a path p_a from the north border to the cell (i_b-1, j_b), and firing graph G_b^g must have a path p_b from the east border to the cell (i_b-1, j_b-1). In this case, either the path p_b intersects the path p_a (contradicting the minimality of escape cell (i_b, j_b) in G_b^g), or the path p_b contains a cell (i, j) such that $j-i < j_b-i_b$ (under the dotted diagonal), which cannot reach cell (i_b-1, j_b-1) by Lemma 4.10 (another contradiction). Consequently this scenario is impossible.
Scenario 2.3 (Figure 4.33c)**.** Consider the rightward crossing with minimum escape cell (i_b, j_b) in G_b^g. Firing graph G_a^g must have a path p_a from the south border to the cell (i_b-1, j_b), and firing graph G_b^g must have a path p_b from the west border to the cell (i_b-1, j_b-1). In this case, either the path p_b intersects the path p_a (contradicting the minimality of escape cell (i_b, j_b) in G_b^g), or the path p_b contains a cell (i, j) such that $j-i < j_b-i_b$ (under the dotted diagonal), which cannot reach cell (i_b-1, j_b-1) by Lemma 4.10 (another contradiction). Consequently this scenario is impossible.
Scenario 2.4 (Figure 4.33d)**.** Consider the rightward crossing with minimum escape cell (i_a, j_a) in G_a^g. Firing graph G_a^g must have a path p_a from the south border to the cell (i_a-1, j_a-1), and firing graph G_b^g must have a path p_b from the east border to the



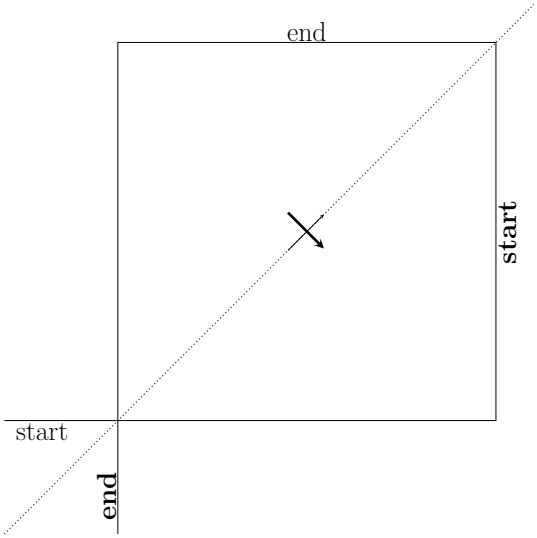
(a) Scenario 2.1.



(b) Scenario 2.2.



(c) Scenario 2.3.



(d) Scenario 2.4.

FIGURE 4.33 : Proof of Theorem 4.22 for neighborhood 39. Thin lines represent G_a^g, and thick lines represent G_b^g. Cells on or under the dotted diagonals cannot have descendants over it by Lemma 4.10.

cell (i_a-1, j_a). We procced by case analysis on the position of cell (i_a-1, j_a-1). Let m be the size of g, and assume without loss of generality that its support is the square of cells (i, j) verifying $0 \leq i \leq m-1$ and $0 \leq j \leq m-1$.

If cell (i_a-1, j_a-1) is such that $j_a-1 > i_a-1$ (it is above the diagonal going from $(0,0)$ to $(m-1, m-1)$), then the starting cell $s = (i_s, 0)$ of G_a^g has to fulfil that $i_s < 0$ in order to have (i_a-1, j_a-1) as a descendant (by Lemma 4.10), which is a contradiction because it means that it is outside of the crossover gate g.

On the other hand, if (i_a-1, j_a-1) is such that $j_a-1 < i_a-1$, then the ending cell $n = (i_n, m-1)$ of G_a^g has to fulfil that $i_n > m-1$ in order to have (i_a, j_a) as an ancestor (by Lemma 4.10), which is a contradiction because it means that it is outside of g.

Finally, if (i_a-1, j_a-1) is such that $j_a-1 = i_a-1$, then the starting cell $e = (m-1, i_e)$ of G_g^b has to fulfil that $i_e > m-1$ in order to have (i_a-1, j_a) as a descendant (by Lemma 4.10), which is a contradiction because it means that it is outside of g. ■

4.3 Timed crossover among subsets of Moore

In this section we go further in the study of the computational complexity of sandpile dynamics, by considering timed prediction problems ($\mathcal{N}$-**TIMED-PRED**) and timed crossover gates. An example of timed crossover gate is presented on Figure 4.34. We will see that some neighborhoods which do not admit a crossover gate, however admit a timed crossover gate and have a P-complete timed prediction problem. This hints at the fact that predicting the dyanmics at a precise time step may be harder than asymptotic prediction of the dynamics (the final stable configuration).

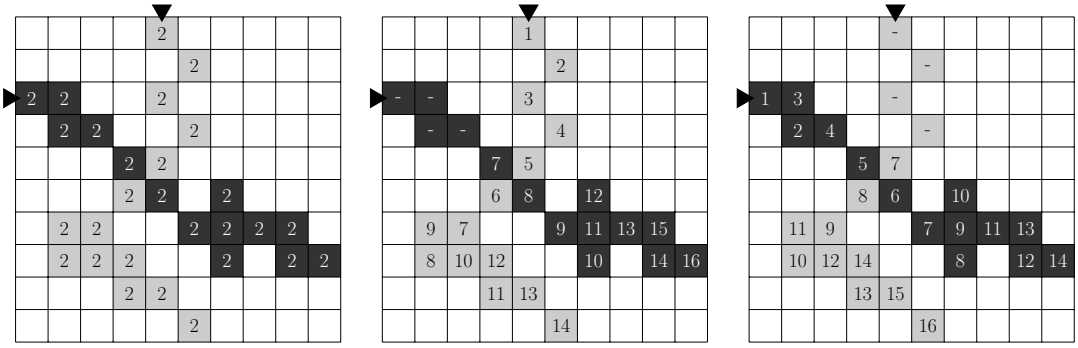


FIGURE 4.34 : Example of timed crossover gate of size 10×10 for neighborhood 131 (left). The delay of the gate is 14. We picture the timestamps of each cell, following a grain addition at n (center), and a grain addition at w (right).

Section outline. We start this section by defining timed firing graphs (Subsection 4.3.1), and prove that timed crossover gate are impossible when the neighborhood is planar (Subsection 4.3.2). Then we exhibit timed crossover gates for many neighborhoods. For some of them we prove that $\mathcal{N}$-**TIMED-PRED** is P-complete (Subsection 4.3.3), and for others we are faced with a delay issue when trying to connect the different gates (Subsection 4.3.4). The results of this section are summarized in Table 4.3.

4.3.1 Timed firing graphs

The pair of firing graphs of a timed crossover gate may intersect (have vertices in common), because the wire in the perpendicular direction can be toppled (provided that it does so some steps behind compared to the 1 signal traversing the gate). We introduce the notion of *timed firing graphs*, which are subsets of the firing graphs.

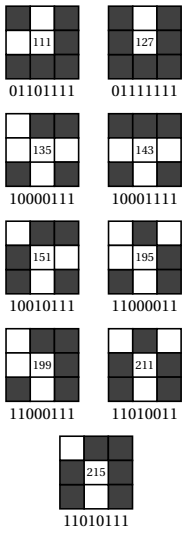
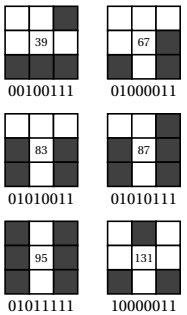
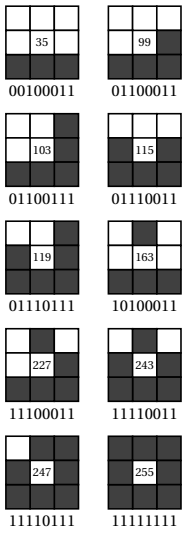
Section	4.3.3	4.3.4	4.3.5
Type	P-complete	Delay issue	Timed crossover impossibility (conjecture)
Neigborhoods			

TABLE 4.3 : Summary of results exposed in Section 4.3.

Given that the arcs of the firing graphs represent the causal relationship among toppled cells, it would be enough to discard the part not necessary to topple one exit cell, by restricting the corresponding firing graph to its ancestors. However, this inductive definition from the exit cell backwards, is not convenient for proofs by induction on the number of steps. We prefer the following definition, based on timestamps, which are now denoted $t_{ns} : V_{ns} \to \mathbb{N}$ and $t_{we} : V_{we} \to \mathbb{N}$ in order to differenciate the two firing graphs. In each firing graph, it takes into account only cells toppling sufficiently early to belong to 1 signals. Recall that the timestamps of the two exit cells are equal to the delay $T \in \mathbb{N}_+$ of the gate. We consider $t_{ns}(v) = +\infty$ when t_{ns} is not defined for v. The same applies to t_{we}.

Definition 4.8 (Timed firing graphs)**.** *Given a timed crossover gate g of delay $T \in \mathbb{N}_+$ on cells $n, s, w, e, [\![m]\!]^2$, its two* timed firing graphs *are the subsets of its firing graphs, restricted to vertex sets :*

$$V_{ns}^T = \{v \in V_{ns} \mid t_{ns}(v) \leq t_{we}(v)\} \text{ and } V_{we}^T = \{v \in V_{we} \mid t_{we}(v) \leq t_{ne}(v)\}.$$

It is necessary that in a timed crossover gate, each of the two exit cells belongs only to its corresponding timed firing graph. The $\leq$ comparison among timestamps aims at discarding cells that have some retard. We generalize Lemma 4.1 from NGUYEN et al. 2018, to the context of timed firing graphs.

Lemma 4.11. *If a sandpile CA $\mathcal{N}$ admits a timed crossover gate, then it also admits a timed crossover gate g with timed firing graphs $G_{ns}^T = (V_{ns}^T, A_{ns}^T)$ and $G_{we}^T = (V_{we}^T, A_{we}^T)$, such that $V_{ns}^T \cap V_{we}^T = \emptyset$.*

Proof. The proof is constructive : given a timed crossover gate g' with timed firing graphs $G'^T_{ns} = (V'^T_{ns}, A'^T_{ns})$ and $G'^T_{we} = (V'^T_{we}, A'^T_{we})$, we :

1. remove all grains from vertices in the intersection $V'^T_{ns} \cap V'^T_{we}$ (which are intuitively useless), so that they do not topple anymore,

2. compensate for the missing predecessors in each timed firing graph : for each vertex in V'^T_{ns} exclusive or V'^T_{we}, add as many grains to it as it had predecessors in $V'^T_{ns} \cap V'^T_{we}$.

We now argue that it gives another timed crossover gate with distinct timed firing graphs. By induction on successive time steps, we have the following three simple facts. First, the vertices in $V'^T_{ns} \cap V'^T_{we}$ do not topple anymore (this uses the fact that sandpile graphs are Eulerian). Second, the vertices not in the intersection are still in their timed firing graph (thanks to the sand grains added at step 2, there is no retard in the topplings). Third, no new vertex appears in each timed firing graph, *i.e.* $V^T_{ns} \subseteq V'^T_{ns}$ and $V^T_{we} \subseteq V'^T_{we}$ (because the sand grains added at step 2 do not create successor relationships in the timed firing graph). Given that $n, s \in V'^T_{ns} \setminus V'^T_{we}$ and $w, e \in V'^T_{we} \cap V'^T_{ns}$, this still holds and the constructed configuration is indeed a timed crossover gate (it still performs a timed crossover of two signals), with sets of vertices $V^T_{ns} = V'^T_{ns} \setminus V'^T_{we}$ and $V^T_{we} = V'^T_{we} \setminus V'^T_{ns}$. ■

4.3.2 Timed crossover impossibility : planar neighborhoods

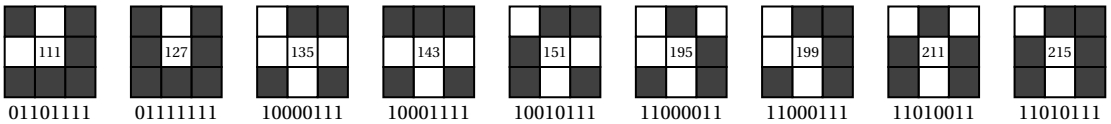
As a corollary of Lemma 4.11, it is impossible to have a timed crossover gate when the neighborhood is planar.

Corollary 4.4. *If neighborhood $\mathcal{N}$ has a planar sandpile graph, then it does not admit a timed crossover gate.*

This proves that neighborhoods 66, 74, 82, 98, 90, 106, 130, 146, 192, 200, 202, 208, 210, 226, 234, 240, 242 and 250 do not admit a timed crossover gate.

4.3.3 Timed crossover possibility : P-complete neighborhoods

Neighborhoods studied in this subsection :



We now prove that some subsets of Moore neighborhood have a P-complete $\mathcal{N}$-**TIMED-PRED** problem. The proof is analogous to Theorem 4.2, taking into account the delay of the gates.

Theorem 4.23. *$\mathcal{N}$-**TIMED-PRED** is* P*-complete for neighborhoods* 111, 127, 135, 143, 151, 195, 199, 211, 215.

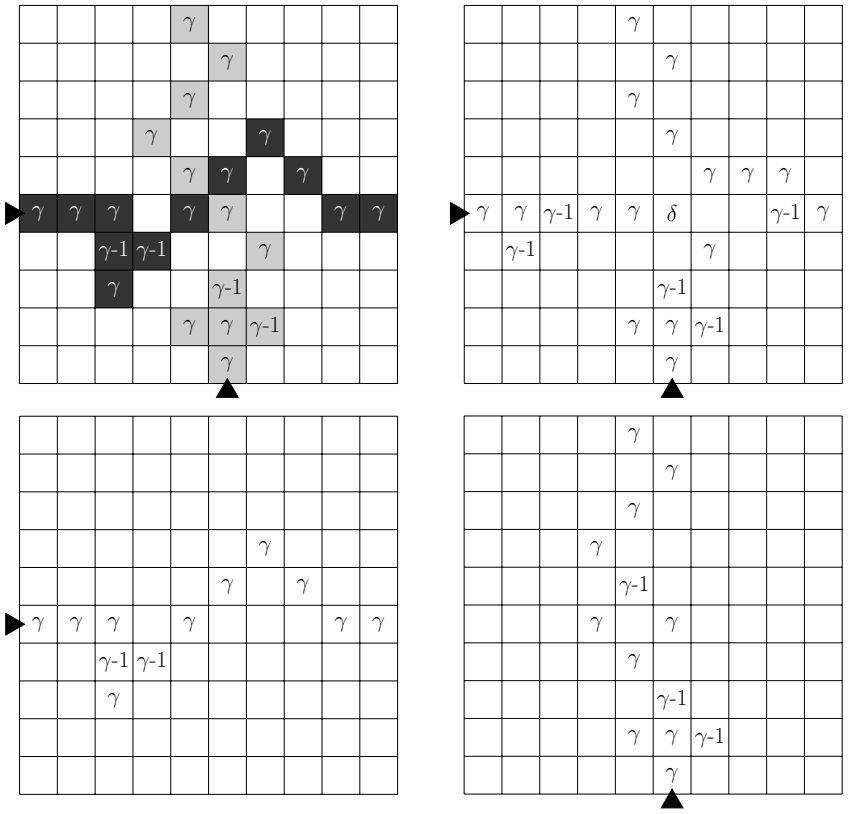


FIGURE 4.35 : Timed crossover gates (top left), *and* gates (top right, with $\delta = \theta_{\mathcal{N}} - 2$), *or* gates (top right, with $\delta = \theta_{\mathcal{N}} - 1$), horizontal diode (bottom left) and vertical diode (bottom right) for neighborhoods 111 and 127 with $\gamma = \theta_{\mathcal{N}} - 1$. The gates have size 10×10 and delay 12.

Proof. For neighborhoods 135 and 143, this is a direct consequence of Theorem 4.2.

For the other neighborhoods, we reduce from **MCVP**. For each of them, we present a set of gates having a uniform delay (it takes the same number of steps for a signal to traverse each gate). Let us denote $d \in \mathbb{N}_+$ this *delay*, and z the peculiar delay of the starting constant 1 gate (time necessary for the single grain addition at p to reach the borders of the constant 1 gate). Following Banks approach, a signal 1 is an avalanche (chain of topplings) reaching the edges of gates at time steps $(z + d \cdot T)_{T \in \mathbb{N}}$, and a signal 0 is a steady state or an avalanche reaching the edges of gates on time steps $z + d \cdot T + i$ with $i > 0$ (retards may accumulate). One can check on the layout of the **MCVP** instance on the grid (Figure 4.3), that following a single grain addition, all the variable signals start synchronized (with avalanches for 1 signals, and steady wires for 0 signals). The uniform delay on all gates ensures that signals keep synchronized, because gates are placed on the diagonal : the two input signals are at the same

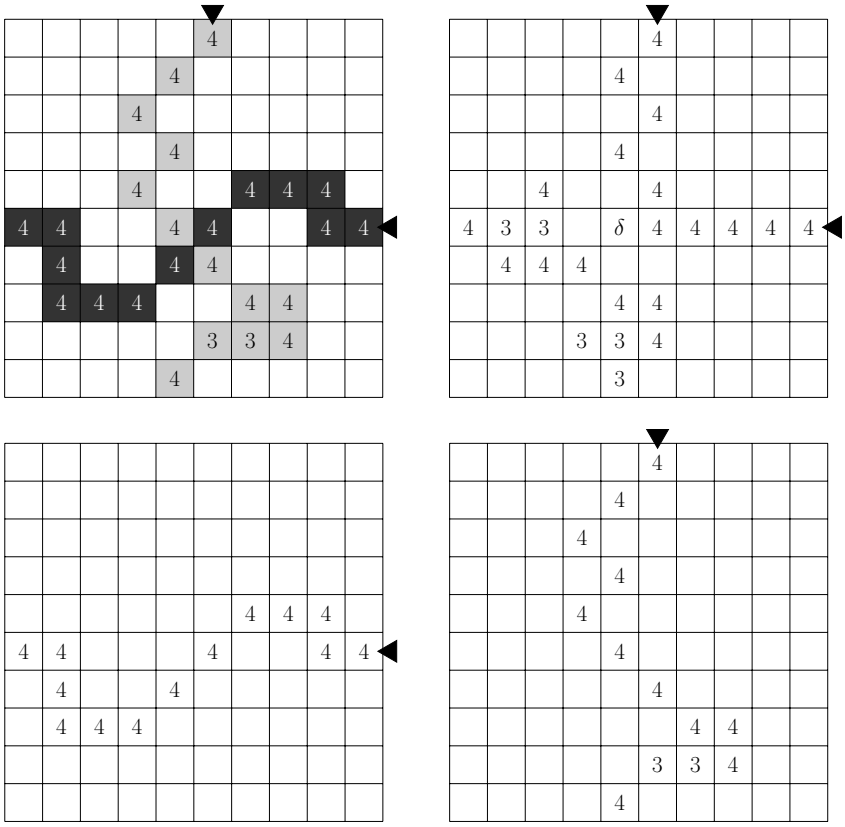


FIGURE 4.36 : Timed crossover gate (top left), *and* gate (top right, with $\delta = 3$), *or* gate (top right, with $\delta = 4$), horizontal diode (bottom left) and horizontal diode (bottom right) for neighborhood 151. The gates have size 10×10 and delay 13.

distance from the origin. Every wire follows a shortest path in the grid toward the questioned cell. The latter therefore topples at time step $z + d \cdot U + y$, with U the distance in the grid from the origin (top left cell) to the questioned cell and y for its inner mechanism (precise distance to q), if and ony if the **MCVP** outputs a signal 1.

The set of gates are presented as follows :

- on Figure 4.35 for neighborhoods 111 and 127,
- on Figure 4.36 for neighborhood 151,
- on Figure 4.37 for neighborhoods 195, 211, 215 and 199.

For each neighborhood we present a crossover gate, an *and* gates, an *or* gate, and diodes. The constant 1 gate, turns and signal-duplication are derived from the *or* gate. The constant 0 gate is again empty. ■

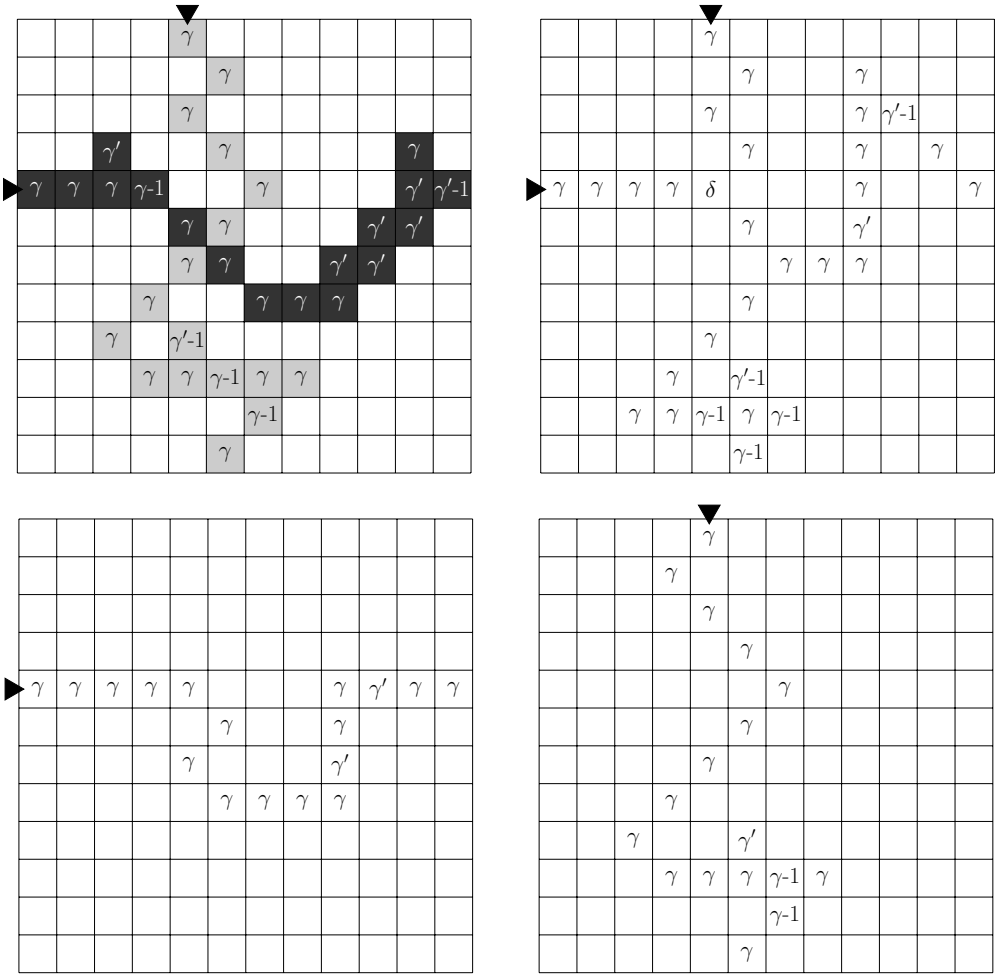
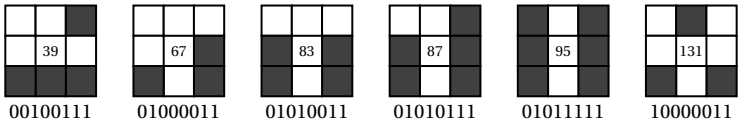


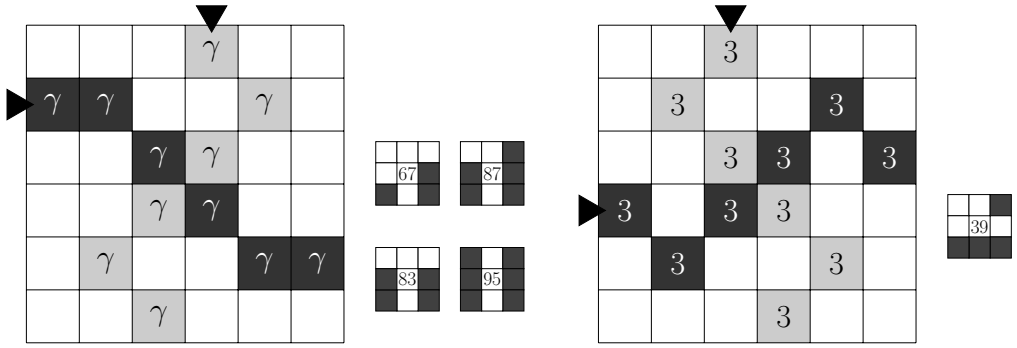
FIGURE 4.37 : Timed crossover gates (top left), *and* gates (top right, with $\delta = \theta_{\mathcal{N}} - 2$), *or* gates (top right, with $\delta = \theta_{\mathcal{N}} - 1$), horizontal diode (bottom left) and vertical diode (bottom right) for neighborhoods 195 (with $\gamma' = \theta_{\mathcal{N}} - 1$), 199 (with $\gamma' = \theta_{\mathcal{N}} - 2$), 211 (with $\gamma' = \theta_{\mathcal{N}} - 1$) and 215 (with $\gamma' = \theta_{\mathcal{N}} - 2$), with $\gamma = \theta_{\mathcal{N}} - 1$. The gates have size 12 × 12 and delay 17.

4.3.4 Timed crossover possibility : delay issue

Neighborhoods studied in this subsection :



For multiple neighborhoods, we are able to construct a timed crossover gate, and even *and* and *or* gates, but we are faced with an issue when trying to plug these gates together. Timed crossover gates for neighborhoods 39, 67, 83, 87 and 95 are presented



(a) Neighborhoods 67, 83, 87, and 95 (b) Neighborhoods 39

FIGURE 4.38 : Neighborhoods next to their timed crossover gate. Each non-empty cell contains $\gamma = \theta_{\mathcal{N}} - 1$ grains. The triangles indicate the direction in which the cells are toppled.

on Figure 4.38, and a timed crossover gate for neighborhood 131 is presented on Figure 4.34.

Let us focus on neighborhood 39 as an example. On Figure 4.39 we present an almost complete set of gates, of size 11×11 with delay 11. Inputs are located on the *west* and *north* borders, and outputs on the *east* and *south* borders. The north input and south output are identicaly located in every gate, and there are two possibles locations of the west input and east output (size 11 is odd) : either exactly in the middle of their respective borders (label A on Figure 4.39), or one cell above (label B). They are meant to be plugged alternatively to each other horizontaly (adapting the construction from Figure 4.3), following a pattern of mono-labeled columns of gates. There are :

- *wires* (also *diodes*) labeled A and B,
- a *crossover* gate labeled A,
- an *or* gate labeled B,
- an *and* gate labeled B,
- a *constant* 1 gate labeled B is derived from the *or* gate,
- a *signal-duplication* gate labeled B is derived from the *or* gate,
- *turn* gates labeled B are derived from the *or* gate,
- a *constant* 0 gate is empty (labeled A and B).

Only a pair of gates is missing : *turn* gates labeled A. If turn gates labeled A exist, then we can connect all these gates together following this mono-labled column approach (using zipzags to switch label) and prove the P-completeness of $\mathcal{N}$-**TIMED-PRED**. However, we show below that turn gates labeled A do not exist : the distance from the west input corresponding to label A to the south output is incompatible with the

movements possible under a delay of 11. We do not know how to proceed with this reduction without turn gates labeled A.

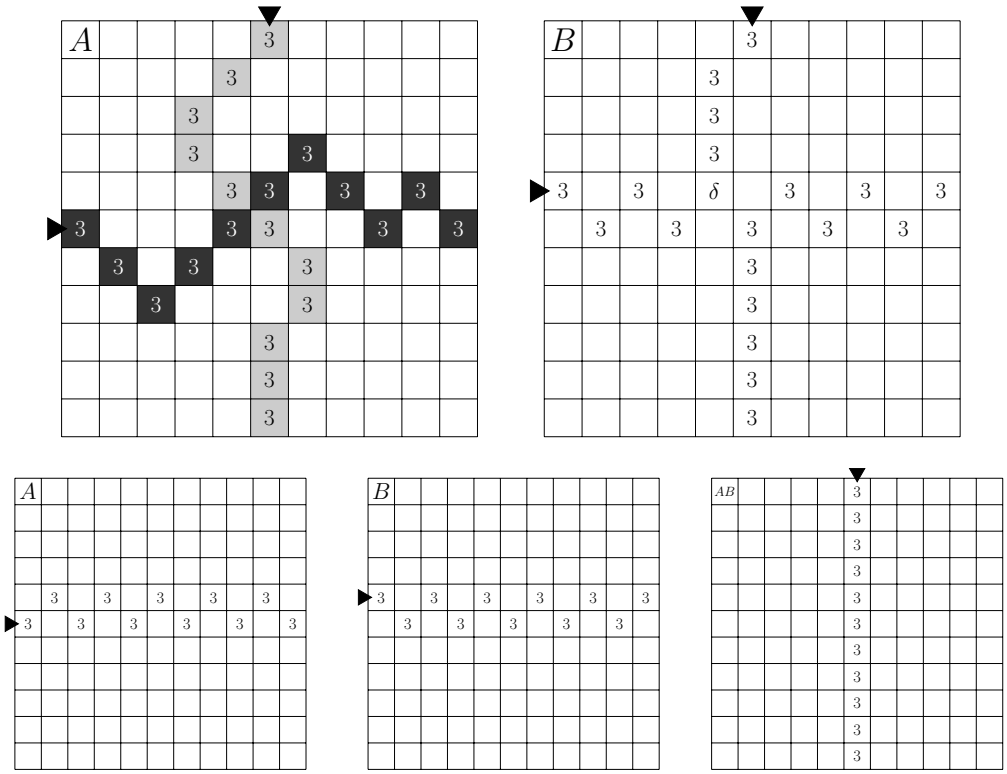


FIGURE 4.39 : An almost complete set of timed gates of size 11 × 11 and delay 11 for neighborhood 39. Timed crossover gate (top left, labeled A), *or* gate (top right, with $\delta = 3$, labeled B), *and* gate (top right, with $\delta = 2$, labeled B), horizontal diodes (bottom left and center, resp. labeled A and B) and vertical diode (bottom right, labeled both A and B).

The synchronization issue is explained by the fact that neighborhoods 39, 67, 83, 87, 95 and 131 are all in one of the two following cases :

- not having cell *north* nor *south* in the neighborhood,
- not having cell *west* nor *east* in the neighborhood.

Consequently, at each step of the avalanche process (chain of reaction of topplings), a signal must move along the perpendicular direction. For simplicity let us assume that neither cell *west* nor *east* are in the neighborhood, so that a signal must move along the y axis at each step. This induces a parity for the timestamps of signals : they have even timestamps on the cells $X_2 = \{(x, y) \in \mathbb{Z}^2 \mid x \text{ is even}\}$ and odd timestamps on the other cells $\mathbb{Z}^2 \setminus X_2$, or odd timestamps on X_2 and even timestamps on $\mathbb{Z}^2 \setminus X_2$.

It turns out that the *or/and* and *crossover* gates presented on Figure 4.39 use a different combination of parities for the two input signals, because :

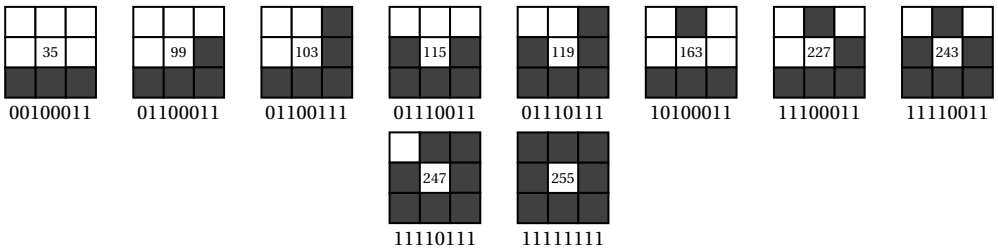
- *or/and* gates require the two signals to *meet* at synchronized timestamps,
- *crossover* gates require the two signals to *be adjacent along the y axis* at synchronized timestamps.

This leads to an incompatiblity in the synchronization between those gates for neighborhood 39. Neighborhoods 67, 83, 87 and 95 face a symmetric issue.

The case of 131 is different. It does not have the cell *west* nor *east* in the neighborhood, therefore it has a parity along the y axis, but the *crossover* gate requires the two signals to be adjacent along the x axis (in order to be in the situation of Figure 4.5). So there is no apparent incompatibility. Nonetheless, we were not able to solve the synchronization issue for neighborhood 131.

4.3.5 Timed crossover impossibility : conjectured

We conjecture that the following neighborhoods do not admit a timed crossover gate :



The difficulty to design a timed crossover gate for these neighborhoods lies in the fact that, when one uses the crossing constraint illustrated on Figure 4.5, it is not possible to take advantage of the diagonal neighbors (in terms of delay) because the cell in-between the two diagonals is also an out-neighbor. We conjecture that this prevents to construct timed crossover gates for these neighborhoods.

4.4 Chapter Conclusions and Future Work

Crossover gate and prediction problem. We have performed a systematic study of crossover possibilities for sandpiles on two-dimenstional grids $\mathbb{Z}^2$ within Moore neighborhood, and have encoutered cases with is a P-complete prediction problem. The correspondance between the existence of a crossover gate and the P-completeness of $\mathcal{N}$**-PRED** is sharp. In Section 4.2 we have established that, among the 255 subsets of Moore neighborhood (the results are summarized on Table 4.2) :

- 22 do not span $\mathbb{Z}^2$ and are therefore not relevant to our study,
- 12 admit a crossover gate and have a P-complete prediction problem $\mathcal{N}$**-PRED**,
- 99 have a planar sandpile structure and therefore do not admit a crossover gate,

- the 123 remaining neighborhoods (including Moore) do not admit a crossover gate.

We can observe that crossover impossibility is not inherited when considering the subsets of a given neighborhood. The conditions for the possibility or impossibility of crossover may be hard to establish (*e.g.* neighborhood 39 occupies 16 pages). This sheds some more light on the subtelty of embedding computation within sandpiles.

Let us recall that the crossover impossibility is also subject to the radius of the neighborhood. For example it is known by GAJARDO et al. 2006 that von Neumann of any radius $r \geq 2$ admits a crossover gate and therefore has a P-complete $\mathcal{N}$**-PRED** problem. It is also known by NGUYEN et al. 2018 that there exists a radius $r_0 \in \mathbb{N}_+$ such that Moore of any radius $r \geq r_0$ admits a crossover gate.

In a broader perspective, given a neighborhood, it is not known yet whether an algorithmic procedure exists to decide the existence of a crossover gate (the problem is semi-decidable). It is also not known whether there are infinitely many neighborhoods (among those spanning $\mathbb{Z}^2$) that do not admit a crossover gate. Indeed, it may be the case that a bound $B \in \mathbb{N}$ exists such that every neighborhood $\mathcal{N}$ of size $|\mathcal{N}| \geq B$ admit a crossover gate.

The impossibility of a crossover gate means that Banks approach fails at proving P-completeness by reduction from monotone circuit value problem (**MCVP**). Nevertheless, it leave open the possibility of a completely different approach of signal encoding in sandpile dynamics, which is still to be discovered DELORME et al. 2002. Neither does it hints at an obvious NC algorithm. Indeed, a naive reduction to monotone planar circuit value problem (**MPCVP**) fails because it requires to encode the integer value of the sand content at each cell on multiple bits (at least 3 of them for values $0,1,2,3$ and toppling state), so each undirected edge of the sandpile graph would induce a non-planar $K_{3,3}$ (complete bipartite graph of two times 3 vertices). There is no straitghforward way of guessing the direction of these edges (in order to make it acyclic and reduce the number of arcs), which would itself require to predict the effective causality among neighboring cells during the dynamics of sandpiles.

We also leave open the graals on von Neumann and Moore. Are $\mathcal{N}_{\mathtt{vn}}$**-PRED** and $\mathcal{N}_{\mathtt{m}}$**-PRED** in NC? P-complete? Neither? The first alternative requires a finer understanding of the fractal structure of sandpile dynamics, which is roughly characterized in PEGDEN et al. 2013; LEVINE et al. 2016 in terms of Apolonian circle packing. The second alternative would be a great advance in the understanding of how complex two-dimensional von Neumann and Moore neighborhoods are, in an algorithmically effective sense. The third alternative could correspond to an intermediate class, whose existence is proven in VOLLMER 1991. Let us also remark the two problems may not have the same complexity.

Timed crossover gate and timed prediction problem. In Section 4.3 we have considered the timed version of the sandpile prediction problem, asking whether, following a single grain addition on a stable configuration, a questionned cell will topple or not at a precise time step $t \in \mathbb{N}$ which is part of the input (problem $\mathcal{N}$-

TIMED-PRED). To study this problem, we have introduced the notion of timed firing graph, and timed crossover gate. The timed crossover gate allows for each of the two input cells to trigger an avalanche eventually toppling both output cells, provided there is a strictly positive retard in the output signal in the incorrect direction (not correspond to a crossing of two signal, *e.g.* from *north* to *south* the avalanche is allowed to topple the *east* cell if the delay of the signal from *north* to *south* is striclty smaller than the delay of the signal from *north* to *east*). This difference of delay aims at being embedded in a reduction from **MCVP** to $\mathcal{N}$**-TIMED-PRED** problem, according to the construction depicted on Figure 4.3 (which has been designed with this purpose). It requires to have *wire*, *turns*, *and* and *or* gates with adequate delays, and with input and output cells that plug well to each other in order to keep the signals synchronized. Our results are partial; among the 255 subsets of Moore neighborhood (the results are summarized on Table 4.3) :

- 22 do not span $\mathbb{Z}^2$ and are therefore not relevant to our study,
- 52 admit a complete timed gates toolkit (including a timed crossover gate) and therefore have a P-complete timed prediction problem $\mathcal{N}$**-TIMED-PRED**,
- 99 have a planar sandpile structure and therefore do not admit a timed crossover gate,
- 34 admit a timed crossover gate, but one is faced with an issue in the delays when connecting the gates together,
- the 49 remaining neighborhoods (including Moore) are conjectured to do not admit a timed crossover gate.

Remark that a crossover gate is also a timed crossover gate, hence the 12 neighborhoods having a P-complete $\mathcal{N}$**-PRED** problem are part of the 52 neighborhoods having a P-complete $\mathcal{N}$**-TIMED-PRED** problem. It is notable that 40 neighborhoods (from 7 equivalence classes) do not admit a crossover gate, but do admit a complete timed gates toolkit allowing to prove the P-completeness of timed prediction problem. Intuitively, following Banks approach, for these neighborhoods the embedding of the circuit evaluation in the sandpile dynamics is possible when one follows precisely the time steps, but asymptotically the sandpile will not stay in a configuration where the result of the circuit can be read out.

The precise characterization of the neighborhoods for which it is impossible to follow Banks approach in proving the P-completeness of $\mathcal{N}$**-TIMED-PRED** is still open, and does not seem to be captured by the existence of timed crossover gate as defined in Section 4.1. Proving that timed crossover gate do not exist for von Neumann and Moore neighborhoods would reveal more obstacles to the embedding of efficient computation within them.

Non-uniform sandpiles. Alternative approaches can be explored by relaxing the definition of the conventional sandpile CA. We propose two potential directions, depicted in Figure 4.40, for which we present the respective crossover gates. On the left hand side, we show an example of a neighborhood of size 3 (neighborhood 67 following our encoding) with a non-uniform distribution of sand grains. Upon reaching 5 sand grains, a cell topples and distributes 1 grain to its east neighbor, 2 grains to its south-east neighbor and 2 grains to its south-west neighbor. Defining all the necessary circuitry for applying the Banks approach for this neighborhood is not particularly challenging, therefore one can prove that this setup has a P-complete prediction problem. On the other side, the illustration on the right introduces a sandpile CA where cells may have one of three different neighborhoods. Here the neighborhoods are of size 1 (neighborhoods 1, 2 and 64) then the toppling threshold is also 1. Although the combination of these three neighborhoods allows to design a crossover gate, we conjecture that it is not possible to conceive an *and* gate, leading to the failure of Banks approach.

It is worth noting that the neighborhoods 1, 2, and 64 can be considered as a decomposition of neighborhood 67 or, similarly, one can say that neighborhood 67 can be composed with neighborhoods 1, 2, and 64. This observation, coupled with the existance of a crossover gate for both previous examples, may prompt to exploring how this notion of composition and decomposition could give insights into whether a sandpile CA admits a crossover gate.

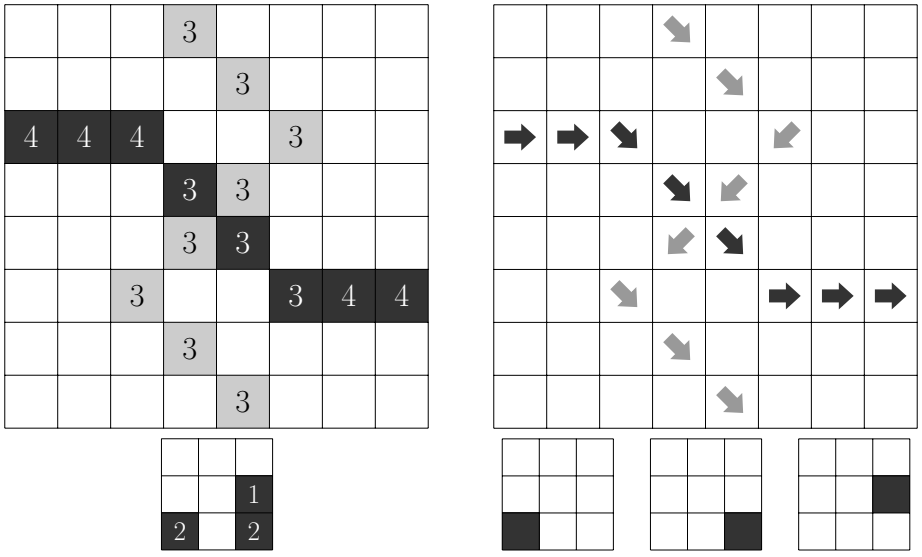


FIGURE 4.40 : Left : Example of a crossover gate using a non-uniform distribution of grains. Note that in this case the threshold is $\theta = 5$. Right : Example of a crossover for a non-uniform distribution of neighborhood using three neighborhoods of size 1. Here the threshold is $\theta = 1$ and the arrows point at the out-neighbor of the cell. The blank cells can have any of the three neighborhoods. All the cells have 0 grains.

5 Discussions

Throughout this thesis work, we gave insights on how difficult it is to predict the majority rule and the sandpile CA under several settings. The main idea of studying the prediction problem, in addition to its theoretical interest, is that it gives a notion of how expensive it is to simulate these cellular automata in terms of space and time.

To achieve this task, Chapter 1 serves as the theoretical ground for the entire thesis. We lay out all the necessary definitions, establishing a framework for understanding the subsequent results. This includes the general framework for cellular automata and computational complexity theory, providing a clear and concise basis for the rest of this work. Primordially, this Chapter introduces the central components of the thesis, the Majority Cellular Automata (MCA) and the sandpile CA. By comprehensively defining these models, the chapter sets the stage for a thorough examination of their computational complexities.

Then, in Chapter 2 a review of the existing literature is presented, offering a panoramic view of the current state of the art. The focus is on the classification of the prediction problem for various MCA and sandpile CA variants, as well as the exploration of their intricate relationships. This chapter not only consolidates the foundational knowledge within the field but also provides a crucial backdrop for understanding the gaps and opportunities that drive the subsequent research.

Consequently, in Chapter 3 we show our contributions on the matter of the majority cellular automata by studying two novel variants. Firstly, we introduce the concept of heterogeneous majority cellular automata, revealing an NL prediction problem when the dimensionality is equal to 1. However, as the dimensionality escalates, it becomes P-complete. These results is related with what appears to be one of the most impactful variables in the realm of prediction problems : dimensionality. There is a clear pattern : as the dimensionality increases, embedding computation can be done more efficiently. Furthermore, within the same chapter, we delve into the freezing dynamics of L-shaped majority cellular automata, establishing a distinct dichotomy based on the neighborhood size. Specifically, the smallest L-shaped neighborhood exhibits an associated NC prediction problem, while any larger neighborhood corresponds to a P-complete prediction problem. This dichotomy holds in this case; however there is no clear correlation between the size of the neighborhood and the computational complexity of the prediction problem. For example, in the context of sandpiles, neighborhood 135 of size 4 has a P-complete prediction problem while neighborhood 255 of size 8 does not admit a crossover gate. Perhaps the dichotomy holds true in more general settings after a certain threshold size, in the fashion of NGUYEN et al. 2018.

In Chapter 4, our contribution to the examination of sandpile prediction involves an

exhaustive exploration of both prediction and timed prediction problems, spanning all possible subsets within the Moore neighborhood. Notably, considering the well-established constraints of the Moore and von Neumann neighborhoods, which do not permit a crossover gate, it is remarkable to discover that a dozen of 2-dimensional subsets within the Moore neighborhood exhibit a P-complete prediction problem. We not only establish the P-complete nature of these subsets but also present evidence indicating lower prediction complexities for all other sub-neighborhoods within $\mathcal{N}_{\mathtt{m}}$, as none of them admit a crossover gate. This exhaustive analysis extends to the timed prediction problem, where we demonstrate that 52 neighborhoods are associated with a P-complete problem. Intriguingly, 34 neighborhoods permit a timed crossover but face challenges related to signal coordination. The remaining neighborhoods are either planar or do not permit a crossover gate. These results suggest an intuition that prediction problems are computationally easier than their timed counterpart. However there is a lack of evidence for ensuring such a strong statement. Considering the latter, it would be surprising to find a CA whose prediction problem is harder than its timed counterpart.

Perspectives

Drawing from the specific inquiries presented in each chapter, this section serves to provide a broader perspective on our work. During this thesis, our exploration of majority cellular automata and the sandpile CA has been conducted separately. However, it is crucial to emphasize that the selection of these two classes of cellular automata was not arbitrary. Instead, it was guided by an seemingly hidden connection between them : their prediction problems exhibit similar patterns. Specifically, for dimensions equal to 1, both majority cellular automata and the sandpile CA feature prediction problems classified within NC. As the dimensionality surpasses 2, these problems escalate to P-complete complexities. Intriguingly, the prediction problems for both become open questions in the bi-dimensional case. Additionaly, as previously mentioned, it is known that for both models, the time required for reaching an attractor is upper-bounded by a polynomial. Consequently, the prediction problems can be solved in polynomial time for the 2-dimensional grid. This observation prompts a question : is the complexity of prediction problem for 2-dimensional MCA and sandpiles somewhere between NC and P-complete, suggesting a form of P-intermediate problem? Perhaps the prediction problem can offer us very deep insights into complexity theory.

Another factor to consider, as mentioned in Chapter 2, is the capability of the 2-dimensional MCA to simulate the freezing version of the 2-dimensional sandpile. Then one can ask, is there a more precise sense of equivalence between these two models (the MCA and the sandpile CA) such that solving the prediction problem for one model concurrently resolves the prediction problem for the other? Perhaps the already known simulation, *i.e.* MCA simulating freezing sandpiles, provides a reduction between the prediction problems of these two variants.

The questions presented above in this section are undeniably very general, demanding considerable scientific effort and likely years of study (if provable). Nevertheless, within the scope of this thesis, there are also short/mid-term directions to pursue. One of them is to determine what is the impact on complexity, measured through the complexity of the prediction problem or with some alternative metrics, when mixing rules. It is quite surprising that we can determine the complexity of the prediction problem for the 2-dimensional heterogeneous majority cellular automata, a task that has remained an open problem for decades in the case of the majority rule.

Additonaly, there are several ways in which this topic can be further extended, like classifying families of CA, such as we did with the L-shaped freezing majority cellular automata, where we narrow down the study of the prediction problem to a restricted yet infinite subset of freezing MCA. It is worth noting that apparently as the generality of the family increases, the difficulty of substantiating any claims or proofs typically intensifies. As an example, solely considering the non-freezing L-shaped MCA, classifying the complexity of the associated prediction problem becomes a challenging task. The latter, nevertheless, is a potential research direction.

Finally, another interesting topic to be explored is the difference between prediction problem and its timed counterpart. Direct questions arise : Do they share the same computational complexity? Does their complexity hinge on the specific cellular automaton under examination, and if so, what are the underlying reasons?

Bibliography

[ADG02a] Zvia Agur, Yoaz Daniel, and Yuval Ginosar. “The universal properties of stem cells as pinpointed by a simple discrete model”. In: *Journal of mathematical biology* 44.1 (2002), pp. 79–86 (cit. on p. 13).

[ADG02b] Zvia Agur, Yoaz Daniel, and Yuval Ginosar. “The universal properties of stem cells as pinpointed by a simple discrete model”. In: *Journal of mathematical biology* 44.1 (2002), pp. 79–86.

[Anc+22] Camilla Ancona, Francesco Lo Iudice, Franco Garofalo, et al. “A model-based opinion dynamics approach to tackle vaccine hesitancy”. In: *Scientific reports* 12.1 (2022), pp. 1–8 (cit. on p. 13).

[BTW87] P. Bak, C. Tang, and K. Wiesenfeld. “Self-organized criticality: An explanation of the 1/f noise”. In: *Physical review letters* 59.4 (1987), p. 381 (cit. on p. 13).

[Ban71] E. R. Banks. “Information processing and transmission in cellular automata”. PhD thesis. Massachusetts Institute of Technology, 1971 (cit. on pp. 14, 24, 30).

[CFL09a] C. Castellano, S. Fortunato, and Lo.. “Statistical physics of social dynamics”. In: *Reviews of modern physics* 81.2 (2009), p. 591 (cit. on p. 13).

[CFL09b] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. “Statistical physics of social dynamics”. In: *Reviews of modern physics* 81.2 (2009), p. 591.

[CLR79] John Chalupa, Paul L Leath, and Gary R Reich. “Bootstrap percolation on a Bethe lattice”. In: *Journal of Physics C: Solid State Physics* 12.1 (1979), p. L31 (cit. on p. 28).

[DM02] M. Delorme and J. Mazoyer. “Signals on Cellular Automata”. In: *Collision-based Computing*. Ed. by Andrew Adamatzky. Springer-Verlag, 2002, pp. 231–275. ISBN: 978-1-4471-0129-1 (cit. on pp. 24, 30, 109).

[Dha90] Deepak Dhar. “Self-organized critical state of sandpile automaton models”. In: *Physical Review Letters* 64.14 (1990), p. 1613 (cit. on p. 19).

[FGM12] E. Formenti, E. Goles, and B. Martin. “Computational Complexity of Avalanches in the Kadanoff Sandpile Model”. In: *Fundamenta Informaticae* 115.1 (2012), pp. 107–124. DOI: 10.3233/FI-2012-643 (cit. on p. 30).

[FP19] E. Formenti and K. Perrot. "How Hard is it to Predict Sandpiles on Lattices? A Survey". In: *Fundamenta Informaticae* 171 (1-4 2019), pp. 189–219. DOI: 10.3233/FI-2020-1879. eprint: 1909.12150 (cit. on pp. 20, 23, 30).

[FK99] Miguel A Fuentes and Marcelo N Kuperman. "Cellular automata and epidemiological models with spatial dependence". In: *Physica A: Statistical Mechanics and its Applications* 267.3-4 (1999), pp. 471–486 (cit. on p. 28).

[GKL78] Peter Gács, G Kurdyumov, and L Levin. "One-dimensional homogeneous media dissolving finite islands". In: *Problems of Information Transmission* 14.3 (1978), pp. 92–96 (cit. on p. 42).

[GT22] Péter Gács and Ilkka Törmä. "Stable multi-level monotonic eroders". In: *Theory of Computing Systems* (2022), pp. 1–32 (cit. on p. 42).

[GG06] A. Gajardo and E. Goles. "Crossing information in two-dimensional Sandpiles". In: *Theoretical Computer Science* 369.1-3 (2006), pp. 463–469. DOI: 10.1016/j.tcs.2006.09.022 (cit. on pp. 14, 30, 53, 55, 58, 109).

[Gol+17a] E. Goles, D. Maldonado, P. Montealegre, et al. "On the Computational Complexity of the Freezing Non-strict Majority Automata". In: *Proceedings of AUTOMATA'2017*. Vol. 10248. LNCS. 2017, pp. 109–119. DOI: 10.1007/978-3-319-58631-1_9 (cit. on p. 28).

[GM97] E. Goles and M. Margenstern. "Universality of the chip-firing game". In: *Theoretical Computer Science* 172.1-2 (1997), pp. 121–134 (cit. on p. 13).

[GMP21a] E. Goles, P. Montealegre, and K. Perrot. "Freezing sandpiles and Boolean threshold networks: Equivalence and complexity". In: *Advances in Applied Mathematics* 125 (2021), p. 102161. DOI: 10.1016/j.aam.2020.102161. eprint: 2101.04204 (cit. on p. 31).

[Gol+17b] E. Goles, P. Montealegre, K. Perrot, and G. Theyssier. "On the complexity of two-dimensional signed majority cellular automata". In: *Journal of Computer and System Sciences* 91 (2017), pp. 1–32. DOI: 10.1016/j.jcss.2017.07.010 (cit. on pp. 14, 29, 30, 36, 45).

[GMT13] E. Goles, P. Montealegre-Barba, and I. Todinca. "The complexity of the bootstraping percolation and other problems". In: *Theoretical Computer Science* 504 (2013), pp. 73–82. DOI: 10.1016/j.tcs.2012.08.001 (cit. on p. 28).

[Gol+20a] E. Goles, M.-A. Tsompanas, A. Adamatzky, et al. "Computational universality of fungal sandpile automata". In: *Physics Letters A* 384.22 (2020), p. 126541. DOI: 10.1016/j.physleta.2020.126541 (cit. on pp. 14, 30).

[Gol+20b] Eric Goles, Diego Maldonado, Pedro Montealegre, et al. "On the complexity of the stability problem of binary freezing totalistic cellular automata". In: *Information and Computation* 274 (2020), p. 104535 (cit. on pp. 28, 29).

[GM14] Eric Goles and Pedro Montealegre. “Computational complexity of threshold automata networks under different updating schemes”. In: *Theoretical Computer Science* 559 (2014), pp. 3–19 (cit. on pp. 13, 29).

[GM15] Eric Goles and Pedro Montealegre. “The complexity of the majority rule on planar graphs”. In: *Advances in Applied Mathematics* 64 (2015), pp. 111–123 (cit. on p. 29).

[GMP21b] Eric Goles, Pedro Montealegre, and Kévin Perrot. “Freezing sandpiles and Boolean threshold networks: Equivalence and complexity”. In: *Advances in Applied Mathematics* 125 (2021), p. 102161 (cit. on p. 28).

[GO80] Eric Goles and Jorge Olivos. “Periodic behaviour of generalized threshold functions”. In: *Discrete mathematics* 30.2 (1980), pp. 187–189 (cit. on p. 23).

[GOT15] Eric Goles, Nicolas Ollinger, and Guillaume Theyssier. “Introducing freezing cellular automata”. In: *Cellular Automata and Discrete Complex Systems, 21st International Workshop (AUTOMATA 2015)*. Vol. 24. 2015, pp. 65–73 (cit. on p. 28).

[GHR95] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, Inc., 1995. ISBN: 0-19-508591-4 (cit. on pp. 20–22, 25, 61).

[Heg98a] R. Hegselmann. “Modeling social dynamics by cellular automata”. In: *Computer modeling of social processes* (1998), pp. 37–64 (cit. on p. 13).

[Heg98b] Rainer Hegselmann. “Modeling social dynamics by cellular automata”. In: *Computer modeling of social processes* (1998), pp. 37–64 (cit. on p. 13).

[Jáj92] J. Jájá. *An introduction to parallel algorithms*. Addison-Wesley, 1992, pp. x + 566. ISBN: 0-201-54856-9 (cit. on pp. 20, 21, 25–27).

[KT97] Ioannis Karafyllidis and Adonios Thanailakis. “A model for predicting forest fire spreading using cellular automata”. In: *Ecological Modelling* 99.1 (1997), pp. 87–97 (cit. on p. 28).

[Kůr97] P. Kůrka. “On topological dynamics of Turing machines”. In: *Theoretical Computer Science* 174.1-2 (1997), pp. 203–216 (cit. on p. 13).

[Lad75] Richard E Ladner. “The circuit value problem is log space complete for P”. In: *ACM Sigact News* 7.1 (1975), pp. 18–20 (cit. on p. 21).

[LPS16] L. Levine, W. Pegden, and C. K. Smart. “Apollonian structure in the Abelian sandpile”. In: *Geometric and Functional Analysis* 26 (2016), pp. 306–336. DOI: 10.1007/s00039-016-0358-7 (cit. on p. 109).

[Mil07] P. Miltersen. “The Computational Complexity of One-Dimensional Sandpiles”. In: *Theory of Computing Systems* 41 (2007), pp. 119–125. DOI: 10.1007/s00224-006-1341-8 (cit. on p. 30).

[MW22] A. Modanese and T. Worsch. “Embedding Arbitrary Boolean Circuits into Fungal Automata”. In: *Proceedings of LATIN'2022*. Vol. 13568. LNCS. 2022, pp. 393–408. DOI: 10.1007/978-3-031-20624-5_24 (cit. on pp. 14, 30).

[MN99] C. Moore and M. Nilsson. “The Computational Complexity of Sandpiles”. In: *Journal of Statistical Physics* 96 (1 1999). 10.1023/A:1004524500416, pp. 205–224. ISSN: 0022-4715 (cit. on pp. 13, 28, 29).

[Moo97] Cristopher Moore. “Majority-vote cellular automata, Ising dynamics, and P-completeness”. In: *Journal of Statistical Physics* 88 (1997), pp. 795–805 (cit. on pp. 14, 28, 29).

[NP18] V.-H. Nguyen and K. Perrot. “Any shape can ultimately cross information on two-dimensional abelian sandpile models”. In: *Proceedings of AUTOMATA'2018*. Vol. 10875. LNCS. 2018, pp. 127–142 (cit. on pp. 30, 58, 62, 101, 109, 112).

[PS13] W. Pegden and C. K. Smart. “Convergence of the Abelian sandpile”. In: *Duke Mathematical Journal* 162 (4 2013), pp. 627–642. DOI: 10.1215/00127094-2079677 (cit. on p. 109).

[Roy+21] S. Roy, M. Shrivastava, C. V. Pandey, et al. “IEVCA: An efficient image encryption technique for IoT applications using 2-D Von-Neumann cellular automata”. In: *Multimedia Tools and Applications* 80 (2021), pp. 31529–31567 (cit. on p. 13).

[Sch78] T. C. Schelling. *Micromotives and macrobehavior*. WW Norton & Company, 1978. ISBN: 978-0393090093 (cit. on p. 13).

[Sch06] Thomas C Schelling. *Micromotives and macrobehavior*. WW Norton & Company, 2006 (cit. on p. 13).

[Tar88] G. Tardos. “Polynomial bound for a chip firing game on graphs”. In: *SIAM Journal of Discrete Mathematics* 1.3 (1988), pp. 397–398. DOI: 10.1137/0401039 (cit. on p. 30).

[TO22] Guillaume Theyssier and Nicolas Ollinger. “Freezing, Bounded-Change and Convergent Cellular Automata”. In: *Discrete Mathematics & Theoretical Computer Science* 24 (2022) (cit. on p. 32).

[Too80] Andrei L Toom. “Stable and attractive trajectories in multicomponent systems”. In: *Multicomponent random systems* 6 (1980), pp. 549–575 (cit. on pp. 15, 42).

[Vol91] H. Vollmer. “The gap-language-technique revisited”. In: *Proceedings of CSL'90*. Vol. 533. LNCS. 1991, pp. 389–399. DOI: 10.1007/3-540-54487-9_72 (cit. on p. 109).

[Yas12] Daadaa Yassine. *Network decontamination with temporal immunity*. University of Ottawa (Canada), 2012 (cit. on p. 13).